ECE395: Ideas for Senior Projects

Jonathan Lam

09/06/21

Three potential project ideas are proposed:

1 Typed Filesystem

Modern tree-like filesystems have never enforced any structure on their contents, other than through the use of permissions. It may be useful for package management, human recall, security, and programmability that certain conventions be followed in a file hierarchy. The *nix OS tend to follow some variant of the Filesystem Hierarchy Standard, but it has no means of enforcing it; installing packages in the wrong place (whether from bad install scripts or by user error), for example, may cause compatibility problems. We propose a new FUSE layer that lies on top of a working filesystem, which enforces a (user-specified) schema. To achieve this, we also introduce new filesystem error messages (caused by lack of adherance to the schema), conflict resolution strategies, and a type hierarchy for use in the schema.

2 SVG Video Compression for Variable-Fidelity, Low-Bandwidth Video

In the COVID-19 era, we live in the age of Internet calls. These are often in the form of voice or video calls. In the latter, a common problem is that of limited Internet bandwidth, which often limits us to the former. A common use for video calls is to transmit a user's casual presence (e.g., an image of their face, facial expressions, and surroundings) rather than fine details such as text, and often users may call from their phone in an Internet-constrained environment (e.g., outdoors) – in this case, we may desire a lower-fidelity, but also low-bandwidth, video streaming solution.

We propose a novel vector-based video format for transmitting vector video in an binary SVG-equivalent format, as well as a deep-learning model to convert videos to SVG frames in real-time based on YOLO-net's real-time object detection. The model will allow tweaking of the of "compression level" (fidelity) by adjusting the output shape density.

3 An Intentional Programming Framework for Programming and SWE Education

(This is an idea for my M.Eng. thesis, but in the spirit of proposing project ideas I'd like some feedback on this as well.)

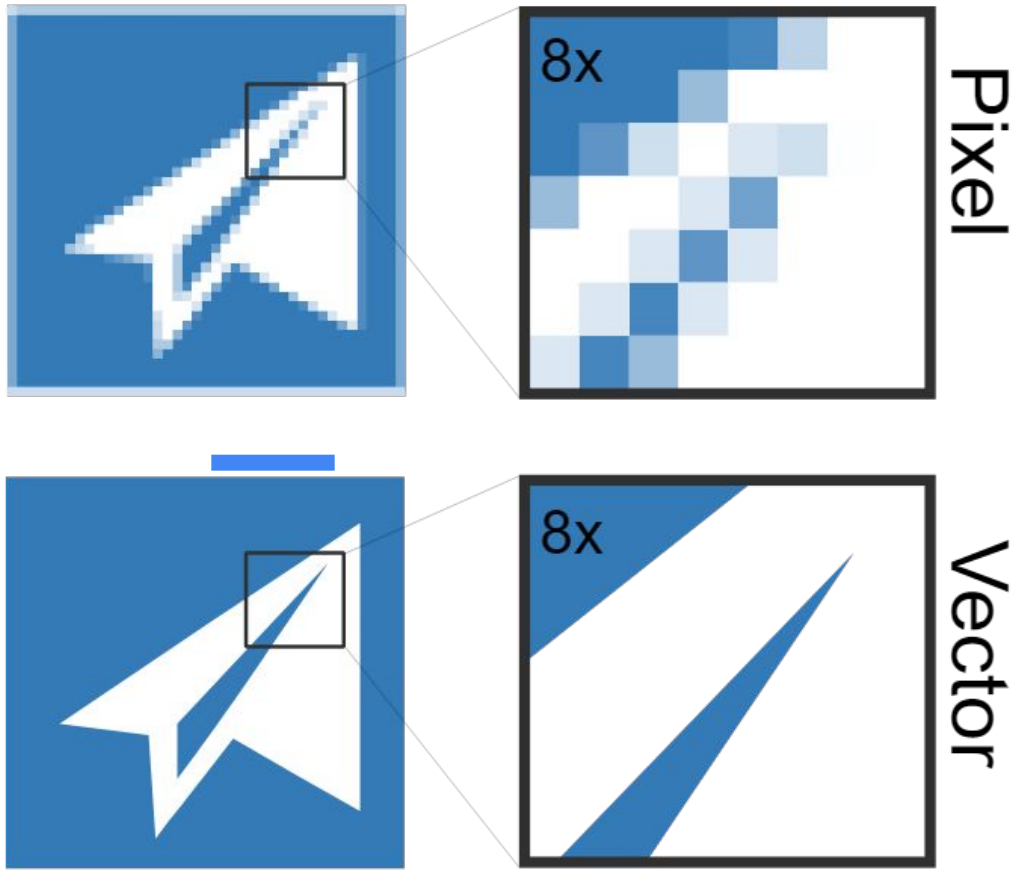
There are many common difficulties among programming students; these include remembering syntax, comprehending error messages, mapping conceptual steps into code, structuring code in a meaningful way, and recalling what a piece of code is supposed to do. While integrated development environments (IDEs), beginner-oriented domain-specific languages (DSLs) (e.g., Scratch), and beginner-oriented techniques such as gradual programming (e.g., Hedy) aid the learning experience, these techniques are still usually locked into the mindset syntax of a single language and do not enforce good programming practices such as abstraction and self-documentation.

I propose an "intent-driven" method for programming education, which involves the creation of an IDE, a DSL, and drivers to transpile the DSL into target programming languages. The DSL will follow the "intentional programming" (Simonyi 1995) programming paradigm, which is a tree-like program representation independent of programming language that allows the user to encode their "intent" at various levels of abstraction. Intents separate purpose from implementation. An intent loosely corresponds to a language-level construct like a procedure, control-flow statement, or special form, but this is transparent to the user, who sees only a building block for abstractions. In this representation, the user describes their intent in an Lisp-like syntax-less manner, and is recursively prompted to define subintents until each intent is defined in terms of primitive intents. The purpose and API of each intent must be documented ("literate programming"). The IDE manages intent definitions, enforces the intent structure, allows "zooming" of abstraction level (similar to code folding), and transpiles to target languages (and relays error messages from the target compiler). The benefits are many-fold. IP: is easier to learn than a "real" programming language; allows experimentation with language features ("genes") across programming languages; encodes SWE best-practices such as top-down design, test-driven development, and self-documentation; relates errors to functional rather than lexical location; reduces cognitive load by intuitive code-folding; encourages learning by-example rather than by-error via a community of (well-documented) user-submitted intents; promotes polyglot programming; and allows grokking overall concepts in their own words ("language-oriented programming") rather than using prescribed formal languages.

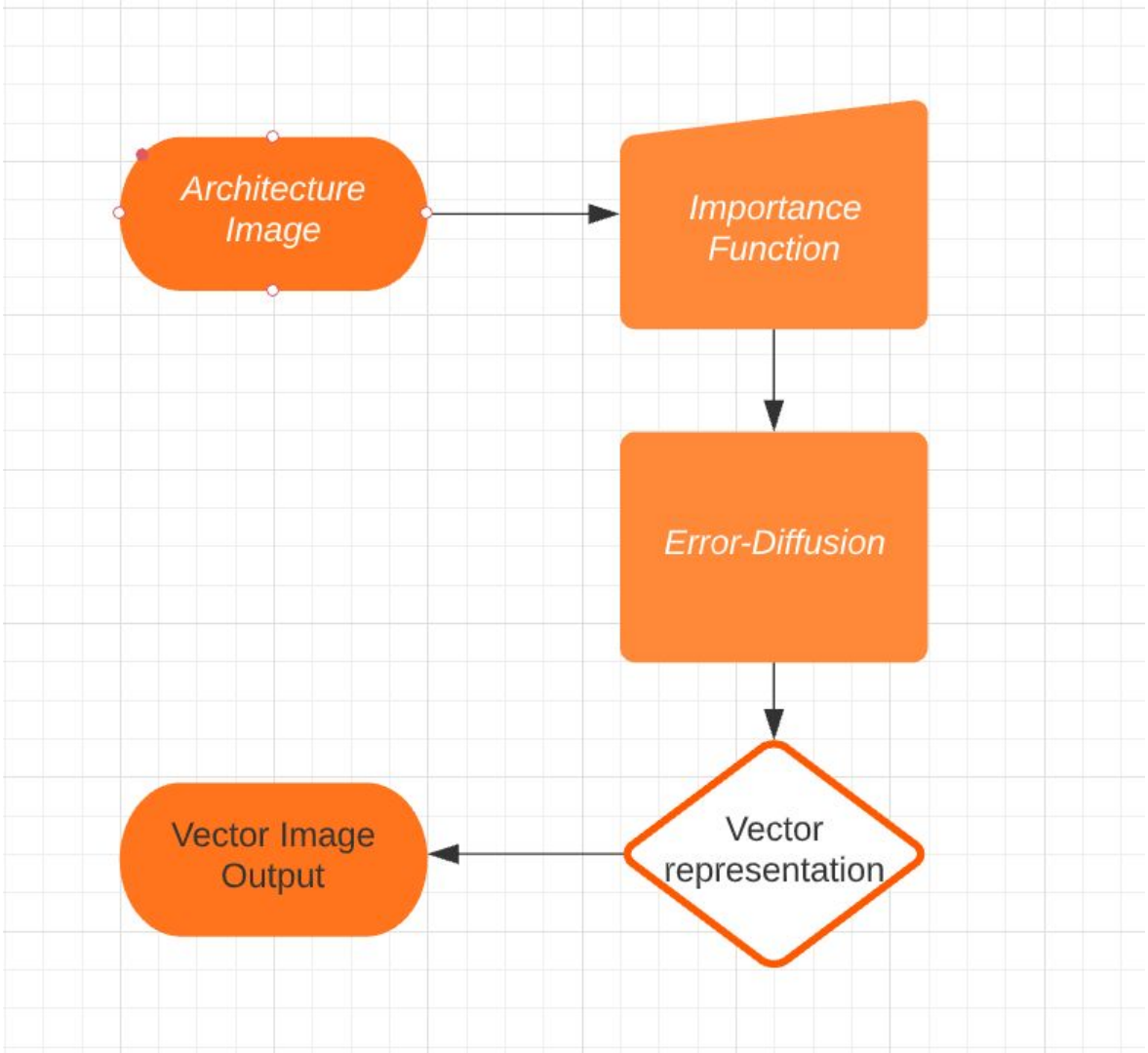
Image Vectorization
for Architecture

Derek Lee, Jon Lam, Victor Zhang

Y Vector



Structure Overview



Importance Matrix

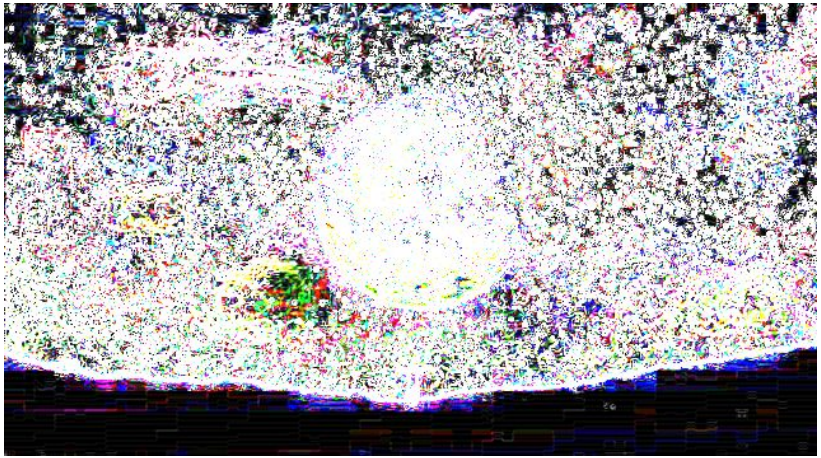
-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

2	1	0
1	0	-1
0	-1	-2

0	1	2
-1	0	1
-2	-1	0

$$F(x) = \left(\frac{x}{\max}\right)^{\frac{1}{\gamma}} * 255, \quad \gamma > 0, x \in [0, \max] \text{ and } x \in Z^{+},$$



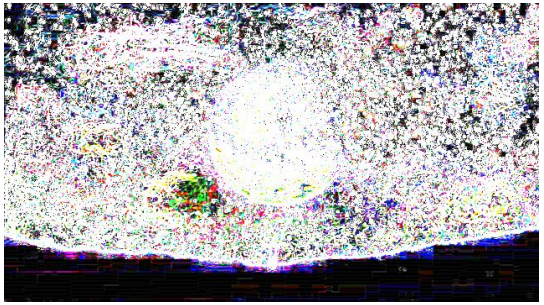
Error-Diffusion

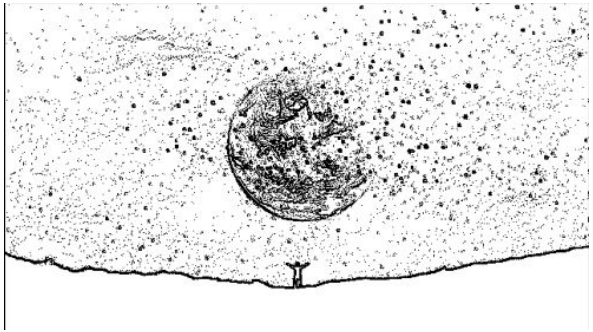
Floyd–Steinberg dithering

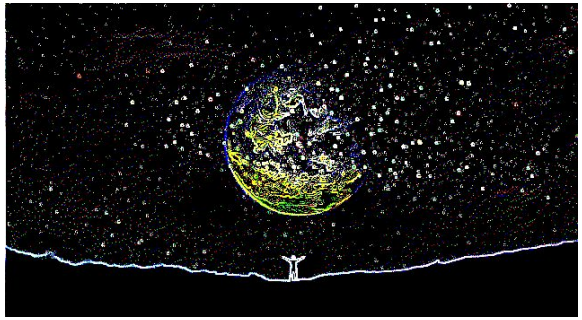
$$\begin{bmatrix} & * & \frac{7}{16} & \cdots \\ \cdots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \cdots \end{bmatrix}$$

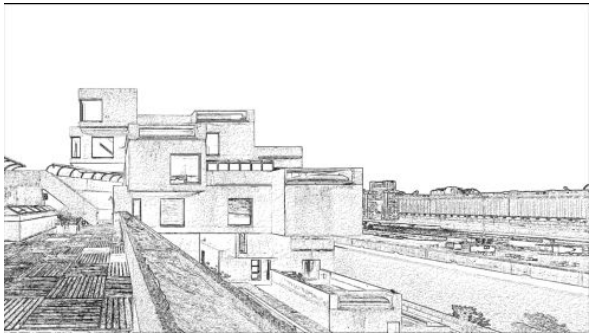
```
for each y from top to bottom do
    for each x from left to right do
        oldpixel := pixels[x][y]
        newpixel := find_closest_palette_color(oldpixel)
        pixels[x][y] := newpixel
        quant_error := oldpixel - newpixel
        pixels[x + 1][y    ] := pixels[x + 1][y    ] + quant_error × 7 / 16
        pixels[x - 1][y + 1] := pixels[x - 1][y + 1] + quant_error × 3 / 16
        pixels[x    ][y + 1] := pixels[x    ][y + 1] + quant_error × 5 / 16
        pixels[x + 1][y + 1] := pixels[x + 1][y + 1] + quant_error × 1 / 16
```

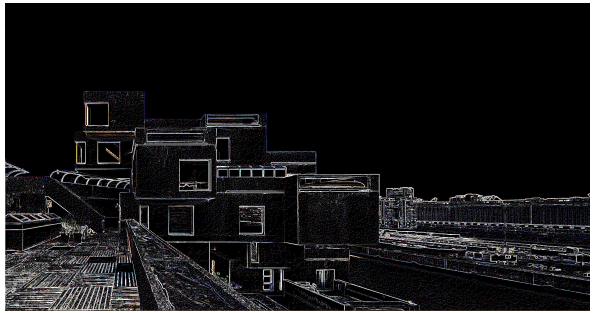
Importance -> Point Cloud (Threshold) -> Sampled Points





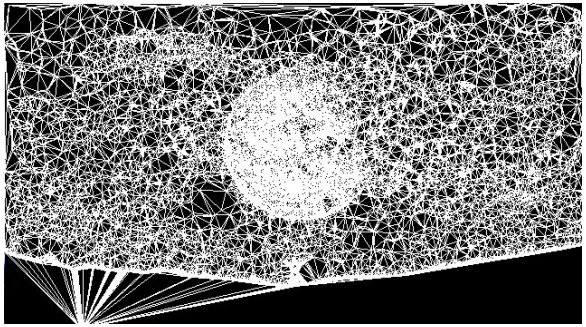


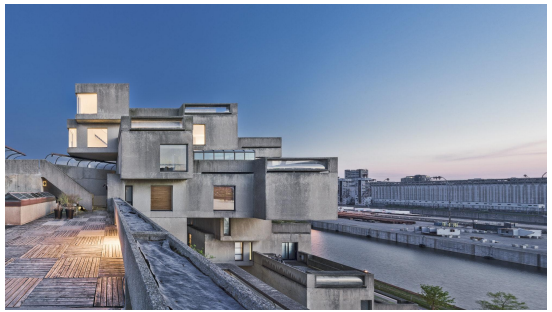


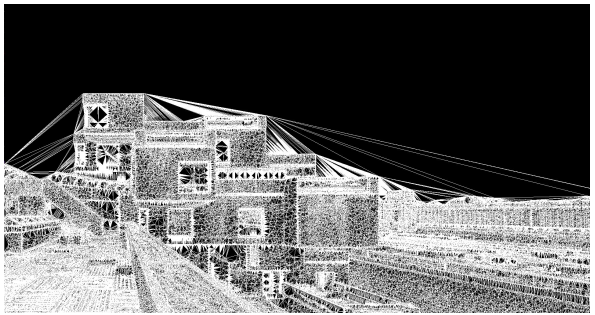


Vectorization

Current Implementation: Triangulation of Point cloud



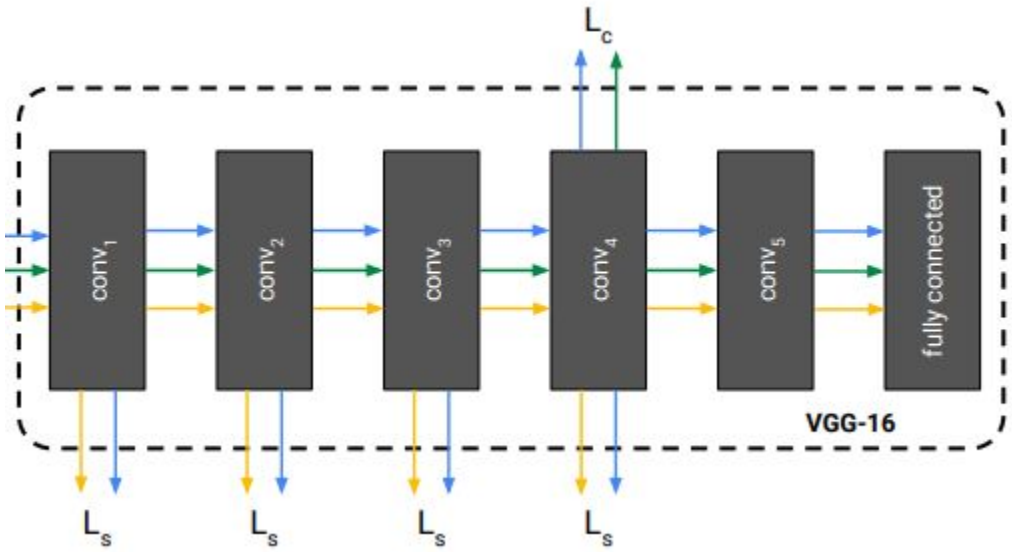




Loss Evaluation

Currently using content loss with a VGG-19

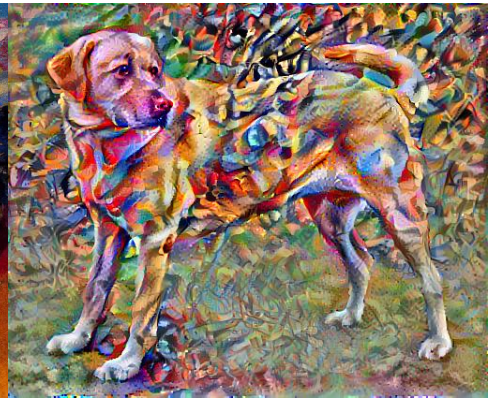
Euclidean distance between the intermediate representations of a trained classifer



Loss Evaluation

Original (Content)

Style

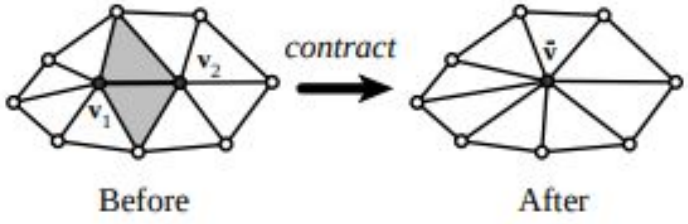


Combined

Image 1 & 2	**Loss**
Original + Combined	132352.05
Original + Style	190958.28
Style + Combined	190266.86

Future Plan

1. Implement Full SVG support
2. Mesh Simplification
3. Explore potential of NN to isolate Architecture



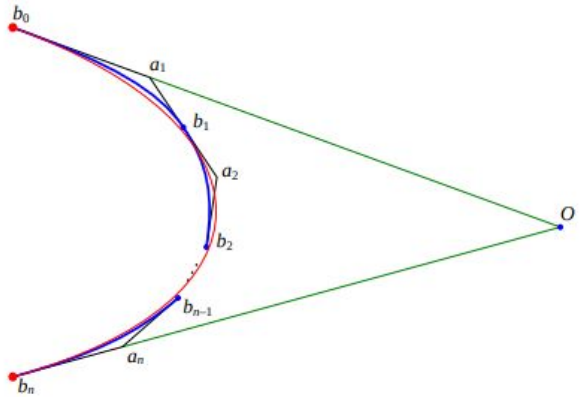


Image Vectorization for Architecture

Senior Projects Proposal

Jonathan Lam, Derek Lee, Victor Zhang

Problem Statement

Raster (pixel-based) images often contain excess information

- Preserve target information while removing excess
- Video call, (usually) only face is of interest, not background
- Geometric images (e.g., architecture), shape-based representation is much more efficient

Convert raster images to a **vector (shape-based) graphics** format?

- Can have information loss as long as target information is preserved

Proposed Solution

Improve image vectorization so that we:

- Retain most information of interest
- Remove most excess information

Use cases:

- Highly shape-based images (e.g., architectural photographs or sketches)
- Machine learning preprocessing

Background

Numerous pre-existing algorithmic approaches to vectorization

1. Edge detection & tracing (two colors only)
2. Blue-noise sampling & triangulation
3. Existing use-cases: charts, maps, clip-art, fonts

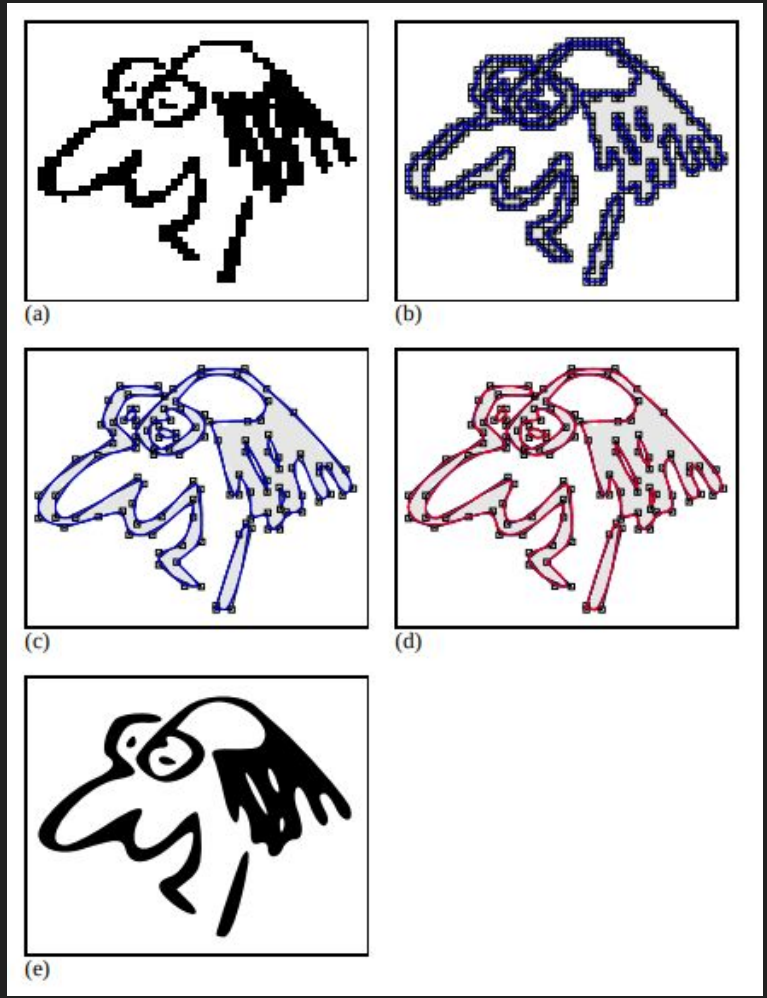
Does not appear to be many machine learning approaches

- Only for simple shapes (e.g. clip-art and fonts)

Potrace

- 2 colors only
- Shape reduction

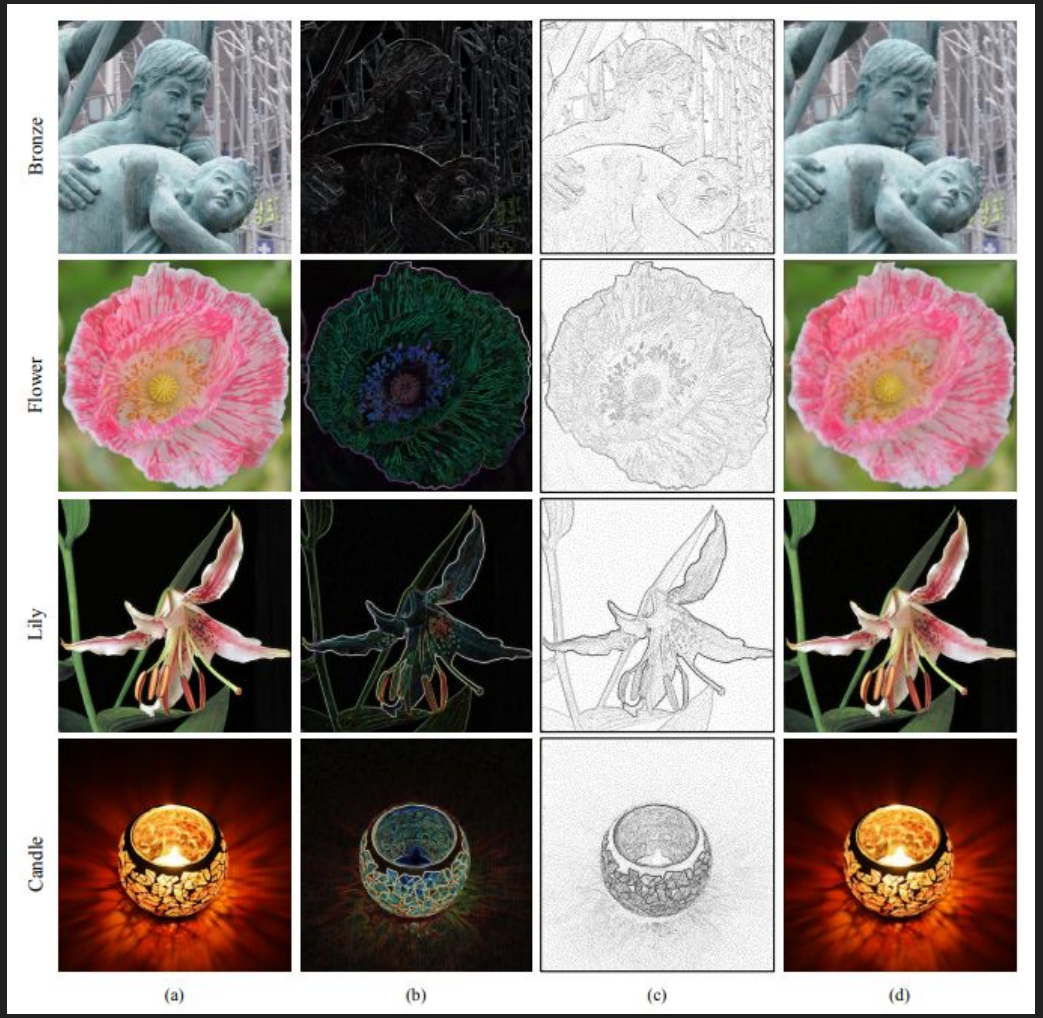
Selinger, Peter. "Potrace: a polygon-based tracing algorithm." Potrace (online), http://potrace. sourceforge. net/potrace. pdf (2009-07-01) 2 (2003).



Blue-noise sampling

- Multiple colors
- Variable resolution

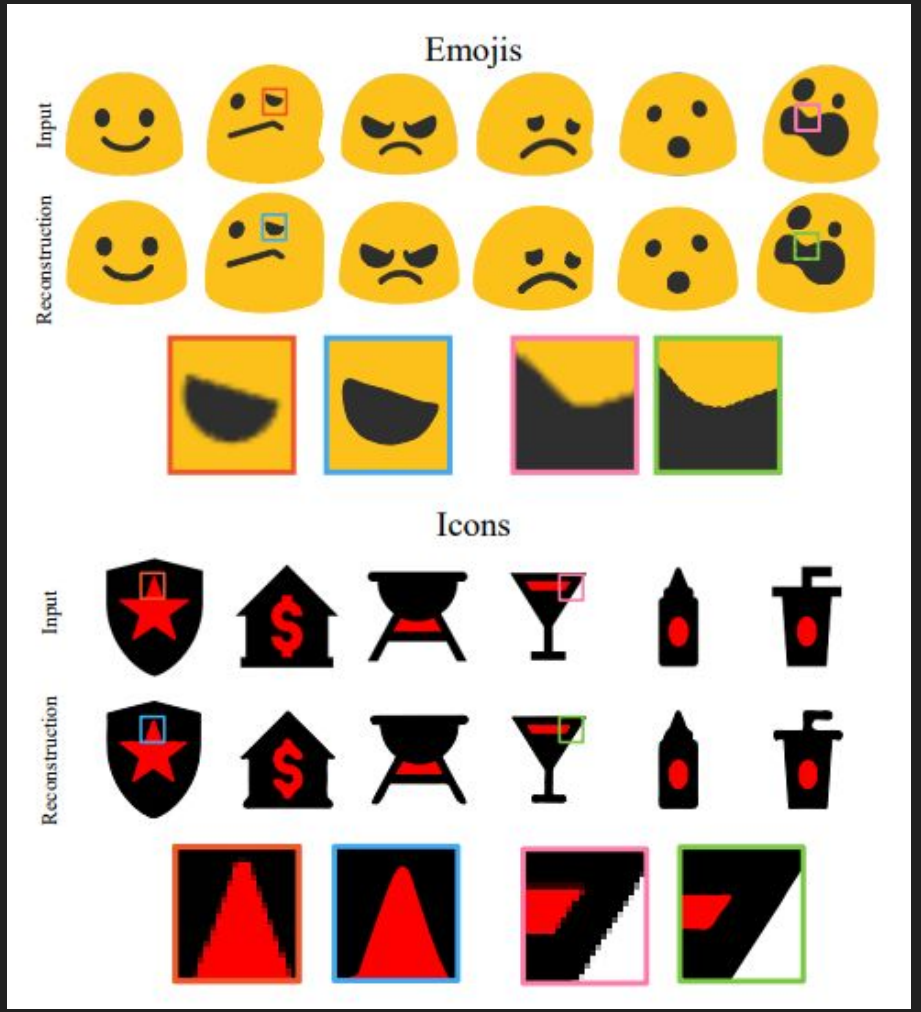
Zhao, Jiaojiao, Jie Feng, and Bingfeng Zhou. "Image vectorization using blue-noise sampling." Imaging and Printing in a Web 2.0 World IV. Vol. 8664. International Society for Optics and Photonics, 2013.



Im2Vec

- Hardcode # shapes/colors

Reddy, Pradyumna, et al. "Im2vec: Synthesizing vector graphics without vector supervision." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.



Experimental Design

1. Experiment with sampling methods
2. Two possibilities
   a. Use machine learning on the blue-noise sampled points or triangulated points
   b. Use shape reduction on the trangulated points

Project Timeline

1. Brainstorming
2. Reimplementation of previous work
3. **Project pitch**
4. First implementation
5. Iterative design process...
6. Write report

Q&A

THE COOPER UNION
FOR THE ADVANCEMENT OF SCIENCE AND ART
ALBERT NERKEN SCHOOL OF ENGINEERING

IMAGE VECTORIZATION FOR ARCHITECTURE

Jonathan Lam, Derek Lee, Victor Zhang

ECE395 Mid-year Report
Professor Sam Keene
December 2021

Contents

1 Abstract

Architecture projects often involve both large structures and detailed components, and are often highly geometric. Digital architecture designs (i.e., created through the use of CAD) are often stored in a vector (shape-based) format for convenient editing. Raster (pixel-based) images, such as photographs, are not as easily used for these purposes, which largely diminishes their usability for the design process. Our project aims to develop a image vectorization method (i.e., a tool to convert from raster to vector format) specialized towards architecture images and explores the potential of vector-based images in the architecture design process and in machine learning preprocessing.

There are several general methods for image vectorization – we propose a sampling-based vectorization method involving three steps. The image is sampled using a blue-noise sampling technique, which extracts the high frequency components in the image and filters out the less important pixels. The sampled point cloud is then simplified to reduce the number of vectors in the final drawing. Finally, the last step involves converting the point cloud to an efficient vector representation. This vector representation is saved as some vector image format, such as SVG.

2 Introduction

2.1 Goal

Our project aims to provide an end-to-end image vectorization tool. Vectorization is the process of generating a vector image that is faithful to the input raster image. We aim to specialize in architectural images specifically, which should allow us to optimize our project for this use case. Architectural images are usually highly geometric, which should allow for an efficient vector representation. In theory, a highly geometric image should have a vector representation that is more efficient than the original image. Of course, with real images, the image will not be perfectly geometric. However, we are comfortable with information loss, as long as the main content of the image is preserved; we will develop a suitable loss metric to quantify the representation efficiency and error loss.

2.2 Previous Works and Motivation

Some of the first image vectorization tools implemented edge tracing, which works well for simple shapes, and especially in black-and-white (or similarly color-thresholded) images. However, architectural images are more complex and can have wide range of colors, rendering traditional method ineffective. A new vectorization method is developed to work effectively for colored complex images, and generates vector representation that is easy to use for architects.

2.3 Methods

Multiple methods has been considered for this project, including traditional edge tracing, machine learning, and sampling. These methods are compared, and the sampling method is proven to be most effective for our use case.

2.4 Potential Applications

Our project can potentially be applied to the architecture design process, along with vector-based machine learning. One of the use cases we envision is to take a photo of an architectural design, use our project to process that image, and use the vector-based output to easily edit the image. Another potential use case is for vector-based machine learning. In computer vision, image data used as input is traditionally in raster format – there is little research performed on how well deep learning performs on vector-based image inputs. We imagine that due to the efficiency of its representation, especially for highly-geometric shapes, we may be able to have more useful information in the deep learning model from the outset. In other words, a conversion to vector-based models, in which the shapes contain meaningful information about the image, may be useful as a machine learning preprocessing step.

3 Background

3.1 Raster Graphics

Raster images are the simple matrix representation of an image. A raster image is conceptually a matrix of pixels, or a bitmap. Each pixel may contain multiple data points representing channels (colors). Historically, bitmaps have been the dominant representation for images due to their conceptual simplicity and the array-based display (and framebuffer) of modern screen technology. Much effort has been spent in enhancing the compression (e.g., lossy JPEG compression vs. lossless PNG compression) and analysis of raster images. Many image analysis methods depend on the grid-like representation of raster images, such as in the case of efficient parallel computation. Standardized raster images tend to have better support than vector graphics on older devices.

3.2 Vector Graphics

Vector graphics are not represented as pixels, but rather as a collection of parameterized geometric shapes. One of the immediate benefits is an efficient representation for purely geometric images, and the efficient and perfect scaling of continuous geometric objects. In order to display a vector image onto a pixel-based screen, the vector image first has to be rasterized, or rendered. Vector graphics are especially useful for web graphics and other highly geometric designs, such as maps, CAD, and typography. However, vector-based designs tend to be more inefficient for arbitrary image data (with high entropy), due to the relative complexity of shapes to pixels.

3.3 Scalable Vector Graphics File Format

Scalable Vector Graphics, or SVG, is a standard for vector graphics that uses the XML text format. All elements are represented using combinations of seven geometric shapes: Path, Rectangle, Circle, Ellipse, Line, Polyline, and Polygon. Since it is a textual format, it can be easily examined and manipulated by computers or by humans. The SVG standard is stable and supported by many applications, including PDF viewers and web browsers.

4 Related Work

4.1 Tracing Methods for Vectorization

One of the successful earlier methods in image vectorization is called tracing, and at the time was synonymous with image vectorization. The general intuition behind this is that a raster image can be thought of as a collection of adjacent image patches, and we can vectorize an image by detecting edges of shapes.

A noteworthy implementation of image tracing is the Potrace algorithm [5]. As the name suggests, Potrace first attempts to convert a raster image into a series of polygonal paths via edge detection and straight-line detection, and then attempts to simplify (optimize) polygons by reducing path cardinalities and introducing Bezier curves. It employs many useful heuristics to improve image quality, such as removing speckles smaller than a given "turd size," detecting and smoothing corners, redundancy coding in the target format, scaling and rotating a small set of parameterized curves, and data quantization. An illustration of the stages of the Potrace algorithm are shown in Fig. 1. The implementation of Potrace is open-source, and the program is highly configurable via command-line options.

This interpretation of vectorization is useful for simple raster images that are indeed a collection of adjacent shapes, such as map data, floor charts, typography, or charts. For such images, the Potrace algorithm is both reliable and efficient. While we do not use Potrace in our implementation, we may borrow some of its features related to curve optimization when simplifying the shapes (this is future work).

One of the drawbacks of tracing is that we can only trace edges on a binary thresholded image; if there aren't clearly defined edges, or if there are image gradients (as is often the case), it doesn't represent an image as well. Tracing can be applied to color images by thresholding the image by color or brightness

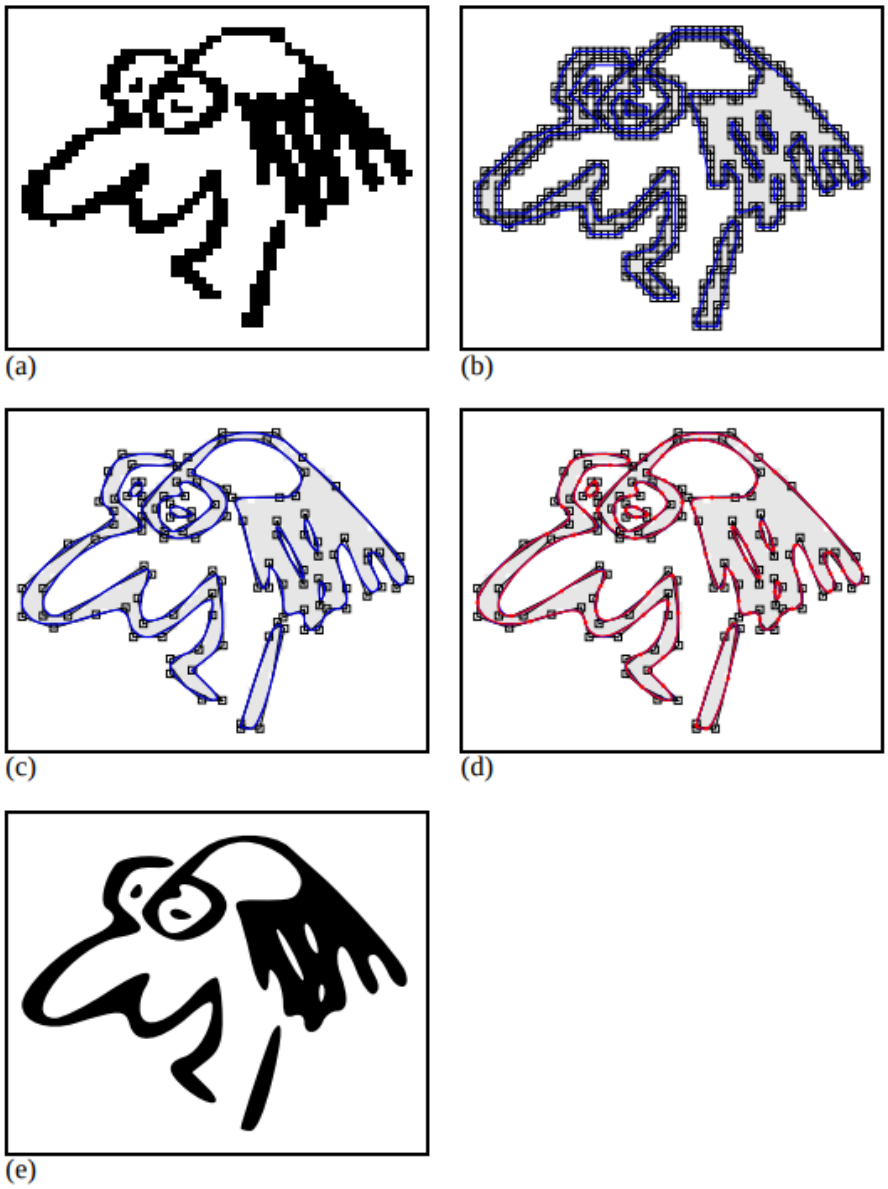


Figure 1: An illustration of the Potrace [5] vectorization process

level, and producing vector images for each thresholded layer, but this may seem choppy and low-quality. Tracing also does not recognize non-contiguous shapes (e.g., simple shapes that intersect other shapes), which can allow for more aggressive optimization and better object recognition. Machine learning approaches are better at recognizing this (citation).

4.2 Machine Learning Approaches to Vectorization

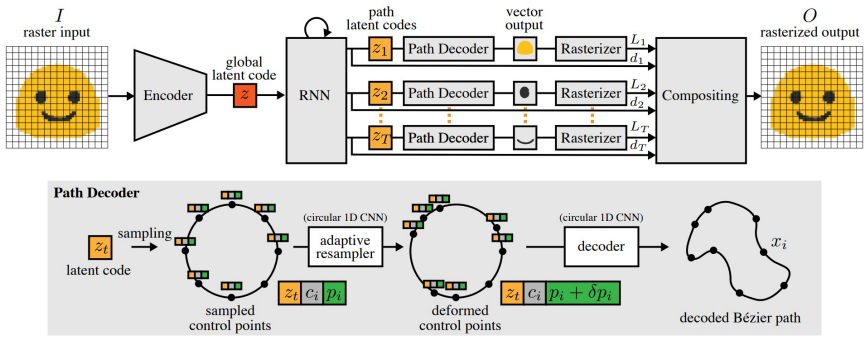


Figure 2: Architecture overview for Im2Vec from [4]

Im2Vec is an end-to-end deep neural network introduced by Reddy et al. [4]. The model takes a raster image as input and outputs an SVG image. The raster image is first passed through an encoder. The encoded representation is then passed through a Recurrent Neural Network (RNN), specifically a bidirectional Long Short-Term Memory network (LSTM), to produce an arbitrary number outputs. Each output is passed through a path decoder, which produces a Bézier path.

The path decoder uses continuous deformation of the unit circle to ensure each path is closed. The authors sample points on the unit circle and concatenate the output of the RNN to each of the sampled points. The authors next use 1D convolutions, equivalent to convolution around the perimeter of the circle,

to determine how to deform the unit circle.

This model is trained by appending a differentiable rasterizer to the output. The rasterizer converts the vector output back to raster format and the resulting image can be compared to the original image to compute a loss. Since the rasterizer is differentiable, that loss can be backpropagated through the model.

We explored using this model as part of our approach, but after looking through the codebase, we discovered that this would likely not work for our purposes. It appears that the authors hard-coded the number of shapes in the input raster image, along with the colors that each shape should be. We are looking to apply our model to various types of architectures, which will likely not be as well-defined as the emojis used in their experiments.

4.3 Sampling Methods for Vectorization

Sampling methods tackle the vectorization problem by stochastically approximating regions of the vector image. This allows us to achieve a reasonable performance and accuracy. Sampling methods may approximate edges less accurately than edge tracing, but they can overcome some of tracing's limitations, namely being limited to binary thresholding. Like tracing methods, it extends fairly well to more complicated images, while machine learning methods currently appear to be more limited to simple images.

An example procedure for image vectorization through sampling is shown in Zhao et al.'s work Zhao, Feng, and Zhou [6]. The sampling method for vectorization is composed of 3 steps. The image is first convoluted with a Sobel differential operator to generate an "importance matrix." This represents the discontinuities or gradients in the original matrix; larger gradients may indicate regions with more detail. We then apply blue-noise sampling to the image using the importance matrix to determine sampling density around each pixel,

with a larger gradient causing a higher sampling density. The sampled points are then triangulated, and the line segments of the triangles form the vector representation of the image.

4.4 Vector Image Optimization

Several methods for optimizing a polygonal path (i.e., a closed polyline) into a smaller polyline, and by expressing curved sequences of edges as a single Bezier curve, are explored in the Potrace algorithm [5]. However, this only deals with simplifying curves, while maintaining the overall topology. The sampling method generates a triangulate mesh rather than a sequence of closed paths; in this method, the optimization scheme may look different and may change the overall topology. For example, we do not have any polylines to curve-optimize unless some edges are removed to form non-triangular polygonal patches; it is then a matter of which edges can be removed while retaining fidelity to the original image.

Hoppe [2] describes an approach to reduce triangular meshes by merging two adjacent vertices in an arbitrary n-dimensional triangular mesh. This technique also results in a triangular mesh, and thus may be useful for 3-D modeling. The method attempts to optimize volume preservation using a quadric error metric and color attribute discontinuities across triangle bounds, illustrated in Fig. 3.

We can be more aggressive than Hoppe's method and not preserve triangular regions, since we are working in a two-dimensional space. We can use their idea of respecting color discontinuities to remove edges, and then perform Potrace's curve optimizations on the resulting polygonal areas.

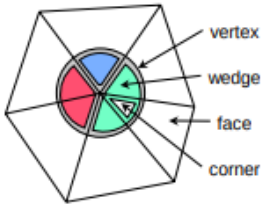


Figure 4: Wedge-based mesh representation.

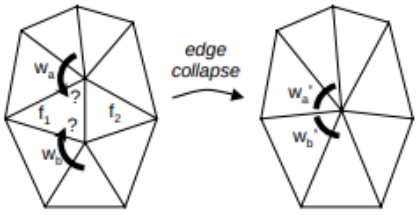


Figure 5: Tests for wedge unification after edge collapse.

Figure 3: Collapsing an edge in [2], taking into consideration color discontinuities

5 Methods

5.1 General

The end goal of the project is to develop an efficient system for image vectorization. The system would take any image as input and output an SVG representation of the original image. The system is mainly composed of 3 different components.

5.2 Sampling

The sampling process is based on [6]. The input image is a regular bitmapped image, such as Fig. 4. The first step is to apply the gradient operator on the image to get the importance matrix, shown in Fig. 5. This involves pixel-wise convolving with four 3×3 Sobel filters (horizontal, vertical, and two diagonals) and finding the magnitude to determine the magnitude of the gradient, which is interpreted as the local information content or "importance" of the area surrounding a given pixel. Next, the importance matrix is thresholded so that only points of high importance are retained, shown in Fig. 6. This is essentially a high-pass filter; low-frequency components of the image are lost. The intuition is that low-frequency elements can be represented with uniformly-colored shapes without much information loss.

Now we perform the sampling to obtain Fig. 7. The sampling density at a pixel is a function of the importance of that pixel; a higher importance leads to a higher sampling frequency. This is known as "blue-noise" sampling, and allows us to focus more detail on regions of higher information content. After sampling, the image is converted to a triangular mesh by performing the Delaunay triangulation, as shown in Fig. 8.

Figure 4: Original image

Figure 5: Importance function applied to the image

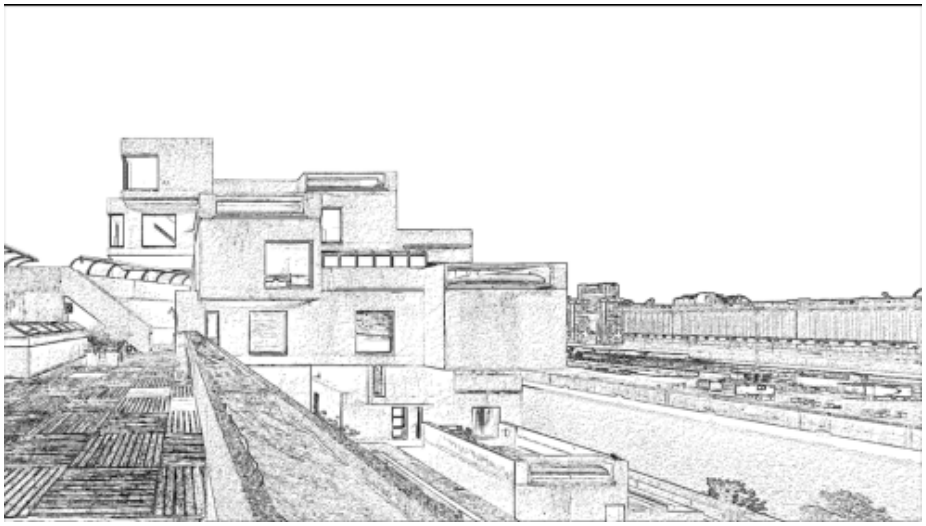


Figure 6: Thresholded points

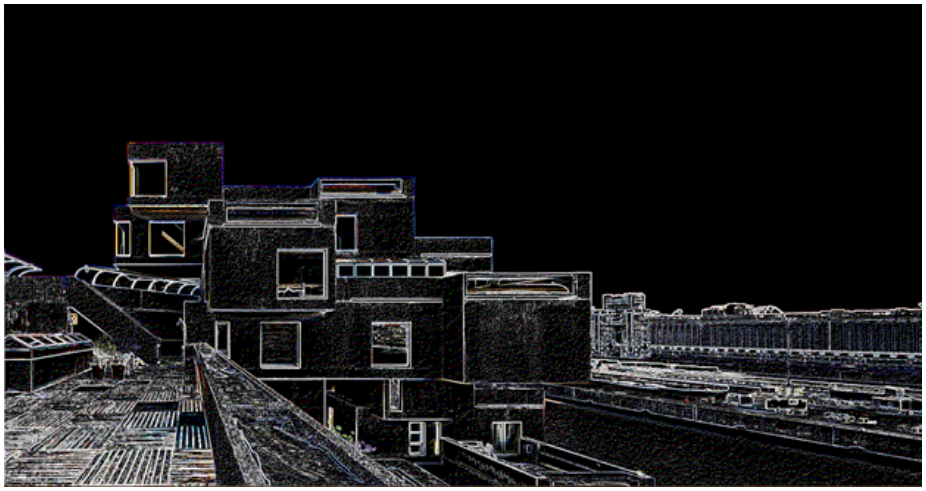


Figure 7: Sampled points

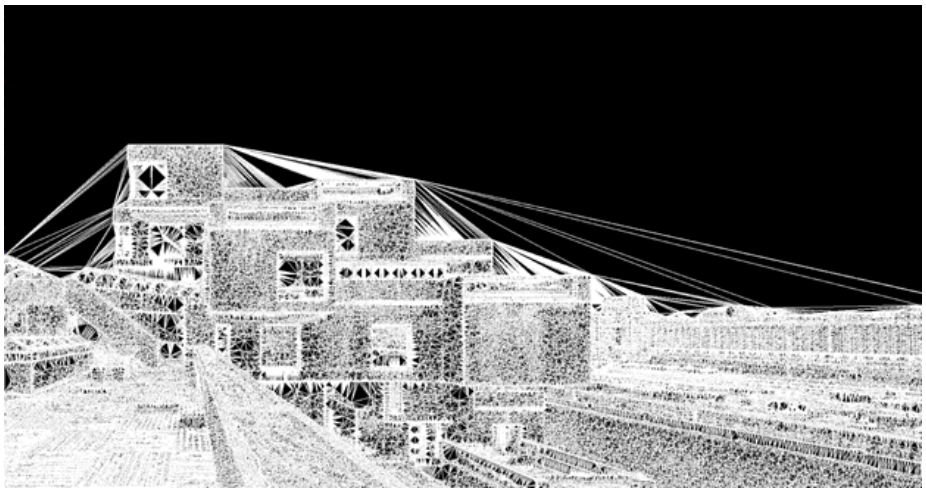


Figure 8: Triangulated image

5.3 Mesh Optimization

This is a future goal for spring semester and has not been implemented yet.

5.4 Writing to SVG

We convert internal representations of triangles (tuples with 3 pairs, representing the coordinates of each point on the triangle) to an SVG file. We use Python's pycairo library to accomplish this.

In the future, we may also attempt to optimize this, such as by coding redundant information.

5.5 Evaluation Metric

We implemented an evaluation metric based on content loss from Dumoulin, Shlens, and Kudlur [1]. Similar to the methodology used by the authors, We use a pre-trained VGG-19 and strip off the $conv_5$ and fully-connected blocks. It is important to note that the authors used a VGG-16. We use a VGG-19,

similar to Huang and Belongie [3]. In order to get the content loss between two images, we take the Euclidean distance between the outputs of the stripped VGG-19.

The reason that we strip off the $conv_5$ and fully-connected blocks is to obtain the high-level feature maps in the classifier. We consider the high-level feature maps to be the intermediate representation of the content of the image. This stems from a hypothesis for deep convolutional neural networks: the early layers in a classifier are used to extract low-level features, such as edges, while the later layers of the classifier combine the low-level features to create high-level features, such as an arm or a leg.

We tested the content loss metric on three images related to style transfer using Generative Adversarial Networks. The images were from `https://www.tensorflow.org/tutorials/generative/style_transfer`.

Figure 9: Yellow Labrador Looking, from Wikimedia Commons

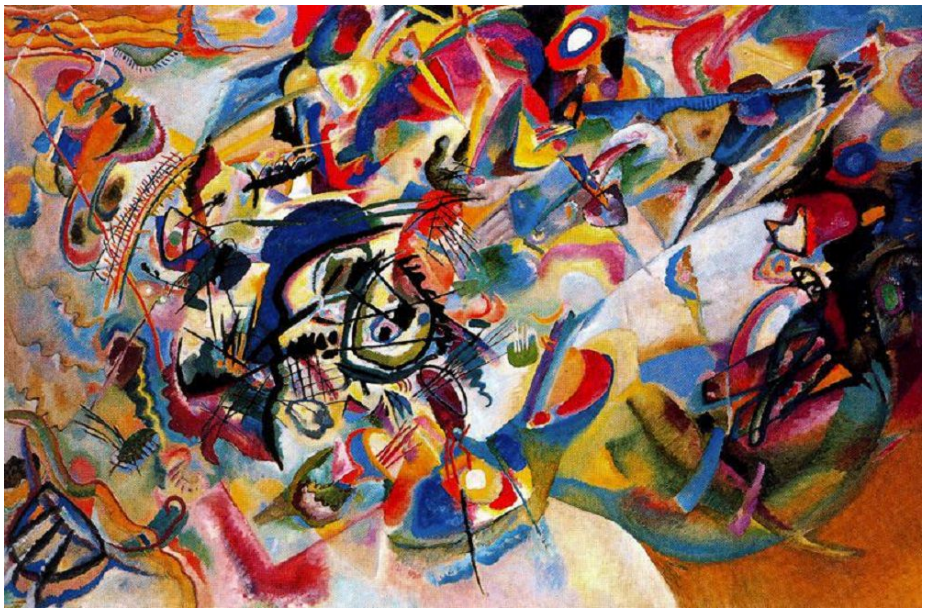


Figure 10: Composition VII by Wassily Kandinsky

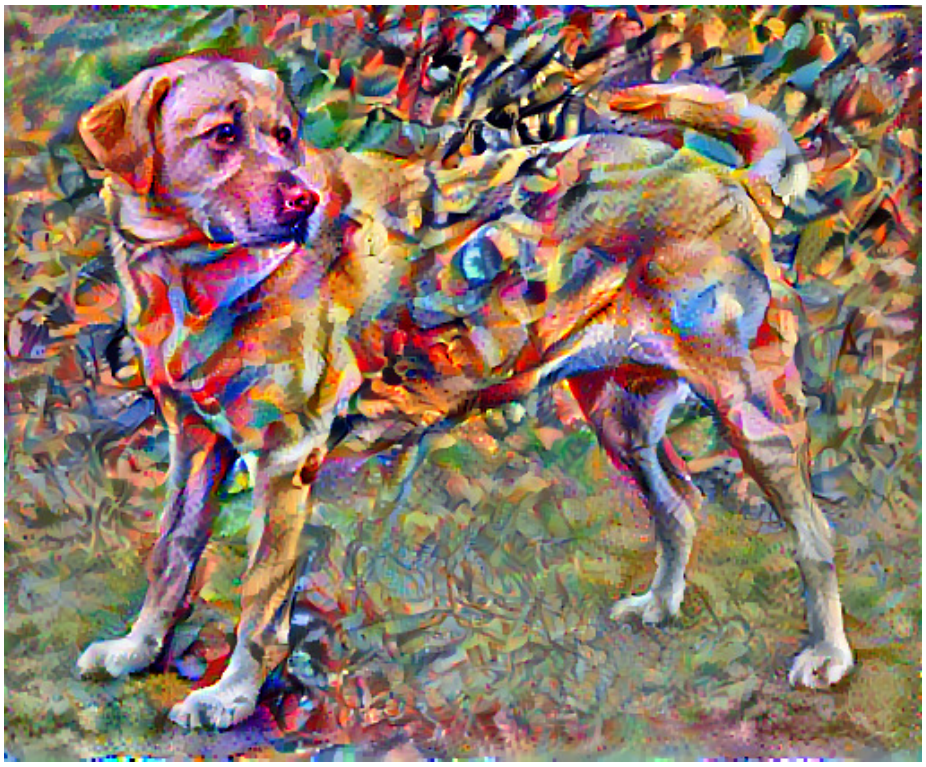


Figure 11: Generated image produced with style transfer

Fig. 11 is generated using the content from Fig. 9 and the style from Fig. 10. We calculated the content loss between all three pairings of the above images. The results are displayed in the table below.

Image 1	Image 2	Content Loss
Fig. 9	Fig. 11	132352.05
Fig. 9	Fig. 10	190958.28
Fig. 10	Fig. 11	190266.86

Table 1: Sample content loss for example images

As can be seen in Table 1, the lowest content loss is between Fig. 9 and Fig. 11, which is expected. In addition, the content loss between Fig. 10 and the other two images is almost the same, which makes sense, because the other

two images should have the same content with different style.

We may want to refine this evaluation metric in the future. The content loss values are extremely large and are difficult to interpret on their own. We may wish to use a baseline image when comparing the results of our project. We would thus compare three images (the original image, the vectorized image, and the baseline image). This baseline image can be arbitrarily selected. The purpose of the baseline image is to provide context to the content loss values. Similar to the above example, we would compare all three pairings of the images. We would want the content loss between the original image and the vectorized image to be the lowest. We would also want the content loss for the other two pairings (with the baseline image) to be relatively similar.

5.6 Implementation stack

The current implementation is a mix of Python and C++. The image processing components utilize OpenCV for GPU support and will be multi-threaded. Efficiency of the process is not a concern at this point; we care more about optimizing for the error metric. Future iterations of this project may consider efficiency of implementation.

6 Conclusion

Image vectorization is the process of converting a raster image to a vector image, and may lead to efficiency gains for highly vectorized images. While vectorization has been used successfully on simple vector-based input images in the past such as map or typography images, we aim to improve its output on highly-geometric but less-exact images, such as architectural images. We believe that this may be useful in the architectural design process, and perhaps in other design processes whose subject is highly geometric.

We have implemented a basic framework for vectorizing raster images, primarily based on [6]. We can take an input image, perform sampling on the image, triangulate the sampled points, and produce an output image. So far we have yet to optimize this method towards architecture – this will involve optimization stages after producing the mesh.

7 Future Work

In the upcoming semester, we would like to experiment with different vectorization approaches. This may involve different sampling methods or different optimizations. We may choose a different method for generating vectors from the sampled points. Currently, we are using a standard Delaunay triangulation, although there may be alternative methods available.

We may change our evaluation metric if we discover a better metric. We will rank each of our approaches according to our evaluation metric. By the end of the upcoming semester, we will also produce a presentation describing our results, along with a final report that contains the final decisions made for our project.

An approximate timeline for the spring semester may look like:

January Begin work on mesh optimization, review prior work.

February Continue mesh optimization, clearly define evaluation metrics.

March Begin gathering results and evaluating using given metric.

April Finish result-gathering, begin final report and presentation.

May Complete final report and presentation, present results.

8 References

[1] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. *A Learned Representation For Artistic Style*. 2017. arXiv: 1610.07629 [cs.CV].

[2] Hugues Hoppe. "New quadric metric for simplifying meshes with appearance attributes". In: *Proceedings Visualization'99 (Cat. No. 99CB37067)*. IEEE. 1999, pp. 59–510.

[3] Xun Huang and Serge Belongie. *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization*. 2017. arXiv: 1703.06868 [cs.CV].

[4] Pradyumna Reddy et al. *Im2Vec: Synthesizing Vector Graphics without Vector Supervision*. 2021. arXiv: 2102.02798 [cs.CV].

[5] Peter Selinger. "Potrace: a polygon-based tracing algorithm". In: *Potrace (online), http://potrace. sourceforge. net/potrace. pdf (2009-07-01)* 2 (2003).

[6] Jiaojiao Zhao, Jie Feng, and Bingfeng Zhou. "Image vectorization using blue-noise sampling". In: *Imaging and Printing in a Web 2.0 World IV*. Vol. 8664. International Society for Optics and Photonics. 2013, 86640H.

Current event: FBI withheld keys to REvil Kaseya ransomware attack

Jonathan Lam

09/22/21

Two months ago, while working at an internship at my makeshift work-from-home office, my mom (also WFH in the next room over) sighed in frustration. When asked, she said that her work systems were down due to a ransomware attack. Since it was work-from-home, all the remote work services were shut down and operations at her office stopped for a few days. Luckily, the company was able to restore their systems from a backup, and all of the employees had to bring in their personal devices to be scanned for malware.

It turns out that this was part of a large-scale ransomware attack on a IT firm called Kaseya, which affected a number of their small- to medium-sized clients [3]. The attack was carried out by a Russian "Ransomware-as-a-Service" group called REvil, which also carried out the ransomware attack on JBS, the world's largest meat supplier, in June. Kaseya creates VSA software, "a unified remote-monitoring and management tool for handling networks and endpoints." This software is high-value for an attacker because all the devices that are managed by this software trust the commands sent out by the device (the Kaseya installation folders must be excluded from antimalware/firewalls for the program to work). As a result, after performing a zero-day attack on the server, malicious commands that were sent to managed devices were not checked by malware and were able to perform the attack. Sophos published a detailed description of the attack [1].

This was a fairly high-level attack (Sophos notes that zero-day supply-chain attacks are rare and sophisticated) so it is not the fault of poor security practices (such as bad password practices). Sophos notes that the antimalware evasion that was performed by the attack (which targets Microsoft Defender) would have been detected by its own antimalware software, as would the certificate that was injected as part of the attack. They also note that this is the

In a high-level technical attack like this, most of the methods to mitigate the fallout is due to roundabout measures. The saving grace for a number of companies (such as my mom's company) was the use of backups – this cannot protect against zero-day exploits or prevent ransomware attacks that exfiltrate data (which this attack does not), but it allows a company to restore operations. Shutting down servers as soon as an attack is discovered can mitigate ransomware spread. Installing more advanced antimalware software, such as Sophos's antimalware or Intercept X's cryptoransomware protection software, would have detected these attacks (as Sophos notes).

Politics is clearly at play here as well. REvil is not the first RaaS group, and they operate with apparent impunity in Russia. They have attacked multiple international corporations, including several large scale attacks involving U.S. corporations. The Washington Post article notes that "The White House has made fighting ransomware a priority, and President Biden has urged Russian President Vladimir Putin to rein in ransomware criminals operating out of Russia." Threat of punitive action by the government would likely be a deterring factor.

A highly related issue of ethics arose three weeks (19 days) later when the FBI released a global decryption key for Kaseya [2]. It turns out that they had obtained from the REvil servers much earlier, but had hoped to "was planning to carry out an operation to disrupt the hackers, a group known as REvil, and the bureau did not want to tip them off" [2]. However, they did not get a chance to because REvil went offline in mid-July, after which they revealed their knowledge of the key to Kaseya. The ethical issue is the tradeoff between helping victim businesses (by releasing the key immediately) or potentially dealing much damage to the attackers (by holding the key and performing a counterattack). The FBI had to constantly weigh the damage dealt by both options.

References

[1] Mark et al. Loman. *Independence Day: REvil uses supply chain exploit to attack hundreds of businesses.* July 2021. URL: `https://news.sophos.com/en-us/2021/07/04/independence-day-revil-uses-supply-chain-exploit-to-attack-hundreds-of-businesses/` (visited on 09/22/2021).

[2] Ellen Nakashima and Rachel Lerman. *FBI held back ransomware decryption key from businesses to run operation targeting hackers.* Sept. 2021. URL: `https://www.washingtonpost.com/national-security/`

ransomware-fbi-revil-decryption-key/2021/09/21/4a9417d0-f15f-11eb-a452-4da5fe48582d_story.html (visited on 09/22/2021).

[3] Charlie Osborne. *Updated Kaseya ransomware attack FAQ: What we know now.* July 2021. URL: https://www.zdnet.com/article/updated-kaseya-ransomware-attack-faq-what-we-know-now/ (visited on 09/22/2021).

Buffer Overflow Analyzer using LLVM

Jonathan Lam and Daniel Tsarev

Motivation

Lab 1
BUFFER OVERFLOW BAD

Threat Model

Buffer overflow compromise the whole system,

S ✔
T ✔
R ✔
I ✔
D ✔
E ✔

Risk Analysis

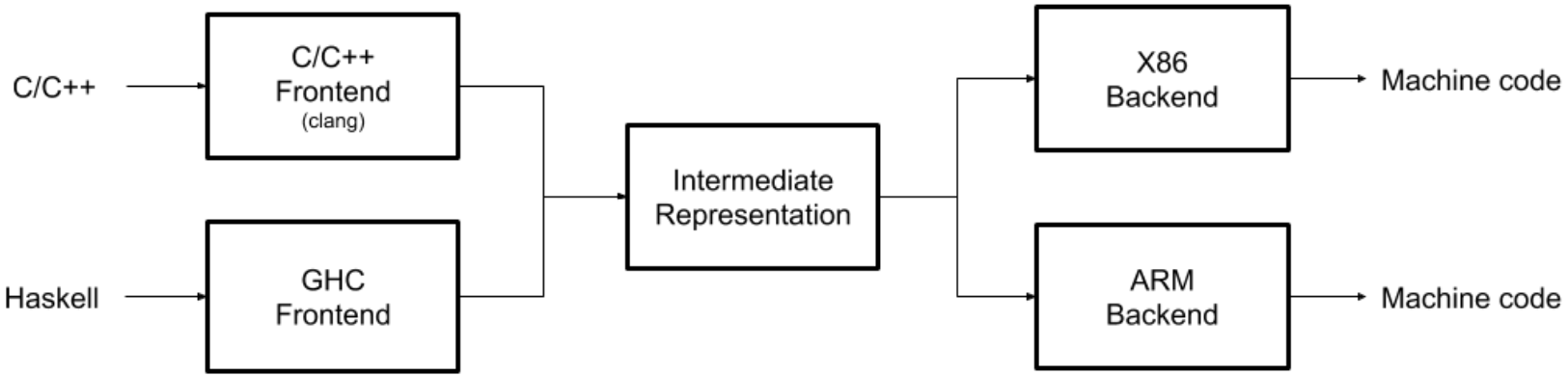
Downtime costs **$$$**

- Gartner, estimates a $5,600 cost per minute of downtime of an IT system

Data leaks can be crippling for a business

- LOTS OF **$$$$**

What is LLVM?


C/C++
C/C++
Frontend
(clang)
Haskell
GHC
Frontend
Intermediate
Representation
X86
Backend
ARM
Backend
Machine code
Machine code

Intermediate Representation (IR)

- Instructions
- Basic Blocks
- Control Flow Graphs

Code

```
W := 0;
X  := W + 1;
Y  := 2;
If (X > Z) {
    Y = X;
    X++;
} else {
    Y := Z;
}
W := X + Z;
```

B1

```
W = 0;
X = W + 1;
Y = 2;
If (X> Z)
```

B2

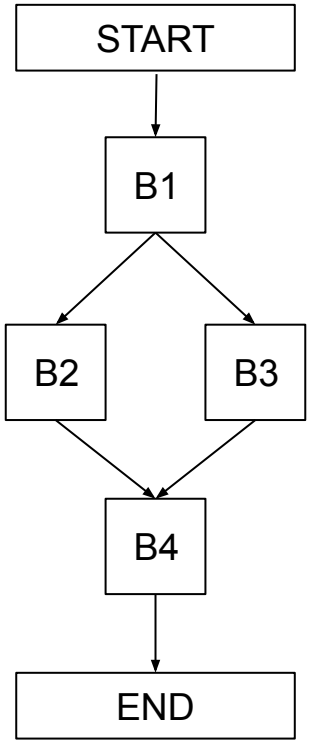
```
Y = X;
X++;
```

B3

```
Y = Z;
```

B4

```
W = X + Z;
```



Intro. to Data-flow Analysis (DFA)

Better with an example

DFA Example: Zero Analysis

X := 0
Y := 1
Z := Y
Y := Z + X
X := Y - Z

X / Y
Y / X

DFA EXAMPLE: ZERO ANALYSIS

```
X := 0
Y := 1
Z := Y
Y := Z + X
X := Y - Z

X / Y
Y / X
```

1. Indeterminate values

DFA EXAMPLE: ZERO ANALYSIS

```
X := 0
Y := 1
Z := Y
Y := Z + X
X := Y - Z

X / Y
Y / X
```

1. Indeterminate values

2. Loss of precision -- conservative

DFA EXAMPLE: ZERO ANALYSIS

```
X := 0
Y := 1
Z := Y
Y := Z + X
X := Y - Z

X / Y
Y / X
```

1. Indeterminate values

2. Loss of precision -- conservative

3. Define invariant and transition fn

INTRO TO DATAFLOW ANALYSIS

Trace some **invariant/property/thing to analyze** throughout the program

We track the analysis over **variables**; the analysis is expressed as a **abstract value**

Abstract values form a **lattice** (partial order); dataflow analysis generalizes all possible abstract values

Decide how each instruction affects the property -- define a **transition function**

Other Data-flow Analyses

Zero analysis: detect divide by zero

Constant propagation: optimization, convenient compile-time computations

Live variables: optimization of registers

Reaching definitions: detect uninitialized variables

Buffer origin (our analysis): detect some cases of buffer overflows

BUFFER ORIGIN ANALYSIS

Invariant: set of static buffer variables that a buffer can refer to

Transition function: pointer assignment, load/store operations

BUFFER ORIGIN DATAFLOW ANALYSIS (BODA)

Invariant: set of static buffer variables that a buffer can refer to

Transition function: pointer assignment, load/store operations

Assumptions:

- Only analyzing attention to stack-allocated fixed-size buffers
- No pointer arithmetic
- Function calls do not affect dataflow analysis (transition fn is identity map)
    - Probably an OK assumption

BUFFER ORIGIN DATAFLOW ANALYSIS (BODA)

```
char *p, d[512], s[512];
p = d;
strcpy(p, s);
```

WORKLIST ALGORITHM

```
function BodaWorklist(f)
    B ← basic blocks in f
    mark B_0 dirty
    B_d ← B[0]
    while B_d is not empty do
        b ← pop from B_d
        if b is dirty then
            analyze(b)
            mark b clean
            if b not at fixpoint then
                for all b_s ∈ B : b_s succeeds b do
                    mark b_s dirty
                    insert b_s into B_d
                end for
            end if
        end if
    end while
end function
```

INTRO TO INTERPROCEDURAL ANALYSIS

```
int f(char *d, char *s) {
        strcpy(d, s);
}

int main() {
        char d1[512], s1[512], d2[1024], s2[1024];
        f(d1, s1);
        f(d2, s2);
}
```

INTRO TO INTERPROCEDURAL ANALYSIS

Loss of precision is common in dataflow analysis -- analysis captures global information and doesn't consider **context** due to only traveling down one branch

Analyzing branches separately is **expensive** but we can provide better context-sensitivity

Dataflow analysis within functions (many instructions, efficiency) but interprocedural call-graph tracing (few fncalls, context-sensitive)

TRACING THE CALL GRAPH

```
function TraceCallGraphRec(f, O)
    for all function invocations i_g() ∈ f do
        p_g ← [O_{i_g}/O]p_g
        if g is dangerous then
            warn about dangerous usage
        else if g is user-defined then
            TraceCallGraphRec(g, p_g)
        end if
    end for
end function

function TraceCallGraph(M)
    f_m ← main function of M
    TraceCallGraphRec(f_m, {})
end function
```

Handling (mutual) recursion: convergence along call stack

DEMO

FUTURE WORK

- Examine more complicated sample programs
- Improve presentation of analysis
- Finish interprocedural analysis implementation
- Handle missing assumptions
- Other dataflow analyses to augment BODA (e.g., constant propagation)
- Study implementation of existing static analyzers

QUESTIONS?

GitHub

Program Analysis textbook

ECE455 – Final Project Proposal

Jonathan Lam and Daniel Tsarev

2021/11/04

1 Interests

We both have found an interest in programming languages and verification. In particular, both of us took the Compilers course last semester with Prof. Hakner, and we both have been looking at functional programming languages like Haskell (Dan is interested in its use in cryptocurrency, while Jon is interested in its use in formal analysis). Jon is also currently taking an I.S. in formal analysis. We are also both interested in LLVM.

2 Proposal

We would like to develop a (very) simple static analyzer for (blatant) buffer overflow issues in a language like C. This would involve a dataflow analysis and an interprocedural analysis to track the common causes of buffer overflow (e.g., unchecked-length buffer functions such as `strcat`) and `printf` attacks. The analysis would also reveal information about possible attack vectors, and would be conservative in the analysis of unknown (i.e., library) functions.

For the implementation, we may use a tool such as clang or WLLVM to parse a C file to LLVM, to avoid manual parsing. The analysis may be conducted in a functional language such as OCaml or Haskell, which are often used in program analysis scenarios. (This would also allow both of us to learn about LLVM and apply functional programming.)

3 Justification

One may wonder if this exercise is useful, since static analyzers can do this already and to a better degree, and modern OSes already have protection against these common attacks. Despite this, it would be a tremendous learning experience that ties together our experiences from this course, formal analysis, and compilers.

4 Project deliverables/Evaluation

The end result is a tool that, when run on a C program, will be able to detect and report simple buffer overflow vulnerabilities. Additionally, this tool should be able to have report some additional information about the vulnerability, such as the possible sizes of the buffer(s), the vulnerable function(s), the calling context, etc.

The end result will be limited in many ways – it may not be able to detect a vulnerability such as the url_decode function, which ranges over the input buffer without considering the length of the destination buffer. Essentially, what we hope to do is look for usages of known vulnerable functions and trace information that may be useful for protecting against or exploiting buffer overflow vulnerabilities. If there is time, the same may be adapted for detecting simple printf vulnerabilities.

To evaluate this project, we will test the program on the Lab 1 program, and perhaps other C programs with known vulnerabilities (to check for detecting of known vulnerabilities) or programs with unknown vulnerabilities (to detect unknown vulnerabilities).

5 Timeline

The following dates are approximate proposed deadlines for major steps of the project. The first three weeks will also be onboarding time for learning LLVM, OCaml, and program analysis techniques.

- Nov 4 – Finish final project proposal.
- Nov 11 – Implement clang/WLLVM parser, understand basics of LLVM and create debugging framework (design helper functions to print out AST/CFG)
- Nov 18 – Design analysis framework (e.g., which abstract variables to track)
- Nov 25 – Turkey Day, begin implementation of analysis framework (implementing the lattice, tracking abstract variables)
- Dec 2 – Continue implementation ....
- Dec 9 – Finish implementation, work on presentation
- Dec 16 – Finalize presentation

6 References/Materials

6.1 LLVM

We will use a LLVM IR (generated by clang or a similar tool) as the representation of a program. The language of choice for our implementation is OCaml.

- LLVM tutorial: Kaleidoscope. https://llvm.org/docs/tutorial/MyFirstLanguageFrontend

6.2 OCaml

We are both exploring functional programming in OCaml.

- Real World OCaml. https://dev.realworldocaml.org/
- OCaml Llvm module tutorial. https://www.wzdftpd.net/blog/ocaml-llvm-01.html
- OCaml Llvm module documentation. https://llvm.moe/ocaml/Llvm.html

6.3 Program Analysis

The first few chapters of this book provide a method (data-flow analysis) for performing an analysis on code using a lattice data structure. In our case we will customize the dataflow analysis to track buffers and vulnerable function calls.

- Program Analysis. https://cmu-program-analysis.github.io/2021/resources/program-analysis.pdf

6.4 Static analysis for buffer overflows

The following papers are used to get an idea for previous buffer-overflow detection using static analyzers, but they are probably too complex to follow in-depth.

- Wikman, Eric C. Static Analysis Tools for Detecting Stack-Based Buffer Overflows. NAVAL POSTGRADUATE SCHOOL MONTEREY CA MONTEREY United States, 2020.
- Shahriar, Hossain, and Mohammad Zulkernine. "Classification of static analysis-based buffer overflow detectors." 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement Companion. IEEE, 2010.
- Zitser, Misha, Richard Lippmann, and Tim Leek. "Testing static analysis tools using exploitable buffer overflows from open source code." Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering. 2004.

Buffer overflow analyzer

Jonathan Lam & Daniel Tsarev

December 18, 2021

Contents

1 Abstract

Buffer overflows are a very well-known form of software vulnerability in which memory buffer overruns allow attacker to read and/or write memory locations, in some cases allowing them to inject crash or take control of a system. They are prevalent in low-level software written in memory-unsafe languages, such as and especially web servers.

We aim to create a simple static analysis tool that tracks the use of stack-allocated buffers to known memory-unsafe functions in C source code, such as the notorious `strcpy` standard library function. This is achieved using a dataflow analysis and call graph tracing on the unoptimized LLVM IR generated using the clang compiler frontend. We develop a novel dataflow analysis method, called "buffer origin dataflow analysis," that allows us to track assignments of buffers. We evaluate this method and describe its shortcomings. A sample implementation can be found on GitHub at @jlam55555/overflow-analyzer.

2 Motivation

The motivation for this project was the first lab[1], where the source code of a toy web server was analyzed and buffer overflows were used to gain unauthorized access. After doing the lab by hand, we wondered if similar vulnerabilities could be detected automatically via static analysis, and were motivated to create a tool that does so.

2.1 Threat model

There are a few closely-related buffer-overflow attacks. Most involve the overflowing of fixed-size local buffers, which are stored on the stack. The stack also contains important information about program control flow that should not be accessible by the programmer or a user of the program. Some low-level languages like C are memory-unsafe and do not check bounds or pointer accesses, and thus overflowing buffers may overwrite these critical locations in memory. A malicious attacker can craft special messages including their own arbitrary code ("shellcode") that are intended to be placed in these buffers, such that they hijack the control flow of the program. This is especially problematic if the application is running as a privileged user.

A seminal paper in the literature of buffer overflows is the paper "Smashing the stack for fun and profit" by Aleph One. Over time, there have been numerous attempts at curbing stack overflows, such as stack canaries, address space layout randomization (ASLR), no-execute (NX) stack memory pages, memory-safe languages, better error warnings about unsafe usages. However, as attackers get more creative (e.g., "return-to-libc" attacks to bypass the NX-bit or heap-spraying to bypass ASLR), there is no complete solution for a C/C++ or assembly programmer working at the low-level OS or firmware level.

If we consider the STRIDE security model, almost all facets are compromised due to the low-level nature of the code. The main concerns are:

Tampering The integrity of the compromised program is no longer secure, along with any data that the program interacts with.

Denial of service Buffer overflows may lead to a program crash.

Elevation of privilege As low-level services tend to be running with special users with special privileges, an attacker with control of the program now has access to these privileges.

Since buffer overflows typically attack a low-level system or service, the entire service becomes compromised. In the case of complex services with many user interactions, such as a webserver, this may compromise the security with spoofing, information disclosure, and repudiability at the application level.

[1] MIT 6.858 Lab 1: Buffer overflows

2.2 Risk analysis

While buffer overflow attacks are one of the oldest and best known attacks, the OWASP Foundation classifies the severity as "very high" and the likelihood of exploit as "high to very high," and states that "buffer overflow attacks against both legacy and newly-deveilop applications are still quite common"[2].

A quick search for buffer overflow vulnerabilities on the Common Vulnerabilities and Exposures (CVE) lists 12323 exploits, 765 vulnerabilities in 2021 alone[3]. Buffer overflow vulnerabilities may cause data leakage or downtime, which are very expensive. Gartner, a technology consulting company, estimates that the typical company loses $5,600 per minute of downtime[4]; IBM estimates that the average data leak for a U.S company costs $4.24 million in 2021[5]. Needless to say, buffer overflow attacks are very prevalent, and the damage is costly.

[2] `https://owasp.org/www-community/vulnerabilities/Buffer_Overflow`

[3] `https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=buffer+overflow`, counted by number of search results at time of writing

[4] `https://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/`

[5] `https://www.ibm.com/security/data-breach`

3 Methods

Buffer overflow analysis is performed through a dataflow analysis on a LLVM IR representation of the source code input. Explanations of IRs, LLVM, and dataflow analysis will be explained in the following sections.

3.1 Review of LLVM and intermediate representations

LLVM is a set of modular libraries used to aid in compiler implementation. One of it core features is a low-level, consistent, human-readable and easily programmable **intermediate representation** (IR). This IR can be thought of as a modern, high-level, machine-independent assembly language. A number of compiler front-ends (i.e., the component of a compiler that manages parsing a program to an abstract syntax tree (AST)) target LLVM, including the C compiler clang. Our implementation uses clang to generate LLVM code from a C source file.

Some general compilers terminology may be helpful when referring to the LLVM IR:

Instruction An atomic statement that models assembly instructions.

Virtual register (VR) An abstract register that is used for storing temporary values from instructions; can be thought of as a variable in the IR.

Basic block A sequence of instructions that do not involve branching (control flow).

Control flow graph (CFG) A sequence of basic blocks that models the flow of control of a program.

Instruction set architecture (ISA) A set of instructions that an abstract model of a CPU can execute.

Three-address IR/ISA A form of IR where each instruction has an operation code (opcode), up to two parameter virtual registers, and an optional destination virtual register.

Load/store IR/ISA An IR/ISA that tends to adhere to the simpler-instruction model of RISC ISAs. In particular, memory operations (such as `load`/`store`) are explicit and separated from computation.

LLVM is a three-instruction, load-store IR. A program is contained in a `llvm::Module`, which stores a collection of global variables and `llvm::Functions`; each `llvm::Function` is a collection of `llvm::BasicBlocks`; and each `llvm::BasicBlock` is a collection of `llvm::Instructions`.

LLVM also includes the concept of a VR, called `llvm::Value`, which has a type (`llvm::Type`). There is a dual between VRs and instructions; each instruction in a three-address ISA has a target VR (if it has a target; otherwise the target value is of type `void`). As a result, each instruction is identified by

its target value, and thus every instruction *is a* value (i.e., `llvm::Instruction` subtypes `llvm::Value`). It may seem strange in the implementation to be using instructions as values, but this is due to this equivalence between instructions and VRs.

3.1.1 Difficulties in OCaml

Our original goal was to implement our analysis program in OCaml, since it is a program often used in programming languages (PL) theory and has a port of LLVM. However, we found that the OCaml LLVM package is outdated, so we switched to the native implementation language, C++.

3.2 Review of dataflow analysis

Dataflow analysis is a principled approach to statically tracking some property throughout the source code of a program, and forms the basis of many static analysis approaches. The type of property, i.e. the thing that we want to analyze, is highly dependent on the type of dataflow analysis.

We may consider **zero analysis** as an example for minimum understanding. In zero analysis, we track whether a variable is zero at any point in the program. We list the instructions in a basic block vertically, and list the variables we want to analyze horizontally, so that we have a table, as shown in Table 1. We start with an initial analysis of all unreachable ($\bot$) abstract values. After each instruction, we update the analysis based on the instruction and the previous analysis, so that the invariant (property) is preserved – the rule that determines the next analysis as a function of the previous analysis and the current instruction is called the transition function.

In the case of zero analysis, we have four abstract values that may determine the possible analyses of a variable:

$$L = \{\bot, Z, N, \top\}$$

where Z means definitely zero, N means definitely non-zero, $\top$ is a special value that means indeterminate (Z or N). $\bot$ is a special value that means that this value has not been analyzed yet, or unreachable, and is useful for simplifying the analysis.

Zero analysis is defined by the following abstraction function α which defines the invariant that we are tracking. In this case, if a variable evaluates to zero, then it is given the abstract value Z; if it evaluates to some non-zero value, then it is given the abstract value N. (Note that there are also the additional values $\bot$ and $\top$ when neither of these analysis hold true.)

$$\alpha(v) = \begin{cases} Z & \text{if } v \Downarrow 0 \\ N & \text{if } v \Downarrow n \wedge n \neq 0 \end{cases}$$

The transition function describes the analysis transition after an instruction. It is defined piecewise on the instruction type. Assume x, y are variables to

Instruction	x	y	z
	$\bot$	$\bot$	$\bot$
$x \leftarrow 0$	Z	$\bot$	$\bot$
$y \leftarrow 1$	Z	N	$\bot$
$z \leftarrow y$	Z	N	N
$x \leftarrow z + x$	Z	N	N
$x \leftarrow y - z$	$\top$	N	N

Table 1: Sample zero analysis

analyze, and c is a constant, and our language is composed only of assignments, additions, and subtractions.

$$f(x \leftarrow c, \sigma) = \begin{cases} \sigma[x \mapsto Z], & \text{if } c = 0 \\ \sigma[x \mapsto N], & \text{else} \end{cases}$$

$$f(x \leftarrow y, \sigma) = \sigma[x \mapsto \sigma_y]$$

$$f(z \leftarrow x \pm c, \sigma) = \begin{cases} \sigma[z \mapsto \sigma_x], & \text{if } c = 0 \\ \sigma[z \mapsto N], & \text{if } \sigma_x = Z \wedge c \neq 0 \\ \sigma[z \mapsto \top], & \text{else} \end{cases}$$

$$f(z \leftarrow x \pm y, \sigma) = \begin{cases} \sigma[z \mapsto \sigma_x], & \text{if } \sigma_y = Z \\ \sigma[z \mapsto \sigma_y], & \text{if } \sigma_x = Z \\ \sigma[z \mapsto \top], & \text{else} \end{cases}$$

Hopefully this simple example gives enough intuition into dataflow analysis. More detailed explanations of some of the terms are given below.

Program point A point in the program's runtime, i.e., a conceptual state if the program were paused between two sequential instructions. A dataflow analysis is defined for each program point, and is identified by the instruction I that precedes it.

Tracked variable This represents a physical element of the program to analyze. For example, we can perform the analysis on each variable in the source code. Tracked variables (or simply "variables") are denoted v.

Abstract value A member of a lattice that represents some information about the dataflow analysis. This represents some property of the tracked variables that we want to track for this analysis. An abstract value is denoted l, and the set (a lattice) that it is defined on is denoted L.

Abstraction function A function $\alpha(v) = \sigma_v$ mapping an tracked variable to an abstract value. This is the invariant that must be maintained by the transition function.

Analysis information (Referring to an instruction, basic block, or function): All of the tracked variables and their corresponding abstract values in the given instruction/basic block/function. Sometimes abbreviated to simply "analysis." Denoted using σ. For the abstract value associated with a single variable v, we can denote this as σ_v.

Lattice An algebraic structure used to represent a partial ordering on a potentially infinite set. In dataflow analysis, we define the subsumption partial order $\sqsubseteq$ on the set of abstract values L. In this context, we pronounce the expression $L_1 \sqsubseteq L_2$ as "L_2 is more general than L_1". We can informally extend this to an analysis on all tracked variables of an analysis, i.e., $\sigma_1 \sqsubseteq \sigma_2 \iff \forall v \in \sigma_1 \cap \sigma_2 . \sigma_{1_v} \sqsubseteq \sigma_{2_v}$. A valid transition function must produce a more general analysis than its input, i.e., $\sigma \sqsubseteq f(I, \sigma)$. For the purposes of our analysis, every lattice is a meet- and join-semilattice, which include the upper-bound $\top$ (pronounced "top") and the lower-bound $\bot$ (pronounced "bottom").

Transition function A function $f(I, \sigma_v)$ taking a program point and an analysis, and returning the updated analysis after the instruction such that $\alpha(v) = f(I, \sigma_v)$; in other words, we can never "narrow" the analysis, but only "widen" it with additional information that generalizes over all known states, especially in the case of combining multiple incoming edges (joining analyses). As before, this can be extended to the entire analysis σ rather than a single variable.

Join operation A binary function $l_1 \wedge l_2$ on a lattice that produces a minimum upper-bound to l_1 and l_2 (unique given a join-semilattice). As before, this can be naturally extended to an analysis $\sigma_1 \wedge \sigma_2$. The join operation is used to "join" the outgoing dataflow analyses of all incoming edges to a basic block, which then forms the initial dataflow analysis for that basic block.

The dataflow analysis is performed by invoking the transition function at each program point and joining incoming edges of basic blocks until the program reaches a fixpoint (i.e., until no operations change the analysis of the output program point). This is performed on a per-function basis using a simple worklist algorithm, described later; the per-function analyses are then used by the interprocedural analysis, which is described in the next section.

We defer to Chapter 4 of the *Program Analysis* textbook (linked under References) for an introduction to dataflow analysis.

3.3 Review of interprocedural analysis

We can extend dataflow analysis to multiple functions in multiple ways. One way is to simply inline all functions and perform a regular dataflow analysis; however, this method is costly and impossible for (mutually) recursive functions. This forces us to not inline function calls, but treat the analysis as a series

```c
void f(char *d, char *s) {
    strcpy(d, s);
}

int main() {
    char d1[512], s1[512], d2[1024], d1[1024];
    f(d1, s1);
    f(d2, s2);
}
```

Figure 1: Issue with specificity of global dataflow analysis

of links, as a call graph. (Note that this is more complicated than a CFG, since no branches occur in the middle of a basic block; a function may invoke another function at any point within the caller, so we need to keep the analysis information at every program point of the caller).

There is some difficulty if we perform the interprocedural analysis in the same manner as the dataflow analysis, i.e., if we converge to a fixed point globally (over all functions). Consider the example shown in Figure 1. If a global fixpoint were achieved, then the analysis σ_d of the parameter `d` would include both `d1` and `d2`, and the analysis σ_s would be `s1` and `s2`. This analysis is not incorrect; these are all the possible values that `d` and `s` can hold over the course of the program. However, we might assume that there is an unsafe `strcpy` due to this, even when this is not the case. The problem is a matter of **specificity** – we assume that all invocations of `f` share the same context and analysis, and this is a general issue with dataflow analysis – it is conservative.

As a result, we perform **context-sensitive** analysis, in which a function's analysis is independent of other invocations of the functions by the caller. However, since this may be non-terminating (in the case of recursion), we limit the analysis until we reach a fixed point in the functions on the call stack; i.e., the analysis of a function will generalize any previous analysis of the function from mutually recursive calls, if any[6].

We simplify the analysis by assuming that the transition function for any function call is the identity function[7] – i.e., functions do not affect the analysis in the current function. Without this simplification, the cached analysis changes not only as a function of the parameters but also as a function of all of its contained function calls (who themselves are functions of their parameter set and internal function calls, and so on). This has two implications: functions cannot modify the analyses (i.e., reassign) of their parameters that are buffers, and functions cannot return a buffer type. Luckily, these two operations are not too common in cases that would affect our analysis (e.g., returned pointers are usually pointing to heap-allocated buffers).

[6] We are not sure if this is conventional.
[7] We are also not sure if this is conventional.

```
char buf[512], **pbuf, *buf2;
pbuf = &((char*)buf);
buf2 = *pbuf;
```

Figure 2: Importance of storing pointers to buffer origins

We defer to Chapters 8 and 10 of the *Program Analysis* textbook (linked under References) for more details on interprocedural analysis, such as a proof of precision and termination.

3.4 Buffer origin dataflow analysis (BODA)

3.4.1 Overview

Buffer A fixed-size stack-allocated (local variable) array.

Candidate buffer A virtual register that has an array or pointer type.

Buffer origin (Of a candidate buffer) Refers to the set of possible buffers or parameter candidate buffers that a candidate buffer may be *referring to*.

Note that when discussing buffer origins, "referring to" is only in the sense of memory equality (referential semantics), not in terms of the equality of its contents (value semantics). For example, after performing `strcpy(dst,src)`, `dst`'s analysis is not updated.

There are multiple ways to hold a reference to a value. For example, we want to be able to handle the scenario shown in Figure 2. In this example, we want `buf2` to refer to `buf`. We also know that `pbuf` refers to `buf`, but it is not equal to `buf` – instead, we maintain in our analysis that it may be a reference to `buf`, denoted as `&buf2`. Similarly, we want to keep track of all virtual registers that may refer to a buffer via more levels of indirection, i.e., `&&buf2`, `&&&buf2`, etc. This also works out nicely given that LLVM uses a load/store instruction set, in which we often have pointers referring to buffers. For example, variable definitions are implicitly pointers to the value type: the variable declaration `char buf[]` defines the virtual register `%buf` with type `[i8]*`.

3.4.2 Definition

The abstraction function for the BODA analysis is shown in (1). For a given candidate buffer b, the BODA analysis returns the set of all possible buffer origins may be referred to by b.

$$\alpha(b) = \{\text{possible buffer origins of } b\} \tag{1}$$

In this case, our lattice structure is the set lattice. For any set lattice, $\bot = \varnothing$, $\top =$ the set of all buffer origins, and the join operator ($\wedge$) is set union.

The transition functions for BODA analysis are defined below. (2) describes the analysis for an array declaration. (3), (4), and (5) address all memory operations (assignment) in LLVM. LLVM's `getelementptr` is more complex than a simple `load`, treating it as we treat the `load` opcode simplifies our analysis. The final rule, (6), is the catchall; for any non-assignment operator, the transition function ignores the instruction.

$$f(\texttt{\%buf = alloca [i8]*}, \sigma) = \sigma[\texttt{\%buf} \mapsto \{\&\texttt{buf}\}] \tag{2}$$

$$f(\texttt{store i8* \%p1, i8** \%p2}, \sigma) = \sigma[\texttt{\%p2} \mapsto \{\&b_o : b_o \in \sigma_{\texttt{\%p1}}\}] \tag{3}$$

$$f(\texttt{\%p2 = load i8** \%p1}, \sigma) = \sigma[\texttt{\%p2} \mapsto \{*b_o : b_o \in \sigma_{\texttt{\%p1}}\}] \tag{4}$$

$$f(\texttt{\%p2 = getelementptr i8** \%p1, 0, 0}, \sigma) = \sigma[\texttt{\%p2} \mapsto \{*b_o : b_o \in \sigma_{\texttt{\%p1}}\}] \tag{5}$$

$$f(\texttt{\%a = op \%b, \%c}, \sigma) = \sigma \tag{6}$$

3.4.3 Assumptions and caveats

Some major assumptions of the dataflow analysis note are listed below.

Limited to fixed-size stack-allocated buffers Only fixed-size buffers are tracked as possible buffer origins. Pointer variables are not regarded as possible buffer origins. Pointers allocated via other methods (heap allocation, or variable-length arrays (VLAs)) cannot be tracked due to requiring runtime information. Some cases of this may be trackable with the aid of constant propagation, but that would unnecessarily complicate our implementation.

No pointer arithmetic In an instruction such as `p=b+3`, where `p` is a pointer and `b` is a buffer, `b` is considered a buffer origin of `p`, even if it is not exactly the case. Future work may attempt to account for this.

Function calls do not affect dataflow analysis This means that the transition function for all function call instructions is the identity function. This has two implications: functions cannot modify the analysis of their parameters (e.g., a function cannot swap the buffer origins of its parameters), and functions cannot return a buffer. Allowing for this would greatly complicate the interprocedural analysis, and would require research to solve efficiently. We feel that omitting this should not detract much from the analysis, since functions do not often modify what their input buffers point to, and functions do not usually return pointers to stack-allocated buffers (returned buffers tend to be heap-allocated).

```
function BodaWorklist(f)
    B ← basic blocks in f
    mark B_0 dirty
    B_d ← B[0]
    while B_d is not empty do
        b ← pop from B_d
        if b is dirty then
            analyze(b)
            mark b clean
            if b not at fixpoint then
                for all b_s ∈ B : b_s succeeds b do
                    mark b_s dirty
                    insert b_s into B_d
                end for
            end if
        end if
    end while
end function
```

Figure 3: The worklist algorithm for basic blocks in a function

3.4.4 Per-function analysis: buffer origin dataflow analysis

The implementation of the dataflow analysis is fairly standard for a dataflow analysis, following the worklist algorithm for computing a fixpoint (at the function level) and using the join operator and transition function to update the analysis (at the basic-block/instruction level). See Figure 3 for the pseudocode of the worklist algorithm. This is a generic formulation that can be applied to any dataflow analysis.

The specifics of the choice of basic block to pop may affect the efficiency of the worklist algorithm (see Kildall's algorithm) but that is not the goal here and was not implemented. In our implementation, B_d is implemented as a simple LIFO queue.

3.4.5 Interprocedural analysis: CFG tracing (using BODA data)

See Figure 4 for the pseudocode for tracing the call graph. At each function invocation, each argument buffer is replaced with its potential buffer origins. This process happens recursively.

The algorithm as stated doesn't handle (mutual) recursion, which will cause an infinite loop. One method to achieve convergence (other than limiting recursion depth) is to use a previous analysis of the function on the call stack as the initial value for an analysis. If there is no change in the analysis of a particular function invocation, then the recursion stops. In other words, there is a convergence among the analysis of the functions along the call stack. We did not finish this in our implementation.

```
function TraceCallGraphRec(f, O)
    for all function invocations i_g() ∈ f do
        p_g ← [O_{i_g}/O]p_g
        if g is dangerous then
            warn about dangerous usage
        else if g is user-defined then
            TraceCallGraphRec(g, p_g)
        end if
    end for
end function

function TraceCallGraph(M)
    f_m ← main function of M
    TraceCallGraphRec(f_m, {})
end function
```

Figure 4: Trace call graph algorithm (not accounting for mutual recursion)

4 Samples and results

Several sample programs and their generated outputs are shown below. The outputs shown below are from the current implementation with the `-DDEBUG` compile flag turned off; with the flag on, much more information is shown about the dataflow analysis. We omit this for brevity.

Figure 5 shows the simplest example of buffer tracing: a pointer referencing a local buffer. In this case, the source buffer in `strcpy` will always be shorter than the destination buffer, which is safe if `b3` contains a well-formed (null-terminated) string. However, there is always the chance that this is not the case, so we warn about `strcpy` anyways.

Figure 6 shows an example where a pointer's referenced buffer is indeterminate. Since there is a possible path such that the source buffer is longer than the destination buffer, we warn about the dangerous usage.

Figure 7 shows an example of interprocedural analysis. Through the function call, the parameters are replaced with their buffer origins when analyzing the function call. The call to `strcpy` is able to determine that it uses an unsafe combination of source and destination buffers located in the `main` function.

Figure 8 shows another example of interprocedural analysis, demonstrating the recursive nature of the call graph tracing. We can track dangerous buffer usages from arbitrarily deep in the call stack.

Figure 9 illustrates the problem with the context-insensitivity of dataflow analysis noted in Figure 1. We are able to consider different function invocations as separate contexts, and thus avoid the false positive that a global dataflow analysis would flag.

Other noteworthy examples are loops and recursion. Loops (even non-terminating ones) should converge in the analysis. Analyzing (mutual) recursion does not terminate in our current implementation. We would also like to include more debugging information, such as backtrace of dangerous usages and source file position (line and column) of the declarations of buffer origins, in addition to the function name.

```c
int main() {
        char b1[8192], *b2, b3[512];
        b2 = b1;
        strcpy(b2, b3);
}
```

(a) Source

```
[Warning]: unsafe function strcpy()
        Possible destinations: main:b1[8192]
        Possible sources: main:b3[512]
```

(b) Output

Figure 5: Sample program: simple buffer tracking

```c
int main() {
        char src1[8192], src2[512], *src, dst[512];
        if (rand() % 2) {
                src = src1;
        } else {
                src = src2;
        }
        strcpy(dst, src);
}
```

(a) Source

```
[Warning]: unsafe function strcpy()
        Possible destinations: main:dst[512]
        Possible sources: main:src1[8192] main:src2[512]
        [Danger]: potential buffer overflow (source:8192 > destination:512)
```

(b) Output

Figure 6: Sample program: indeterminate buffer origin

```c
void f(char *dst, char *src) {
        strcpy(dst, src);
}

int main() {
        char b1[512], b2[4096];
        f(b1, b2);
}
```

(a) Source

```
[Warning]: unsafe function strcpy()
        Possible destinations: main:b1[512]
        Possible sources: main:b2[4096]
        [Danger]: potential buffer overflow (source:4096 > destination:512)
```

(b) Output

Figure 7: Sample program: buffer tracking through function invocations

```c
void g(char *s) {
        char buf[512];
        strcpy(buf, s);
}

void f(char *s) {
        g(s);
}

int main() {
        char str[8192];
        f(str);
}
```

(a) Source

```
[Warning]: unsafe function strcpy()
        Possible destinations: g:buf[512]
        Possible sources: main:str[8192]
        [Danger]: potential buffer overflow (source:8192 > destination:512)
```

(b) Output

Figure 8: Sample program: multi-level buffer tracking

```c
void f(char *d, char *s) {
        strcpy(d, s);
}

int main() {
        char b1[512], b2[512], b3[1024], b4[1024];
        f(b1, b2);
        f(b3, b4);
}
```

(a) Source

```
[Warning]: unsafe function strcpy()
        Possible destinations: main:b1[512]
        Possible sources: main:b2[512]
[Warning]: unsafe function strcpy()
        Possible destinations: main:b3[1024]
        Possible sources: main:b4[1024]
```

(b) Output

Figure 9: Sample program: specificity of call graph tracing

5 Conclusion

Buffer overflows are a very prevalent software vulnerability in low-level programming. We attempt to address common cases of buffer overflows by deep tracing of stack-allocated buffers throughout functions to well-known dangerous functions, using a custom dataflow analysis called buffer origin analysis. Our analysis successfully covers the given test scenarios, and is conservative in its analysis (as dataflow analysis tends to be). To achieve greater accuracy, call-graph tracing is applied for function calls.

This method of dataflow analysis, while applied in BODA to buffers, can be applied to track any local variable pointers throughout a program. We are not aware of any use cases other than buffer overflow detection at this time.

Due to a number of strong assumptions made in the analysis, there are a large number of uncovered scenarios that can be addressed in the future, in order to address more advanced use cases. Our algorithm may also be improved by handling recursive cases using the call-stack convergence method described earlier, as well as by improving logging output to use LLVM debugging info. It will also be informative to examine our program on more complicated programs than our artificial test cases, and to examine the implemenatation of existing static analysis tools and how they handle buffer overflow vulnerabilities.

6 Resources

Program Analysis **(Textbook)** `https://www.cs.cmu.edu/~aldrich/courses/17-396/program-analysis.pdf`

Security review: digital contact tracing

Jonathan Lam

09/22/21

1 Overview

Contact tracing is a method to identify individuals who have been in contact with people known to be infected. This technique came to prominence during the COVID-19 pandemic, primarily through the use of (non-mandatory) government- and privately-owned smartphone applications. Digital contact tracing is highly desirable because manual contact tracing uses a large number of human resources, and relies on people known to have COVID-19 to immediately report to a health authority who they have been in contact with recently. This process is not very accurate, as the infected person may not remember everyone they have been in contact with or may have been around unidentified strangers. Digital contact tracing aims to (semi-)automate the process of collecting information about who an individual has been in proximity with so that this process is carried out with greater speed and accuracy, using commonly-available smart devices (smartphones), and potentially offering better anonymity (in the case of decentralized networks). Governments of various scale have adopted official (but non-mandatory) digital contact tracing applications, but there are also non-government-sanctioned apps that exist as well.

Most contact tracing applications work using Bluetooth (e.g., BlueTrace [1]) or GPS (e.g., TraceTogether [4]), which are technologies available on most modern smartphones. (One of the newer systems, NOVID, uses ultrasound.) A user's smartphone generally shares anonymized, time-shifting tokens (Ephemeral ID's) with the other users in their proximity and keeps a local log of the interactions. Several governments (notably in Malaysia, Australia, and New Zealand) have also adopted QR code tagging associated with certain locations. In some systems there may still be some human interaction, such as to filter out likely false positives, but some systems attempt to completely automate the process. The reporting systems associated with

these techniques can be either centralized or decentralized; the latter are more recent (stemming from a MIT paper in May 2020 [3]) and arose due to privacy concerns.

2 Assets

Data assets:

- **User data**: This includes personal information (e.g., contact info) about users, where they have been, and what people they have been in contact with. Most of the security concerns are relevant here. **Confidentiality** (privacy) is the major concern here: we want to make sure that adversaries, or other parties in general (such as the government or organization that may have access to the location data) do not have access to this data. **Authorization**: the only people who should access this data are the health officials who are administering manual parts of the contact tracing. **Authenticity** (users should not be able to spoof their identity as someone else) and **integrity** (contact and location data should not be tampered with) also ensure that the data in the system is correct; loss of data or incorrect data would cause the contact tracing to be less effective.

Hardware assets:

- **User smartphones**: Vulnerabilities in the implementation of wireless protocols may result in the compromise of the smartphone. **Confidentiality** may be a concern: in Bluetooth an RF signal is emitted every 200ms, which can alert others of the user's presence. Confidentiality may also be compromised if the wireless protocol can be fingerprinted.
- **Centralized servers**: If the contact and location information is stored on a centralized server, then we may have to worry about **availability** (due to DoS attacks) or **tampering** by on-site workers. Decentralized networks are less at risk of these.

3 Adversaries/Threats

- **Government spying and tampering**: If the **government** has access to the (centralized) database on which the location and contact records are stored, then they can spy on user whereabouts and contact behavior. This would threaten **confidentiality** of user data. If they

are also given authorization to modify records (i.e., to hide the fact that a political official went somewhere), this would threaten **integrity**.

- **Network attacks**: A **tech-savvy COVID-denier** with a few Bluetooth devices at hand may attempt to cause incorrect data or loss of data. A Bluetooth device can plausibly be overloaded by a DoS attack performed by the adversary's Bluetooth devices, compromising **availability** of the contact tracing system.

4 Potential weaknesses/vulnerabilities

- **Bluetooth**: Bluetooth was already mentioned in the previous section, since implementation bugs may cause problems. A recent set of bugs in Bluetooth called Braktooth is known to allow DoS and even the ability to run arbitrary code, which can compromise the user device in general [5].

- **Tracking identifiers shared by clients**: Even if the identifiers shared by users are anonymized, they are still the same for each user. Since a user is continually emitting this identifier, an adversary may be able to identify the device associated with an anonymous ID.

5 Potential defenses

- To mitigate Bluetooth (or other networking) attacks, users should make sure that their devices have the most up-to-date firmware, and app creators should make sure to run their apps with the minimum permissions necessary to function correctly.

- To protect against tracking identifiers, identifiers are made ephemeral, i.e., "Ephemeral ID's" [1]. This mitigates the ability to identify a device ID because the ID is changed after a set interval of time. There is a tradeoff between how often the key changes and storage space. This is implemented in some contact tracing systems, but it is unknown whether all implement this.

6 Risks

The risk of surveillance deterred many people from the use of contact tracing. Many people might feel that the ability to track people (both their location

and who they contact) feels very much *1984*-ish. Decentralized networks were created as a response to decrease the risk. Confidentiality is also put at risk by local adversaries who are tracking static identifiers – this is mitigated by ephemeral identifiers but cannot be avoided completely.

The risk of Bluetooth vulnerabilities or other networking are probably fairly low. Using an established wireless technology is probably fairly robust, and new vulnerabilities are widely publicized. However, technologies that use more obscure or recent technologies (such as NOVID's use of ultrasound) may be more susceptible to implemenation bugs simply due to smaller community and smaller security efforts aimed at the technology.

7 Reflections

The contact tracing technology has rapidly evolved as it became popularized during the COVID-19 pandemic. There have been several variants of this technology: different proximity-detection technologies, decentralized and centralized reporting techniques, and ephemeral vs. non-ephemeral ID's. Future evolutions may involve IoT devices specialized to this, which would introduce a number of new considerations about usability (would the user want to carry around a new hardware token?), anonymity (doesn't require a login or use of a personal smartphone), and security (dependent on the implementation of the IoT device).

Current decentralized systems offer low-risk of surveillance, or any of the other risks associated with a centralized database. This should offer protection against the largest concern users have: privacy of their location data. However, use of contact tracing requires devices in the proximity to be aware of a user's device which (despite being anonymized) may still trigger the irrational fear of being watched.

Despite the advantages digital contact tracing has to offer in terms of accuracy and convenience, it is still not widely adopted during the pandemic, mostly due to surveillance concerns [4]. A team at the Australian National University concluded that the efficacy also relates to a high testing rate that wasn't being actualized [2]. As a result, it is difficult for contact tracing to achieve its maximimal efficacy.

References

[1] Jason Bay et al. "BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders". In: *Government Technology Agency-Singapore, Tech. Rep* (2020).

[2] Manuel Cebrian. "The past, present and future of digital contact tracing". In: *Nature Electronics* 4.1 (2021), pp. 2–4.

[3] Ramesh Raskar et al. *Apps Gone Rogue: Maintaining Personal Privacy in an Epidemic*. 2020. arXiv: `2003.08567 [cs.CR]`.

[4] Dewey Sim and Kimberly Lim. "Coronavirus: why aren't Singapore residents using the TraceTogether contact-tracing app". In: *South China Morning Post* 18 (2020).

[5] Satsuki Then. *BrakTooth vulnerability impacts Bluetooth devices*. Aug. 2021. URL: `https://asset-group.github.io/disclosures/braktooth/` (visited on 09/22/2021).

ECE455: Student Self-Survey

Jonathan Lam

09/06/2021

Contents

1. What does "cyber-security" mean to you?

   I understand cybersecurity to be mostly about making sure computer software (and hardware) runs as it should run, and in particular giving access to the intended parties to perform certain actions or access data. I believe good security usually means writing robust software that is intuitive for the user, but also involves the quick detection, investigation, and resolving of attacks.

2. What do you expect to get out of this course?

   As you mentioned in the first lecture, I'm hoping to get a general overview of the different subfields within cybersecurity. I do not have much background in cybersecurity – the little that I know mostly comes from web development to prevent against common attacks.

   The three more-concrete goals I have from this course are: looking for ways to further secure the software that I write; understanding the ways to protect my own digital actions; and looking for potential research areas for myself in the future.

3. Do you have any special topics or ideas you'd like to explore?

   I've recently been interested in (and am currently taking an independent study on) program analysis, i.e., the formal analysis of programming languages. This seems to be an area of active research and has implications for cybersecurity – it allows a person to "formally verify" an implementation: check that an implementation does exactly what the programmer expects it to, such as preventing race conditions, memory leaks, and bad failure modes. There are current attempts to make fully-verified compilers and operating systems now. I'm still very new to this topic, so I would like to learn about it in this class as well.

CUDB: A simple document-based NoSQL DBMS

Jonathan Lam, Derek Lee, Victor Zhang

The Cooper Union for the Advancement of Science and Art

2021/12/14

What is CUDB?

Cooper **U**nion **D**ata**B**ase a.k.a. CUDA++

Document model Natural, schema-less representation

File-backed persistence Mmapv1-inspired buffer/storage engine

B-tree indexing User-declared indices speed up range queries

MongoDB-like CRUD API Familiar NoSQL CRUD operations

Value Types

```rust
pub enum Value {
    /// Special type for unique identification,
    /// similar to MongoDB's `_id`.
    Id(String),
    /// Fixed-size integer type.
    Int32(i32),
    /// Arbitrary-length strings.
    String(String),
    /// Recursive documents (hashtables).
    Dict(Document),
    /// Array types.
    Array(Vec<Value>),
}
```

Figure: `cudb::value::Value` definition

Collection management API

```
impl Collection {
    /// Create a collection from a path.
    pub fn from(path: &str) -> Collection {}

    /// Close collection and underlying file pointer.
    pub fn close(self) {}

    /// Drop collection data and indices.
    pub fn drop(self) {}
}
```

Figure: `cudb::db::Collection` management API

Collection CRUD API

```
impl Collection {
    insert_one(doc: Document) {}
    insert_many(docs: Vec<Document>) {}
    find_one(query: Query) -> Option<Document> {}
    find_many(query: Query) -> Vec<Document> {}
    find_all() -> Vec<Document> {}
    update_one(query: Query, update: Document) {}
    update_many(query: ConstraintDocument,
                update: Document) {}
    delete_one(query: Query) {}
    delete_many(query: ConstraintDocument) {}
}
```

Figure: CRUD API

Sample query syntax

```
SELECT name, dob
FROM students
WHERE gpa>3.0 AND grade<>9
ORDER BY gpa DESC;
```

Figure: SQL

Sample query syntax

```
db.students.find({
  gpa: { $gt: 3.0 },
  grade: { $ne: 9 }
}, {
    name: 1,
    dob: 1
}).sort({ gpa: -1 });
```

Figure: MQL (JS)

Sample query syntax

```
Query(
  constraints: {
    ["gpa"]: Constraint::GreaterThan(Value::Double(3.0))),
    ["grade"]: Constraint::NotEquals(Value::Int32(9)))
  },
  projection: {
    ["name"]: Projection::Include,
    ["dob"]: Projection::Include
  },
  order: Some([
    ResultOrder::Desc(FieldPath)
  ])
)
```

Figure: CUDB (RON)

Index schema and index instance

```
// Index schema
index_schema = [
  ( field_path: ["a"], default: Value::Int(0) ),
  ( field_path: ["b", "c"], default: Value::String("World") ),
  ( field_path: ["b", "e"], default: Value::String("some value") ),
]

// Document
document = {
  "a": Value::Int32(42),
  "b": Value::Dict({
    "c": Value::String("Hi"),
    "d": Value::Int32(-2)
  })
}

// Index instance for this schema and document
index_instance = [
  Value::Int32(42),
  Value::String("Hi"),
  Value::String("some value")
]
```

Figure: Index schema and index instance

Performance and memory bounds

Operation	Time	Memory
Insert	$O(I \log C)$	$O(D)$
RUD (no index, unsorted)	$O(I + C)$	$O(CD)$
RUD (index, unsorted)	$\approx O(I + \log C)$	$\approx O(\log CD)$
RUD (no index, sorted)	$O(I + C \log C)$	$O(CD)$
RUD (index, sorted)	$\approx O(I + (\log C)(\log \log C))$	$\approx O(\log CD)$

Table: Expected performance characteristics

Resources

- GitHub
- API
- Report
- Presentation

ECE464 – Final Project Proposal

Jonathan Lam, Derek Lee, Victor Zhang

2021/10/19

1 Overview

We want to build a simple RDBMS to explore the concepts and challenges of physical design. In particular, we would like to incorporate the major ideas from System R described in the paper "Access Path Selection in a Relational Database Management System", such as:

- Data and index pages (or files), clustered indices if possible
- B-tree implementation of index pages (as described in the paper)
- Very simple access path selection (e.g., checking if a predicate matches an index); we won't have time for most optimizations

2 Functional requirements

This DBMS should support a simple subset of SQL queries with common clauses such as `SELECT`, `FROM`, `WHERE`, simple conditional clauses, and (time-permitting) simple nested queries and joins.

A simple interface (either a CLI or a socket interface and language API) will be developed to interface with and test the DBMS' capabilities. The capabilities of this minimal query language should be well documented.

For the sake of time, transactions and multithreading will only be attempted if time permits. This means that the ACID principles will not apply (which apply to transactions and require concurrency).

3 Evaluation

A test suite will be used to check the correctness of various queries specified in the minimal query language. A series of benchmarks may also be applied to test the empirical scalability of the system, and the scalability may be compared to existing RDBMSes.

ECE464 – Final Project Proposal

Jonathan Lam, Derek Lee, Victor Zhang

2021/11/08

1 Overview

We want to build a simple NoSQL database structured after MongoDB. This database will be in the form of a Rust library that supports basic CRUD operations similar to MQL operations and a document-based data model. Indices will have to be explicitly requested to be built by the user, and will be implemented using a B-tree data structure.

2 Functional requirements

The database should support the basic CRUD operations and MQL-like queries. The data must be persistent (i.e., the database should not be solely in-memory).

For sake of time, many non-essential features of MongoDB may not be implemented: a custom query language (like MQL), transactions, concurrency, ACID/BASE principles, replica sets, sharding, and cross-table operations such as $lookup.

3 Tentative architecture

3.1 Data structures

- Document (in-memory): hashtable or association list mapping field names to (type, value) pairs (since Rust is strictly-typed)
- Document (physical): JSON or BSON
- Collection: vector of documents
- Index: B-tree
- Query: information about collection(s), list of constraints (each constraint contains information about the field, operator, type, value), ordering
- Schema: (none?)

3.2 Architecture implementation details

Physical storage of records will be similar to MongoDB's MMAPv1 storage engine, which uses power-of-two fixed-size blocks and a residency bitmap (i.e., a free-blocks list); each document (after creation or update) will occupy the next free block that has size of the smallest power of two greater than or equal to the size of the document. This allows for some tradeoff between time and space efficiency. There is no provision against fragmentation due to an accumulation of free blocks. This scheme also prevents against an efficient clustered B-tree index, which would probably be made much more complex than if fixed-size records were necessary (MongoDB doesn't have clustered indices anyway, presumably for this reason).

Indices will be created on demand. Indices may comprise one or more fields (multi-field comparisons will be are lexicographic). If no indices match the query constraints (i.e., no index contains a subset of the fields on which a constraint is placed), then the collection is scanned sequentially. Fields present in a query but missing in a document will return a special NOEXIST value (this is subject to change – not sure how MongoDB handles this case).

Queries contain information about the collection to query, a set of constraints on fields, ordering/limit operations, and aggregation operations. (These are analogous to the FROM, WHERE, ORDER BY, LIMIT, GROUP BY, and aggregate operations in SQL.) As is traditionally done in MongoDB, the the structure of the query constraints models the structure of the data, using a similar recursive structure with constraints in the place of values. (Alternatively, nested constraints may be modeled using a "field path," e.g., "field1.field2.field3".)

4 Timeline

- 11/09: Finalized proposal
- 11/16: Implement data structures for documents, queries, collections, set up API, write unit tests
- 11/23: Implement MMAPv1, B-tree (indices), BSON
- 11/30: Implement CRUD operations
- 12/07: Finish implementation of queries, work on presentation
- 12/14: Presentation

5 Evaluation

A test suite will be used to check the correctness of various queries specified in the minimal query language. A series of benchmarks may also be applied to test the empirical scalability of the system, and the scalability may be compared to existing RDBMSs and NoSQL DBMSs.

CUDB: A simple document-based NoSQL DBMS

ECE464 Final Project

Jonathan Lam, Derek Lee, Victor Zhang

December 11, 2021

Contents

This document will include information pertinent to the functional and architectural design decisions, as well logistics of the final project for the sake of the class. The project source and build/test instructions can be found on GitHub at jlam55555/cudb. The API documentation can be found at `https://jlam55555.github.io/cudb`.

1 Abstract

CUDB (Cooper Union DataBase; alternatively CUDA++) is a simple document-based, MongoDB-inspired NoSQL database for learning purposes. The database is in the form of a Rust library that supports basic CRUD operations similar to MQL and a document-based data model. CUDB supports basic database features such as indexing using B-trees and persistence, but does not support many advanced features found in commercial DBMSes.

2 Changes from proposal

The original proposal can be found here. Surprisingly few changes have been made from the original proposal; though we were shrouded in ambiguity when writing the proposal, many of the proposed changes and much of the proposed schedule has stood up to plan. In particular, we support the following features:

- CRUD operations
- File-backed persistence using the OS's page cache similar to MongoDB's mmapv1
- Manually-created indices
- B-tree indexing
- Index- and table-scan lookups
- Nested queries using field paths

and don't support the following features:

- Clustering (MongoDB doesn't support this either)
- Transactions/ACID
- Concurrency
- Distributed store/sharding
- Schema definition or enforcement
- Aggregate operations and cross-table operations (e.g., `$lookup`)

The only changes from the original proposal were minor: we do not use BSON (instead using Rust's default binary serialization library), and the block allocation scheme does not have a residency map like mmapv1 (and is thus much simplified – more details can be found on the repo README). Both of these changes were made for the sake of simplicity, so that the project can be finished in time. The switch from BSON may make the storage format less portable but more performant (because it doesn't have to perform the conversion). The simplification to the block allocation scheme will cause a large loss of (space) efficiency when a database is delete- or update-heavy. In a insert-only database, it will not be inefficient.

Additionally, the original proposal was somewhat vague in describing the data structures and API regarding queries: over the course of the project, this had to be refined. The technical details of the final design can be found in Section 4, and on the online API documentation.

3 Functional overview

This section describes the CUDB public API and user-facing data structures, including idiosyncrasies and other design decisions.

3.1 Data structures

3.1.1 Documents and values

The `cudb::document::Document` struct allows for a "document" similar to that in MongoDB or Javascript/JSON. It is implemented as a `HashMap` mapping strings to `cudb::value::Value`s.

`cudb::value::Value` is an enum of possible value types. Its current definition is shown in Figure 1. Note that recursive documents (a.k.a., nested documents, or subdocuments) and array types are allowed. Note also that only scalar types of the same variant (i.e., two ints or two strings, but not a string and an int) are comparable using relational comparators or equality; comparisons between different variants or with a document or array type will fail. (This partial comparison will necessitate typed indices, which will be described in a later section.)

3.1.2 Collections

CUDB supports organizing documents into collections (`cudb::db::Collection`), but does not support organizing collections into databases like MongoDB. (This is not a problem, as cross-database operations are hardly, if ever, used.) Collections can be opened and closed using the management API shown in Figure 2. Each collection is backed by a file, and is opened by specifying the file path. If the file does not exist, then a new collection will be created (this is similar to MongoDB's behavior). The specifics of the query data structures will be discussed in the next section.

```rust
pub enum Value {
  /// Special type for unique identification,
  /// similar to MongoDB's `_id`.
  Id(String),
  /// Fixed-size integer type.
  Int32(i32),
  /// Arbitrary-length strings.
  String(String),
  /// Recursive documents (hashtables).
  Dict(Document),
  /// Array types.
  Array(Vec<Value>),
}
```

Figure 1: `cudb::value::Value` definition

```rust
impl Collection {
  /// Create a collection from a path.
  pub fn from(path: &str) -> Collection {}

  /// Close collection and underlying file pointer.
  pub fn close(self) {}

  /// Drop collection data and indices.
  pub fn drop(self) {}
}
```

Figure 2: `cudb::db::Collection` management API

Collections support the ordinary CRUD operations. The definition for the `cudb::db::Collection` API is shown in Figure 3.

3.1.3 Queries

A stereotypical simple (without joins or aggregate operations) query is shown in SQL, MongoDB (MQL), and CUDB (in RON format) in Figure 4. This simple query format shares a common format: a set of constraints (the SQL `WHERE` clause), a set of projections (the SQL `SELECT` clause), and a sort order (the SQL `ORDER BY` clause). In MQL, the constraints and projections are positional arguments in the `Collection::find()` function; however, we make them explicit in the `cudb::query::Query` struct. In particular, the `cudb::query::Query` struct comprises three fields: a `cudb::query::ConstraintDocument` (type-aliased to `HashMap<cudb::query::FieldPath, cudb::query::Constraint>`); a `cudb::query::ProjectionDocument` (type-aliased to `HashMap<cudb::query::FieldPath, cudb::query::F` and an optional sort ordering (`Option<Vec<cudb::query::ResultOrder>>`). Updating and deleting documents requires a constraint, and sort order may be provided for the `cudb::db::Collection::updateOne()` and `cudb::db::Collection::deleteOne()` operations. See the API documentation for more details.

3.1.4 Indices

To speed up queries, users can create an index (`cudb::index::IndexSchema`) on a collection. An index is a non-empty set of field names. An index may be used to speed up a query to logarithmic (B-tree search) rather than linear time (linear scan), if the index matches the query's constraints. There is no restriction on the number of indices on a collection. Indices must be created manually by the user.

A `cudb::index::IndexSchema` comprises an ordered set of field specifications (`cudb::index::FieldSpec`). Each field specification includes a field path (`cudb::query::FieldPath`) and a default value (`cudb::value::Value`). The default value serves both as the placeholder for missing values and the type specification for that field. The `cudb::db::Collection` API supports functions to create, list, and delete an index. See the documentation for more details.

Some notes about CUDB's indices:

Immutable indices The default values for fields in an index are immutable. To change the default values, the index must be deleted and re-created with the desired default value.

Failing to create an index due to a document If a document has a field contained in the index, and its value is of a different type than the default value specified in the index, then creating the index will fail.

Failing to create a document due to an index If an index on the collection contains one of the fields that is contained in a document to insert or update, and the type of the default value for that field in the index is

```rust
impl Collection {
  /// Insert one document.
  pub fn insert_one(&mut self, doc: Document) {}

  /// Insert a vector of documents.
  pub fn insert_many(&mut self, docs: Vec<Document>) {}

  /// Fetch at most one document matching the query.
  pub fn find_one(&mut self, query: Query)
    -> Option<Document> {}

  /// Fetch a vector of documents matching the query.
  pub fn find_many(&self, query: Query)
    -> Vec<Document> {}

  /// Update at most one document that matches the query.
  pub fn update_one(&self, query: ConstraintDocument,
                    update: UpdateDocument) {}

  /// Update all documents matching the query.
  pub fn update_many(&self, query: ConstraintDocument,
                     update: UpdateDocument) {}

  /// Replace at most one document that matches the query.
  pub fn replace_one(&self, query: ConstraintDocument,
                     replace: Document) {}

  /// Delete at most one document that matches the query.
  pub fn delete_one(&self, query: ConstraintDocument) {}

  /// Delete all documents that match the query.
  pub fn delete_many(&self, query: ConstraintDocument) {}
}
```

Figure 3: `cudb::db::Collection` CRUD API

```sql
SELECT name, dob
FROM students
WHERE gpa>3.0 AND grade<>9
ORDER BY gpa DESC;
```

(a) SQL

```
db.students.find({
  gpa: { $gt: 3.0 },
  grade: { $ne: 9 }
}, {
    name: 1,
    dob: 1
}).sort({ gpa: -1 });
```

(b) MQL

```
Query(
  constraints: {
    ["gpa"]: Constraint::GreaterThan(Value::Double(3.0))),
    ["grade"]: Constraint::NotEquals(Value::Int32(9)))
  },
  projection: {
    ["name"]: Projection::Include,
    ["dob"]: Projection::Include
  },
  order: Some([
    ResultOrder::Desc(FieldPath)
  ])
)
```

(c) CUDB

Figure 4: Stereotypical SQL, MQL, and CUDB queries

different than the type of the value in the document to insert or update, the insert or update operation will fail.

Failing to create an index due to an index If an existing index contains a field that the new index contains, and the fields have different type, then creating the new index will fail.

Indices and missing values Creating an index will not fail if a document does not contain one of its fields. Instead, it will be treated as if that field were given the specified default value.

Index matching An index will "match" a query if all of its fields are used in the query's constraints. Note that this is not optimal: in a B-tree, we can actually match an index to any query where the (unordered) constaint fields form some prefix of the (ordered) index fields. Our approach is taken for simplicity.

Index selection The selection process is fairly simple for our database: an index with the maximal number of matching fields (i.e., a matching index with the maximal number of fields). A better database should have better cost metrics due to cataloging or manual user input.

Primitive values Index fields can only include scalar types (i.e., not subdocuments or arrays). However, field paths can traverse subdocuments; only the final path component has to be a scalar type.

3.2 Sample usage

See the README on the GitHub repository for sample client code.

3.3 Expected performance characteristics

See Table 1. I represents the number of indices on a collection, and C indicates the cardinality of a collection. D represents the average size of a document in the collection.

Inserts Insertions require two steps: writing to the file and inserting into all indices. Writing to a file involves finding the insertion offset ($O(1)$ due to the simple buffer allocation scheme). Writing to each index (B-tree) takes $\log C$ time, for a total of $O(I \log C)$ time. The constant-time file allocation is simple and fast but memory-inefficient, because it does not attempt to reclaim memory.

Read/Update/Delete (RUD) Each RUD operation first involves looking for the best matching index, which takes time $O(I)$. Without a matching index, read/update/delete (RUD) updates require a full table scan. With a matching index, RUD operations are roughly logarithmic (due to a B-tree implementation). If all of the constraint fields are used in the index,

Operation	Time	Memory
Insert	$O(I \log C)$	$O(D)$
RUD (no index, unsorted)	$O(I + C)$	$O(CD)$
RUD (index, unsorted)	$\approx O(I + \log C)$	$\approx O(\log CD)$
RUD (no index, sorted)	$O(I + C \log C)$	$O(CD)$
RUD (index, sorted)	$\approx O(I + (\log C)(\log \log C))$	$\approx O(\log CD)$

Table 1: Expected performance characteristics

then the lookup is logarithmic; if there are additional fields in the index that are not contained in the index, then there needs to be an additional linear scan on the index-matching documents to filter out documents that do not match the remaining fields in the constraint. Sorts are implemented using the default Rust unstable sort (linearithmic time and in-place).

4 Architectural overview

4.1 Physical storage

A `cudb::db::Document` is file-backed. The data structures and routines to handle this physical storage are located in `cudb::mmapv1`. The physical representation of a collection is a "pool" (`cudb::mmapv1::Pool`), which is stored in a regular file.

The desired operations on a pool are a pool scan, pool insert, pool write, pool read, and pool delete (at some offset). To achieve this, the pool file is structured as a sequence of contiguous "blocks" (`cudb::mmapv1::block::Block`). Each block has an offset (in bytes from the beginning of the pool), and a size (a power of two between 32 and 1MB). Contained in each block are a fixed-size header (containing the following metadata: whether the block is (soft-)deleted, the size of the data in the block, and the total block size) and a serialized `cudb::document::Document`. The pool operations may return a wrapped version of a document (`cudb::mmapv1::TopLevelDocument`) including its block information. The pool also maintains a pointer to the end of the file as the insert location. With this setup, the pool operations are implemented as follows:

Read document at offset Seek to the offset in the file. Read the block header to get the block capacity and data size. Read the serialized document, deserialize it, and return the document.

Pool scan Start with an initial offset of zero. If you are not at the end of the pool, read the document header to get the block data size, soft-deleted status, and capacity. If the document is deleted, ignore it; otherwise, read at the current offset and push the document into a vector of documents. Advance the current offset by the block size and repeat.

Insert document Serialize the document and find the size d of the serialized data. Find the smallest power of two n such that $n > d + h$, where h is the (fixed) header size, also in bytes. Write a new block at the end of the file, and update the pool size. Return the pool offset of the document as a `cudb::mmapv1::TopLevelDocument`.

Write/update document at offset Serialize the document. Read the header at the specified offset. If the block size is too small to contain the updated document, soft-delete the document (see below) and insert the document as if it were a new document (allocating a suitably-sized block in the process). Return the (potentially-updated) pool offset of the document as a `cudb::mmapv1::TopLevelDocument`.

Delete document at offset Update the header at the specified offset by setting the soft-deleted flag to true.

The pool can be closed by simply closing the file. The pool can be deleted by simply deleting the file. When reopening the file, the only metadata/state that has to be restored is the current insert index, which is the end of the file (the size of the file in bytes), which is easy to calculate using some filesystem API.

This storage engine is named after MongoDB's old storage engine, mmapv1. CUDB's storage engine is similar to mmapv1 in several ways: documents are stored contiguously; documents are stored with power-of-two-sized allocations; and memory paging is managed by the OS. However, CUDB does not use `mmap` to bring pages into memory (despite the name), instead relying on the OS for its caching properties as well. This is a tradeoff of control over caching, in exchange for simplicity of implementation.

Note that there are a great number of possible error conditions if any of the following occur: pool file permissions changed, pool file corruption, or any operation at an invalid block offset. Thus, we assume that the filesystem is tamper-proof and that the user is never able to modify block offsets via encapsulation in the API's (e.g., they will never be exposed to a `cudb::mmapv1::block::Block` descriptor and thus not be able to perform an operation at an invalid offset). Of course, this is a very weak assumption, and only suitable for a simple academic project where we can assume no bad actors.

4.2 Indices and index schemas

There are two closely related data structures that are relevant to the discussion of "index" in CUDB. An "index schema" (`cudb::index::IndexSchema`) contains the definition of an index: an ordered collection of fields with a type and default value – this is what is referred to when saying "index" colloquially. An "index instance" (`cudb::index::Index`) is the ordered set of values from a document that correspond to the fields specified in a index schema. If the field is missing in the document, then it is given the specified default value.

As an example, consider the index schema shown in Figure 5. The index schema specifies three fields that we want to index by. If any constraint contains all three of these fields, then we may use this index schema to speed up that query. Note that each field path is specified by an array: i.e., the path to the field `"c"` within the subdocument `"b"` is given by the fieldpath `["b", "c"]`. The document is specified normally. The index instance is created by extracting the fields from the document, in the same order as in the index schema, and checking that the types are consistent. Note that the last field in the index schema does not exist in the document, so it is filled in with the default value specified by the schema.

For each index schema, a B-tree is created from all of the documents in the collection. The B-tree maps index instances to a set of block offsets of documents who have that index instance. A collection stores a hashmap of index schemas to corresponding B-trees. When a user declares and index schema, the B-tree for that index schema is created; if any of the documents have a type conflict with the index schema, building the index schema fails. Alternatively, if any other index schema has conflicting types, then the index schema creation also fails. Similarly, when a user inserts or updates a document, it must be updated in all index schemas; if the new document has any type conflicts with any of the existing index schemas, then the operation fails.

Note that indices are necessarily typed. This is because indices must be stored as (keys of) a B-tree, and thus must be totally-comparable. As mentioned when first discussing documents and values, only scalar values of the same variant (type) are comparable. Thus, the "type" of a field in the index schema is defined using a default value. Note that in the case of documents that are missing that field, they are treated and ordered as if their value was the default value.

4.3 Explanation of querying

It has already been discussed how to perform operations on a pool when a document offset is known. However, arguably the most important role of databases is to be able to query data based on some query constraint. This section will be an overview of the steps taken to perform a query operation with the aid of indices. (A query that does not use any indices is the degenerate case, in which the matched index is empty.)

Once we have the ability to query, the CRUD operations are straightforward to implement. In other words, querying can be thought of as a function that takes a constraint and returns a set of offsets, which are used as inputs for pool operations. The CRUD operations can be summarized as the following combination of querying and pool operations shown in Table 2.

The first three steps (access path selection, creating B-tree ranges, and the linear scan over all fields referenced in the constraints[1]) are solely concerned

[1] We don't have to scan over every field. We only have to scan over all fields that weren't in the index and all fields that were in the index that had exclusive bounds. The reason for the latter is explained in Section 4.3.2.

```
// Index schema
index_schema = [
  ( field_path: ["a"], default: Value::Int(0) ),
  ( field_path: ["b", "c"], default: Value::String("World") ),
  ( field_path: ["b", "e"], default: Value::String("some value") ),
]

// Document
document = {
  "a": Value::Int32(42),
  "b": Value::Dict({
    "c": Value::String("Hi"),
    "d": Value::Int32(-2)
  })
}

// Index instance for this schema and document
index_instance = [
  Value::Int32(42),
  Value::String("Hi"),
  Value::String("some value")
]
```

Figure 5: Illustrative example of an index schema and an index instance (using RON-like syntax for illustrative purposes)

Operation	Query?	Pool operation
Create/Insert	No	Insert
Read/Find	Yes	Find at location
Update	Yes	Update at location
Delete	Yes	Mark deleted at location

Table 2: Summary of CRUD operations

with the constraints document. The last step (sorting and projection) deals with the specified result order and field projection, and may not be applicable to all CRUD operations.

4.3.1 Access path selection (a.k.a. finding the best matching index)

The "access path selection" step of DBMS query optimizers is usually used to select the best method to find the requested records. In the case of CUDB, there is only one form of access path: find the best matching index, and then linearly scan to filter out documents that don't match the constraints[2]. Thus, the only access path selection lies in selecting the best matching index.

This step is very simple. A matching (candidate) index schema is one that has all of its fields match the constraint document (including type matching); i.e., the fields in the constraint document should subsume the index schema. A maximal candidate index schema is chosen as the "best" index schema. If there are multiple maximal candidate index schemas, an arbitrary one is chosen.

Note that this is about the simplest query optimization scheme possible, and could be greatly improved with catalog information. It would also have to be greatly revamped if there is a more advanced set of query access paths.

4.3.2 Creating B-tree ranges to scan

B-trees can provide range searches: if the key type is some totally-ordered set S, then we can easily query all documents in a specified interval $[S_1, S_2]$. The time complexity of finding this range and iterating over its documents is $O(\log C + |C[S_1 : S_2]|)$, where $|C[S_1, S_2]|$ represents the number of documents in the collection in the specified interval. Note that equality is a degenerate form of range search, in which $S_1 = S_2$.

In our case, our key type is a n-tuple of values, sorted lexicographically. Thus a range must be specified as a pair of n-tuples. Assuming we have an index schema with integer fields $[a, b, c]$ (we relax the notation in this section to illustrate B-tree properties), and a constraint $(a \leq 3) \wedge (b = 2) \wedge (c \geq 5)$, then this translates to the following n-tuple range:

$$\begin{bmatrix} -\infty \\ 2 \\ 5 \end{bmatrix} \leq \begin{bmatrix} a \\ b \\ c \end{bmatrix} \leq \begin{bmatrix} 3 \\ 2 \\ \infty \end{bmatrix}$$

This approach is relatively straightforward, and is how range searches on multi-field indices are constructed, but there are a few difficulties with this approach:

Mixing inclusive and exclusive constraints Assume in the above example that the constraint included the exclusive bound $(a < 3)$. Then the above problem could not be expressed using an inequality, because there is a mix

[2]Theoretically, we only have to scan over all non-index fields. However, due to a quirk in our implementation, we must scan over every field. The reason for this is explained in Section 4.3.2.

of inclusive and exclusive constraints in the upper bound of the range. To solve this, CUDB changes all fields to inclusive bounds, and then in the linear scan re-checks the constraints on index fields to filter out elements which might have been falsely satisfied due to the incorrect bounds.

Upper and lower bounds on unbounded types In the above example, values for negative and positive infinity are required for lower and upper bounds of one-sided inequalities. This can be solved in two ways: by establishing a minimum and maximum value for each value type, and substituting that value for infinity; or by establishing special value types `Value::PosInf` and `Value::NegInf` that always compare larger and smaller than any other value, respectively (care must be taken that these values may never exist in a document). The former approach was taken, but the latter approach was thought of after the fact and is more robust in the case of unbounded types such as strings, for which there is no true maximum value and only an approximation for a maximum value can be used.

Conjunction (intersection) and disjunction (union) of constraints Consider a conjunctive constraint on a single field such as $(d > 3) \wedge (d < 5)$. This requires merging the ranges of the left and right subconstraints. Also consider a disjunctive constraint on a single field such as $(e < 3) \vee (d > 5)$. In this case, we have to split the query into two ranges. In general, we can expect to represent any set of range constraints as a set of disjoint continuous ranges in disjunctive normal form (DNF); each continuous range is represented by a single or pair of constraints, and these may all be connected together via disjunctions.

Additional research into how range searches with a diverse set of constraints are implemented on multi-field indices may be greatly helpful for simplifying this process. An alternative proposed scheme is to use chained B-trees, where each field has its own level of B-tree and thus may be constrained independently; however, this is prohibitive in the number of B-trees it spawns. It is unknown how MongoDB implements this.

4.3.3 Linear scan over non-index fields

The previous step filters out all documents that don't match the constraint on the fields specified in the index. An additional step is required to filter out all documents that don't match the constraint on the fields not specified in the index.

Note that if no index is declared or if no index matches the constraint, this is simply searching using a linear scan on the entire collection.

Note that this step should also check the constraints on the index fields, due to problems with perfect matching on index fields in the previous step. In other words, the previous step will produce a superset of the matching documents, but some of those documents may be false positives (in the case of documents

missing index fields, where we use the default value instead, or the problem with inclusive/exclusive bounds).

Note that this step can also check more complicated constraints than index constraints. Recall that indices were constrained to scalar types (which have a total ordering, assuming all values of the field have the same type). In this step, CUDB can check constraints such as the `cudb::query::Constraint::In` constraint, which checks for the existence of the value in a set of values.

In this step, the idea of what it means for a constraint to match an inconsistent value becomes relevant. At this step, some fields are either missing or of the wrong type. For simplicity of implementation, if a value is either missing or of the wrong type, then it will not match the document. Note that this is even true of the not equals constraint; in other words, neither checking equality or inequality against a value will return a document. This may feel inconsistent, but it is an issue with the "three-valued boolean logic" or nullable values. Future work can be done making this behavior more intuitive.

4.3.4 Sorting and projection

Sorting is performed on the documents matching the constraints using Rust's standard `Vec::sort_unstable` function. Projection is performed by mapping the result set through a projection function guided by the projection constraints.

5 Evaluation

At the time of writing this document, we still have to finish queries/CRUD operations on the DBMS. Evaluations of CUDB for speed and correctness will be added to the README once this is complete.

6 Known bugs/incomplete features

6.1 Assumptions

Unit and integration tests were written to test for common bugs and use cases. The assumption is that CUDB will be run under intended conditions. E.g., the following are not accounted for:

Data corruption We assume that the OS/block device is robust against bitrot, and that there are no malicious actors modifying the backing store file.

Concurrent database access We assume that only a single process is accessing a collection at a time. This may be enforced using a filesystem lock (?), but we haven't done this yet.

Insufficient disk space We assume that the inserted data will not exhaust available disk space. (Note that this may be a poor assumption due to the simplistic block allocation scheme.)

Indices fit in RAM While our data pages are distributed throughout page files and can be paged-in with random-access/seek operations in the file, we don't have the same functionality with our indices because we use the builtin B-tree implementation and have little control over how it's serialized in memory. Thus all B-tree indices are loaded into and assumed to fit within RAM.

Reasonable OS page cache scheme The buffer manager simply reads pages from disk when necessary. We assume that (in a *nix environment) blocks from the backing store file are cached in the page cache so that we can quickly retrieve information. We also assume that page caches are invalidated once available memory runs low (e.g., in a LRU fashion). Our mmapv1-like implementation exerts no explicit mmap caching (ironically, given its name), instead relying on the OS to perform it for us.

6.2 Known issues

Some known issues of CUDB are:

Inefficient page allocation scheme The memory/buffer manager is extremely simple for the purposes of this project. First of all, it relies on the underlying OS buffer manager and does not attempt to smartly pre-fetch or evict pages. Also, currently the buffer manager never reclaims deleted pages. This allocation scheme is simple because we simply keep a reference to the end of the pool as a place to allocate new buffers, and this is fine for insert-only buffers, but is highly inefficient for collections that involve much deleting or resizing. A smarter memory allocator would keep a list of free spaces (perhaps in a residency bitmap) and would have better alignment criteria (e.g., block-aligning records). The design of CUDB is fairly modular, and thus the memory allocator can be easily switched by replacing the `cudb::mmapv1` module.

Inelegant/inefficient query range selectors for multi-field indices All searches using an index are implemented as (inclusive) range searches over the index's B-tree, followed by a linear scan to match other fields that are not contained in the index. In the standard library B-tree implementation, there is a method to search over some interval (with optionally closed, open, or unbounded endpoints). However, in our case our B-tree keys are vectors of field values, sorted lexicographically; there is no (easy?) way to sort with different endpoint types on a per-field basis. Thus, the range search is performed over an inclusive range, which raises some issues. One issue is that inclusive upper- and lower-bounds have to be defined on all types, including unbounded types (e.g., strings; an arbitrary upper bound of string comprises 32 characters with the value 255 was chosen.). Another issue is that we cannot simultaneously accommodate inclusive and exclusive bounds, so we have to re-check all constraints on the index fields even

though we use the index's B-tree. More research is necessary to figure out how multi-field indices are implemented in MongoDB.

Overlapping ranges in conjunctive clauses An `OR` constraint in an index field will cause two ranges to be searched. For now, the two sub-constraints in the conjunction are assumed to be disjoint (non-overlapping). If they are non-disjoint, then the overlapping regions will be iterated twice. (This can be solved by checking for and merging non-disjoint regions. We just haven't gotten around to that yet.)

Poor error handling Much of the error handling is performed through the use of the `panic!()` macro. While this is considered "safe" error handling in Rust, it is not very intuitive and leads to a poor user experience. Rather, more graceful error handling should be done through by returning `Option` or `Result` types.

6.3 Missing features

Most of the important missing features are mentioned above. Some missing features that may be relatively simple to implement (given our existing progress) include:

Automatic ID's. Currently, there is an ID type supported but it is unused. To distinguish between documents with the same value, the user should manually create a unique ID field.

Collection schema Enforce a specification on each document in the collection. CUDB already enforces a specification on any field that is part of an index; this functionality can be furthur extended to the entire document. Of course, this will somewhat limit the flexibility of the documents.

Database-level organization Currently, there are only collection-level operations. CUDB does not support organization of collections into databases.

Aggregation operators CUDB does not support anything like the MongoDB Aggregation Framework, such as grouping operations or cross-table lookups.

7 Conclusion

It was a great learning experience for all of us. This project allowed us to learn about Rust, a programmer's favorite. Rust was a good choice for a combination of speed, high-level safety-features, and understandable documentation/error messages. We were also able to think through many of the difficult design decisions that govern schema-less, dynamically-typed, document-based DBMSes like MongoDB. While we did not implement many of the more advanced features related to scalable and reliable (distributed) systems that are characteristic of NoSQL DBMSes, we were also forced to consider some of those design details as well.

Program Analysis

Prof. Sable

Independent Study Syllabus

1 Overview

Program analysis is a logic-based approach to analyzing software. One of the most important uses is to verify the correctness of programs – i.e., that it functions exactly as the programmer intends. For example, verification is useful for checking the correctness of new programming paradigms (such as declarative concurrent languages or intermittent programming). Other uses of program analysis are to find general bugs, augment software test frameworks, and optimize software. Possible resources for this I.S. are:

- Stanford: CS 357: Advanced Topics in Formal Methods
- CMU: 17-355/17-665/17-819 Program Analysis
- Resources for Teaching with Formal Methods

The primary text and schedule are based off of the CMU course. The text is *Program Analysis* by Jonathan Aldrich, Claire Le Goues, and Rohan Padhye.

2 Workload

The workload would mostly consist of readings and weekly class discussions. Supplementary exercises are also available on the CMU course webpage. There would also be a final project.

Two potential final projects are:

- Perform research on an advanced topic (e.g., the topics covered at the end of the CMU course), and present the knowledge in a report and verbal presentation.
- Show some of the results from the class using formal methods software, such as Coq.

3 Weekly Schedule

Some of the chapters are out of order – we mostly follow the CMU schedule.

1. Initial discussion and course planning
2. Ch1-3: Introduction, Program Representation, and Syntactic Analysis
3. Ch4: Dataflow Analysis and Abstract Interpretation
4. Ch5: Data Analysis Examples
5. Ch6: Dataflow Analysis Termination and Correctness
6. Ch7: Widening Operators and Collecting Semantics
7. Ch8: Interprocedural Analysis
8. Ch9: Control-Flow Analysis for Functional Languages
9. Ch11: Hoare-style Verification
10. Ch13: Symbolic Execution
11. Ch14: Concolic Testing
12. Ch16: Fuzz Testing
13. Ch12: Satisfiability Modulo Theories (SMT)
14. Ch15: Oracle-Guided Synthesis
15. Final Project Presentations

What We Can Learn About Running from Barefoot Running: An Evolutionary Medical Perspective[1]

Daniel E. Lieberman

Philip Blumin, Jonathan Lam, Joshua Yoon

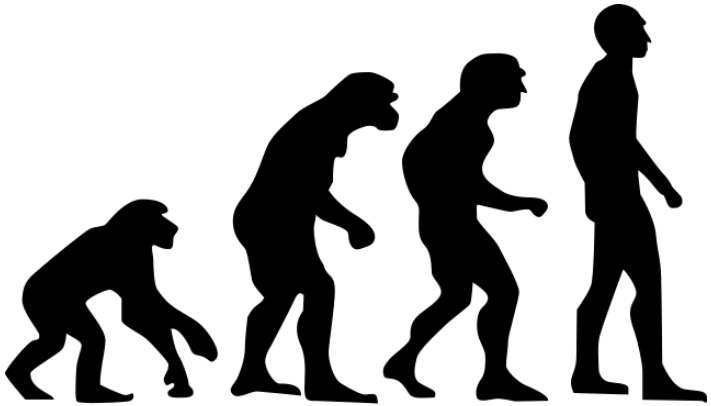
[1] Lieberman, Daniel E. "What we can learn about running from barefoot running: an evolutionary medical perspective." *Exercise and sport sciences reviews* 40.2 (2012): 63-72.

Introduction

- People’s and researcher’s perspectives on barefoot running
- Reasons why runners get injured a lot
  - Running is by nature an injurious activity
  - People are not acquainted to log distance running to do biomechanical abnormalities (asymmetries)
  - “Training errors”
- Barefoot running may help runners avoid injuries
- Author’s proposed hypothesis
  - Human bodies adapted to barefoot style
  - Generate less forceful impact peaks, which may strengthen feet

Why Does Evolution Matter

- Evolutionary field medicine
- Mismatch hypothesis
- Consequences shoes could have on injuries
  - Shoes limit proprioception
  - Modern shoes could encourage different running forms that human nature not used to
  - Shoes can lead to weak and inflexible feet
- Evolutionary medicine perspectives
  - Correct null hypothesis is that running barefoot is less injurious than running in a shoe

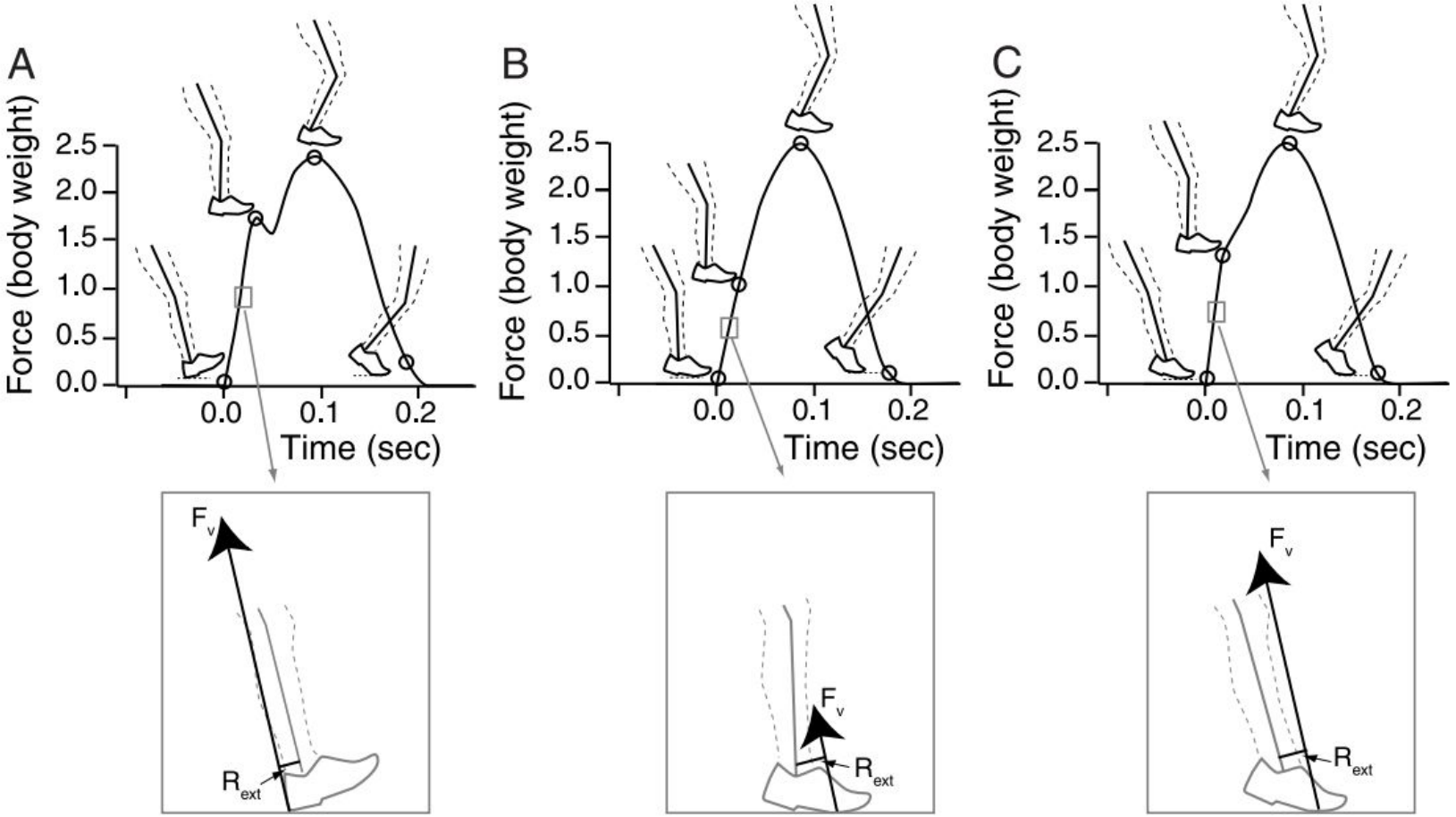


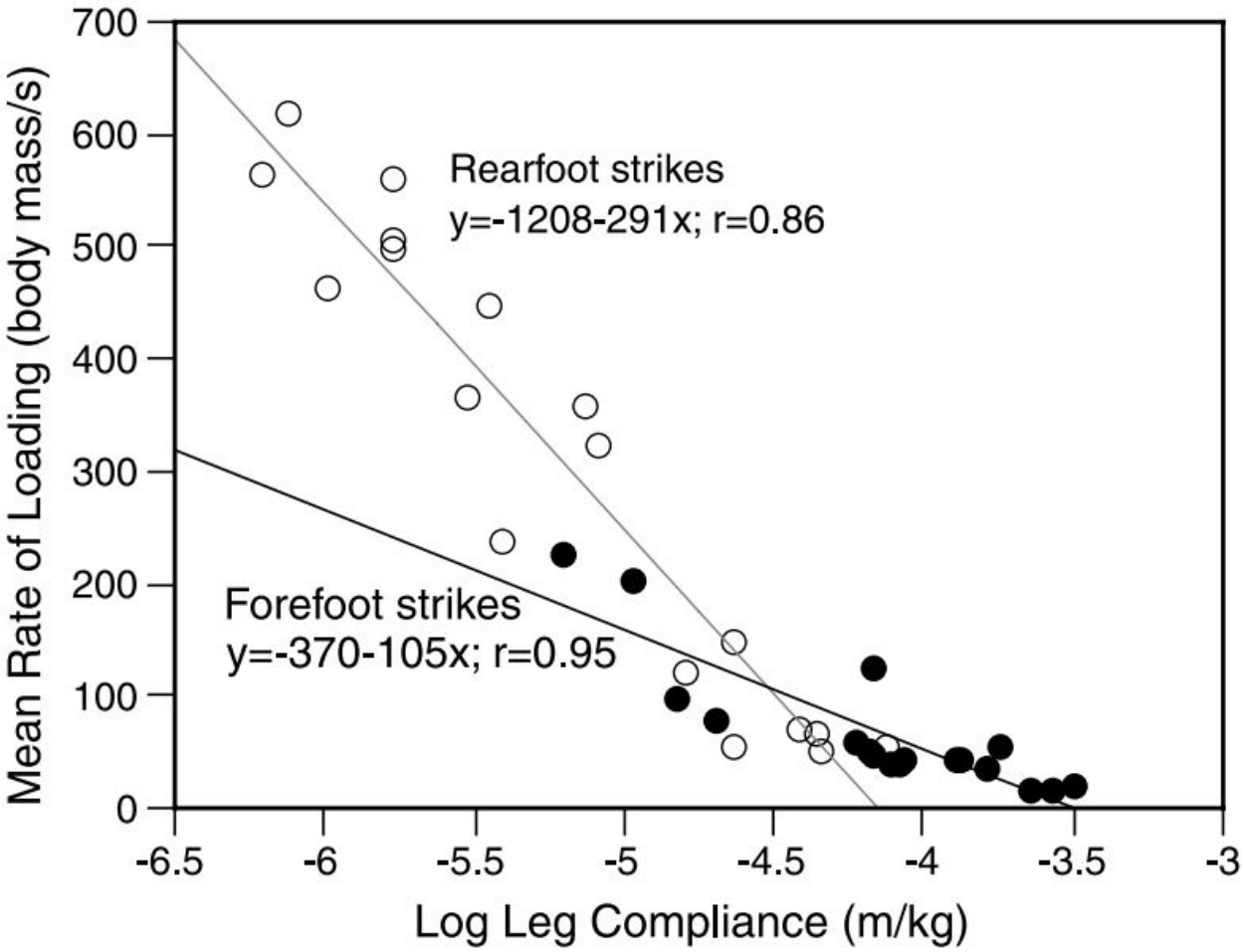
What Do We Know About Barefoot Running?

- Use habitually shod runners (people running in shoes) for previous studies
- Findings
  - Shod runners likely to rearfoot strike (RFS) while running on **soft surfaces like grass**
  - Shod runners likely to forefoot strike (FFS) and midfoot strike (MFS) when running on **hard surfaces**
  - Habitual barefoot runners have relatively short strides and a fast stride rate regardless of speed
    - Over 170 steps per minute
    - Short stride explains why barefoot runners land on foot more vertically aligned with knee and hip

Footstrike

- Rear vs. mid. vs. front footstrike (RFS, MFS, FFS)
- RFS causes **much higher initial ground reaction force**
  - Running shoes greatly decrease rate of loading but not as much peak load
- Barefoot runners tend to MFS or FFS
- RFS: **stiff ankle**, high impulse/rate of reversal of momentum
- FFS: **compliant ankle**, controlled dorsiflexion
- Several ways to modify lower extremity compliance other than using shoes
  - E.g., shorter strides, more knee flexion, less overstride


A
B
C
Force (body weight)
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.1
0.2
Time (sec)
F_v
R_ext


Mean Rate of Loading (body mass/s)
700
600
500
400
300
200
100
0
-6.5
-6
-5.5
-5
-4.5
-4
-3.5
-3
Log Leg Compliance (m/kg)
Rearfoot strikes
y=-1208-291x; r=0.86
Forefoot strikes
y=-370-105x; r=0.95

Stride Rate and Length

- Elite shod runners: 170-180 steps per minute (spm)
- Nonelite shod runners: 150-160spm
  - Why? We don’t know
- Nonelite barefoot runners: 175-182spm

Stride Rate and Length

- Elite shod runners: 170-180 steps per minute (spm)
- Nonelite shod runners: 150-160spm
  - Why? We don’t know
- Nonelite barefoot runners: 175-182spm

Why shorten stride?

- Tends to avoid RFS
- Knee, ankle tends to be more flexed and compliant
- Requires less plantarflexion to FFS

Anatomical Adaptations

Calluses form on toe and heel due to stimulation from friction

- Do not reduce impact but protect against injury

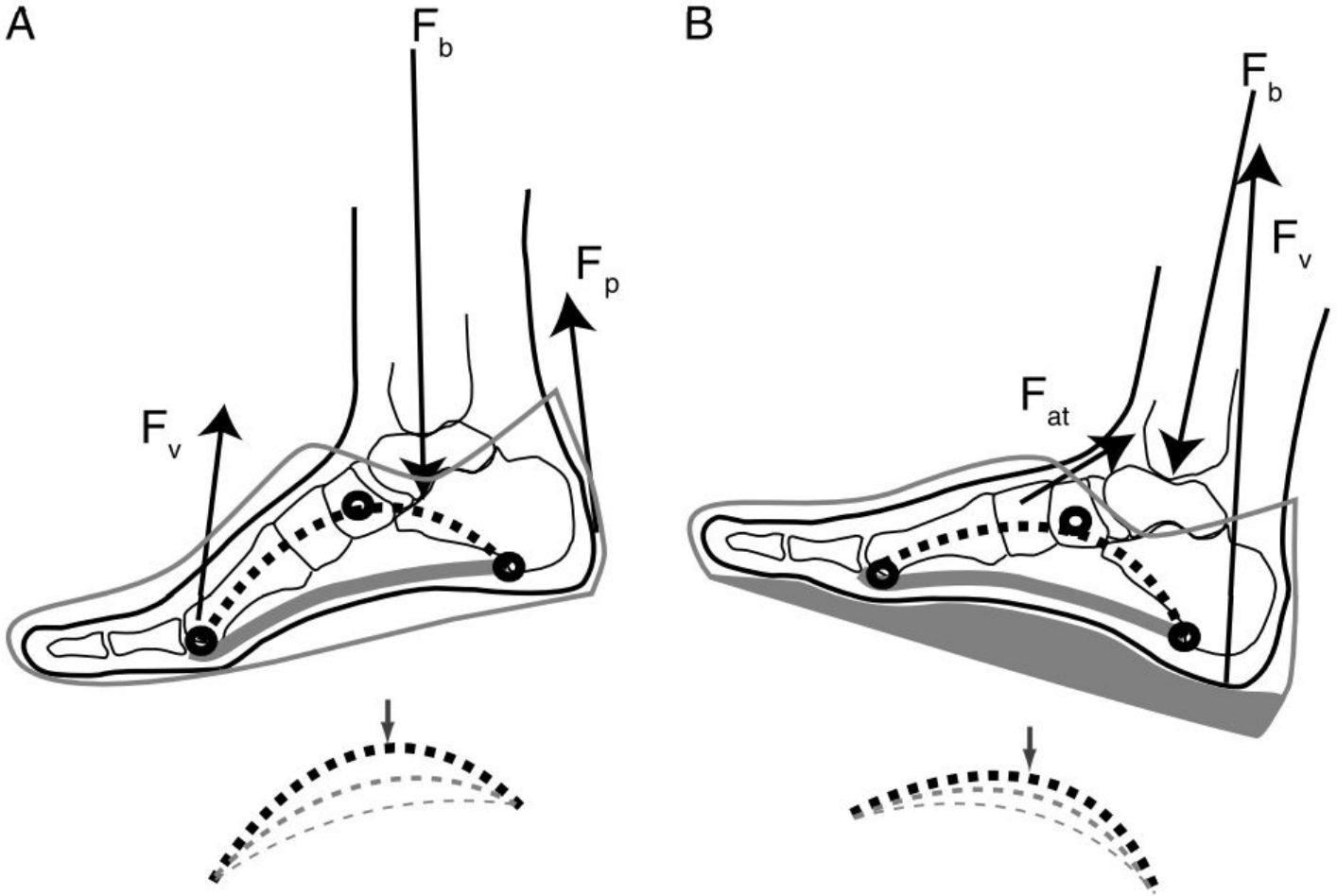
Anatomical Adaptations

Calluses form on toe and heel due to stimulation from friction

- Do not reduce impact but protect against injury

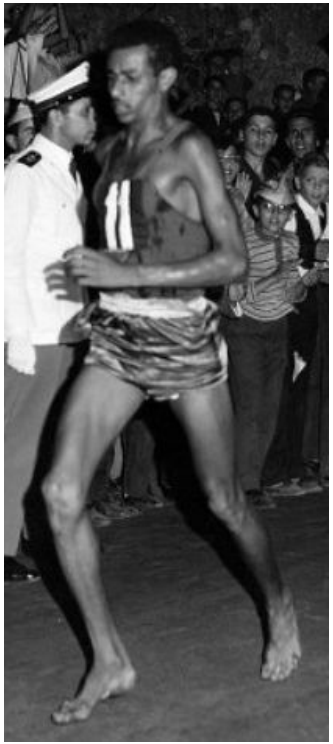
FFS/MFS cause more eccentric loads than RFS → stronger plantar muscles

- Both the plantar flexors (posterior leg muscles) and sole muscles
- Foot arch gets stretched more due to FFS than RFS


A
F b
F p
F v
B
F b
F v
F at

Performance

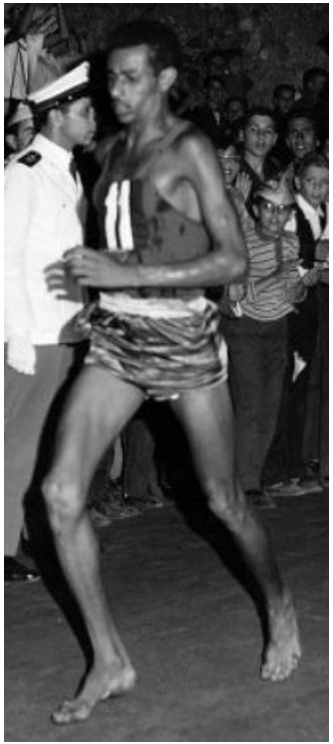
- FFS runners hold many world records



Abebe Bikila

Performance

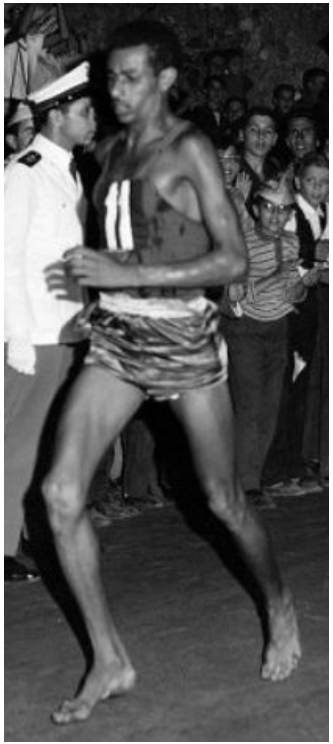
- FFS runners hold many world records
- Shoe Mass: Increases running cost by 1% for every 100g per unit per mass



Abebe Bikila

Performance

- FFS runners hold many world records
- Shoe Mass: Increases running cost by 1% for every 100g per unit per mass
- Running in minimal shoes is 2.4% - 3.3% more economical than running in standard running shoes



Abebe Bikila

How Might Barefoot Running be Relevant to Injury?

- “What about the way that barefoot runners tend to run affects injury rates and patterns?”

How Might Barefoot Running be Relevant to Injury?

- “What about the way that barefoot runners tend to run affects injury rates and patterns?”
- Way you run vs what you wear

How Might Barefoot Running be Relevant to Injury?

- “What about the way that barefoot runners tend to run affects injury rates and patterns?”
- Way you run vs what you wear
- Impact peak vs high loads on Achilles tendon and plantarflexors vs overstriding

How Might Barefoot Running be Relevant to Injury?

- “What about the way that barefoot runners tend to run affects injury rates and patterns?”
- Way you run vs what you wear
- Impact peak vs high loads on Achilles tendon and plantarflexors vs overstriding
- Shoe cushioning vs Foot Sensors and Proprioception vs Leg stiffness

How Might Barefoot Running be Relevant to Injury?

- “What about the way that barefoot runners tend to run affects injury rates and patterns?”
- Way you run vs what you wear
- Impact peak vs high loads on Achilles tendon and plantarflexors vs overstriding
- Shoe cushioning vs Foot Sensors and Proprioception vs Leg stiffness
- Shoe protection vs Foot Sensors and Proprioception

Conclusion

- Running barefoot is not inherently bad.

Conclusion

- Running barefoot is not inherently bad.
- “Humans not only evolved to run but also to run barefoot.”

Conclusion

- Running barefoot is not inherently bad.
- “Humans not only evolved to run but also to run barefoot.”
- Questions:
  - “How much do variations in running form …. affect injury rates?”

Conclusion

- Running barefoot is not inherently bad.
- “Humans not only evolved to run but also to run barefoot.”
- Questions:
  - “How much do variations in running form …. affect injury rates?”
  - “How much does the lack of prioprioception in a minimal shoe affect running form?”

Conclusion

- Running barefoot is not inherently bad.
- “Humans not only evolved to run but also to run barefoot.”
- Questions:
  - “How much do variations in running form …. affect injury rates?”
  - “How much does the lack of prioprioception in a minimal shoe affect running form?”
  - “Can we identify which runners are most likely to benefit from or should avoid barefoot running?”

Conclusion

- Barefoot runners have the following characteristics:
  - more proprioceptive feedback
  - shorter strides
  - higher stride frequency
  - avoid RFS and lessen impact peaks,
  - strong feet

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

09 / 27 / 21

Final Paper Topic Proposal

I want to study the *communication* between the native people of the Arctic. Clearly, this includes spoken and written language: this can involve the origin of words and how they fused as cultures collided, and a comparative analysis between different languages. We have already been informally introduced to several vocabulary from the Arctic, especially terms that don't have a direct analog in English.

It will also be interesting to study non-verbal communication and language idioms, and how these were affected by the environment. Communication also includes transportation and travel, which is much more difficult, so communication has to be able to handle this (fault-tolerance).

Communication is also an everlasting concern in electrical engineering, especially as related to efficiency (information theory) versus usability, transmission technologies, and fault-tolerance. These all have analogues to natural language and it may be interesting to draw comparisons when possible.

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

10 / 04 / 21

Final Paper Topic Proposal – Revised

I want to study the notion of death for the native people of the Arctic. We have encountered human mortality in the Arctic many times, both in fiction (Shelley) and nonfiction (failed northern expeditions, Kpomassie). In particular, Kpomassie brings up many haunting images of death, not only of humans, and in ways that range from horrifying to vibrant. Importantly, death appears to live much closer to the average person, and there seems to be a greater detachment between the native people and others in order to better prepare themselves for death.

Also in Kpomassie's paper, I was inspired by the image of elderly Inuit hunters performing "suicide" by walking out into the cold, never to return, or staying behind in an igloo, never to leave. It is silent and accepting, haunting and noble, and I would like to investigate what other sources have to say about this custom. You mentioned in class a film about "preparing for death" – that may be an interesting parallel, albeit in a different culture.

There are many aspects of death, apparently mostly accidental, so that may be too broad a topic. I think I would like to focus on the views of death by the elderly population. But if this yields too little information, I may broaden my search. This may involve anything from the language relating to death, rituals and stories relating to death, statistics on the age demographics, etc.

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

10 / 11 / 21

Final Paper Topic Proposal – Revised Again

(Sorry this is brief, I really didn't have much time to work on it this week.)

Area of research: The perception of death and mortality by the Inuit people, especially by the elders. What it means to prepare for death (if applicable) and why.

Materials to be researched:

- Kpomassie: This was the inspiration for this topic, already a lot of material to study in more detail from this book. The other works from this class less so. We also get some notion of death of explorers in Beattie's work, although all of it is via secondhand speculation and investigation.
- There are quite a few articles that pop up when Googling some combination of the keywords"inuit," "death," "arctic," and "stories." The last keyword is not the most related, but it may give some insights, as stories/legends/rituals probably have many mentions of death. Sample sources (and probably many more, need to sort out good sources):
  https://lithub.com/death-and-dying-in-the-canadian-arctic/
  https://dartcenter.org/resources/death-arctic-community-grieves-father-fights-change
  https://storymaps.arcgis.com/stories/2d0e3bb60cd24e40bc798d92dc0910bb
  http://www.inuitcontact.ca/artifacts/pdf/Inuit_legend.pdf
- Resources related to the death of other cultures (e.g., that Japanese death), and death of other animals (which bore semblance to the imagery of "preparing for death" mentioned in Kpomassie's narrative).

Question(s): What are the Inuit people's views on death (including life after death)? When is one ready to die (if at all)? How have these views been shaped by the Arctic environment and wildlife?

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

10 / 10 / 21

Response to Excerpt's from Beattie and Geiger's *Frozen in Time*

We have finally reached a mention of a "true" North-West Passage (the Simpson Strait), even if the men on the Franklin expedition did not know at the time that it was a NWP. We also hear about the first successful traversal of a NWP in 1903 by Amundsen, almost half a century after the final Franklin expedition. We then return in another eighty years with Beatty's forensic treatment. It is interesting to compare the differences in perspective between these three accounts.

Chapter 7 of *Frozen in Time* details various attempts at recovering the bodies or further details of the fates of the explorers in the years shortly after the expedition. The nature of the evidence is primarily in three forms: verifiable artifacts of the Franklin expedition (mostly through trading with the Inuit people), anecdotal stories of human remains, and second-hand stories from the Inuit people. Of these, only the first category are verifiably true. The second category, as discovered by Beatty, is more than likely to have some false positives due to human remains from people other than those from the Franklin expedition. And the Inuit people, who only happen upon the Franklin expeditioners by chance, can only provide sparse accounts to the best of their memory, also without much chance for verifiability. At this time in the mid-19th century, cameras were not used in the recovery missions (most likely; I don't know enough about camera technology history), and we are only presented artistic depictions of various scenes, such as the the depiction of M'Clintock discovering the lifeboat containing skeletons (87). Moreover, there is almost no surviving written work by the explorers: we do get a mysterious "Peglar papers" with backwards writing (81) and the notes left by Lieutenant Gore and Captain Fitzjames (83), but other books that had been found had been carelessly destroyed by Inuit children (75). The lack of concrete evidence, especially written evidence, provides much towards the

mystery of the Franklin expedition. If a single written first-hand history of a man (e.g., if one of the books that were ripped up by the Inuit children) were retrieved, the Franklin expedition would be much less well known.

Beattie only briefly mentions the Gjoa, whose six-person crew led by Amundsen was the first ship to successfully traverse a NWP. The only detail that is mentioned is that the boat was an “old wooden sloop” (100), that Amundsen was inspired by Franklin’s 1819 expedition, and that Amundsen would go on to die in a plane crash in the Arctic in 1928. However, from a quick perusal of the Wikipedia page on the Gjoa, I discovered that the boat and crew are so small in response to the Franklin tragedy – the reasoning is that “intended to live off the limited resources of the land and sea through which he was to travel, and reasoned that the land could sustain only a tiny crew (this had been a cause of the catastrophic failure of John Franklin's expedition fifty years previously) … [and] her shallow draught would help her traverse the shoals of the Arctic straits.” Unlike Franklin, Amundsen was well aware of the possibility of spending multiple winters stuck in the ice, which it was. The Wikipedia article also notes that the ship was stuck for two years in the same spot (like Franklin’s ship) but the crew spent much time learning from the Inuit people, thus avoiding the cultural failure of not learning from the Inuit people on how to be self-sufficient (and also avoiding scurvy on the way).

Finally, Beattie’s journey in the 1980’s was a breath of fresh air for me, as it felt like the first modern account of scientific investigation in our readings thus far. We are able to achieve a modern level of confidence on various conjectures – the evidence is of a very different nature, via studies of bone samples in cannibalism scenarios and bone sample identification. While it is obvious that over a century’s worth of Arctic climate had erased more of the evidence, it is interesting that only one skeleton of the 129 men was identified. We do not get any accounts of finding Inuit skeletons from the earlier explorers, so it is unclear whether those earlier explorers attributed all human remains to the doomed expedition, or if they ignored unidentified skeletons. It gives proper closure to the accusation of cannibalism, at the cost of making the Franklin expedition much less an object of the imagination.

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

10 / 18 / 21

Response to Blum, Herskos, Singh: Ecomedia

I felt that these texts were much more difficult than the previous (long) texts we have read so far: the Blum paper for the density of ideas, and the description of the two artists' works about the poles (Herskos and Singh) because I do not easily grasp artistic subtleties (e.g., I cannot hope to imagine what many of Singh's works have to do with the Arctic, such as the sand timer with the magnetic dust). The Blum paper in particular was charged with a richer vocabulary and density of ideas, so that I feel that I could use a lot more time digesting it.

The title of Blum's book, *The News at the End of the World*, intrigued me. At face value, the poles are the *extremes* of the Earth. But to call them the *end* of the Earth – what exactly does that mean? The term is used colloquially to mean some sort of extrema, but the Earth has a spherical rather than a linear geometry. We may reach the pole (a point on the surface of the sphere, or a ray from the center of the Earth) and then walk past it to begin moving southward again, and in this way it is not strictly an "end." On the other hand, we are also talking about the "end" in the temporal dimension, as climate change is bringing the poles to their finale. Arguably, the second meaning of the word "end" may be the more important one, as Blum repeatedly talks about the "motility" of the poles and man in the poles. This means that the ice is changing over time, perhaps with some movement, but more generally shifting in various ways – emitting trapped gases, opening undersea caves for explorers, or changing form to become seawater.

Hersko dramatizes not the beauty of the Arctic, but the story of its end. The melting of the poles is drawn akin to a second Holocaust. This characterization gives the disaster a more human and personal aspect, rather than appealing to the scale of the calamity that is climate change. For instance,

she takes time to examine the tragedy of snail shells dissolving due to ocean acidification. Bloom describes the purpose of this approach: “In some ways her work addresses the failure of perception and cognition, the result of which is our inability to deal with critical changes facing us over extended time” (Bloom 21) – while we cannot comprehend the commonplace statistics about flooding levels or a global temperature difference of two degrees Celsius (after all, daily temperatures fluctuate much more wildly), we can observe the dying snail and mourn the loss of its shell.

When reading about Hersko’s work, I was thinking about two movies that have affected me most strongly: *Princess Mononoke* and *Laputa: Castle in the Sky* by Studio Ghibli. Ghibli’s director, Hayao Miyazaki, is well-known for being an environmentalist, and Ghibli’s animated movies are famous for their humanity, partially due to subtleties such as their pacing: mixing significant and insignificant moments (perhaps like calving glaciers versus a partially-dissolved snail shell). These two movies are about, respectively, the end of a gentle nature and the end of a grand civilization, both at the hand of greedy men. While such a plot in movies usually feels whimsical and dull to me in many movies (“just another dystopian apocalypse”) Miyazaki’s movies have the character of great wonder and great tragedy. I feel that this has much to do with focusing on the small, and giving a human-sized narrative to something otherwise too large to feel sad about, in the same way Hersko aspires to do.

When thinking about ends (or future ends), I am tempted to make the comparison once again to space (as Singh also notes). Perhaps in our lifetime, we may be able to terraform the Moon or Mars. We already have probes that have reached the edge of the solar system, and have scanned much farther than that. The ability of man to affect its environment, and even its planet – as Blum notes, man does not only live *on* the planet, we live *with* it now – means that we have the ability to cause harm at the planetary level. Now, when new planets are our frontier as continents used to be, perhaps there may be a realization and a tragedy akin to climate change on the scale of multiplanetary systems waiting after the height of space exploration.

Jonathan Lam

Prof. Germano

HUM324 Polar Imagination

09 / 26 / 21

Without Schedule nor Urgency

Response to First Half of *An African in Greenland*

The idea of the seasonal day-night cycle has been a recurring theme of one of the major differences between the Arctic and life in the temperate zone. Lopez explores the concept quite physically, explaining what would happen to the position of the sun as dimensions of time-of-year and latitude vary. He also explains how the flora and fauna have been naturally selected to survive this seasonal change. But through Kpomassie's account we see human (psychological) responses to this change, especially in contrast to warmer climates.

As Kpomassie is on his final leg to Greenland aboard the *Martin S*, he already loses track of the day-night cycle, and finds himself without schedule and "never [knowing] quite when to go to bed" (Kpomassie 75). He wonders about how the inhabitants sleep – he finds that, during the summer months, sleep is light and often interrupted.

The lack of daily rhythmicity may be shocking to one who is familiar with the circadian rhythm. This involves a regular 24-hour sleep-awake cycle that maintains hormonal levels (and thus regular bodily functions) and lack of it may cause weight gain and impulsivity, according to a study by Rockefeller University (Society for Neuroscience). Recent research suggests that the Circadian rhythm is still present in Arctic species, so it is not restricted to the temperate zones (Walter et al.). We observe the manifestation of a severe psychological effect: the "near insomnia" (Kpomassie 139) during the summer months and the "polar hysteria of the Arctic autumn" (Kpomassie 138) that occurs as the days elongate. The seasonality of this pathology is suggestive of the Circadian rhythm, although this is speculative on my part.

An important predicate to the disease other than the seasonal changes, and a recurring theme throughout the narrative, is that of urgency and idleness. Kpomassie mentions that the ones most likely to be affected are the ones who also suffer "darkness and inactivity … Generals, being always busy, were not affected" (Kpomassie 139). Luckily, our narrator is very active and ceaselessly amazed (albeit not always positively) throughout his adventures, so he is not at high risk of inactivity and thus this hysteria. But Kpomassie is able to observe this in Erik, one of the (apparently average) residents.

If we consider the activity levels of the residents of Greenland, they largely seem to live slow lives. This is evidenced by his various accounts of the Greenlanders: the jailees at the Godthab (who are quite impassive at their jailing); the fishermen at Frederikshab (who care more about seal and birds and conversing during storms than fishing – "I could appreciate the Greenlanders' helping one another outside their villages but lamented their ability to get on with the job" (129)); the incessant partying, sexual encounters, and visiting habits of those at Cape Farewell; and the general drinking habits (which is not a feature isolated to Greenland but is definitely characteristic). In the penultimate case, there are only two true hunters and Kpomassie seems puzzled at whether the rest of the village actually has jobs or are purely living off a Danish allowance from the government. From his experience with strict rules in Togo restricting many of his actions, and his recent encounter of bustling European cities, where the busy-ness of a life governed by schedules and deadlines is familiar to us readers, this life seems quite relaxed and carefree.

The polar north necessitates a disorientation from the day-night cycle (lack of schedule) but the connection with a disorientation from active life (lack of urgency) is fuzzier.

One plausible explanation for the apparent laid-back-ness is that it is simply as protection against the harshness of the environment. It seems any sort of travel, even at the southern tip of Greenland (as noted by the shipwrecks around Cape Farewell), is perilous, never mind during the winter months. In order to preserve their life and lifespan, and since commercial fishing and allowances are enough to live off, there is no need for urgency.

Another (less plausible) explanation is that the Greenlanders (subconsciously) still do abide by the schedule dictated by the sun, and experience our daily deadlines as yearly experiences. This means that their life may be one of as many “days” as years. This can be metaphorically extended in many ways – perhaps this means that their lives’ ups and downs are much more majestic and exaggerated than those in the temperate zones where state of mind is reset every 24-hours. Perhaps our daily disasters are their yearly ones, and our yearly crises their lifelong ones. Perhaps the identity crisis felt by elderly Eskimo people who are miserable enough to commit suicide when they are unfit to hunt anymore is akin to the modern mid-life crisis, but that mid-life crisis that will be endured for a decade in a fast-paced life may be the twilight remainder of an Arctic dweller’s life.

We haven’t yet (in terms of book progress) encountered the Eskimo people that Kpomassie idealizes, so this analysis may be only relevant for the laid-back people of southern Greenland.

Bibliography

Society for Neuroscience. "Disruption Of Circadian Rhythms Affects Both Brain And Body, Mouse Study Finds." *ScienceDaily*. ScienceDaily, 28 October 2009. <www.sciencedaily.com/releases/2009/10/091026225744.htm>.

Arnold, Walter, et al. "Circadian rhythmicity persists through the Polar night and midnight sun in Svalbard reindeer." *Scientific reports* 8.1 (2018): 1-12.

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

10 / 04 / 21

Detachment

Response to *An African in Greenland* 2

The tone over the second half of the book evolves from fanciful and imaginative to that of ordinary life. We see this change in his reaction to and perception of several aspects of northern Greenland: the resistance to death, the acceptance of polygamic customs, and the willingness to return.

If the frequency of deaths were not evident enough in the first half of the book, this motif returns stronger in the second. The nature of the deaths also becomes more vivid, gruesome, or otherwise haunting – drunk parents falling on their baby, babies devoured by dogs, puppies devoured by dogs, a middle-aged man devoured by dogs, an execution of dogs suspected of manslaughter (as in Isabel's response, dogs feature quite prominently in the latter part of the book), able-bodied men falling through thin ice and drowning, a carelessly drunk murder due to a relationship dispute, an accidental gunshot through the leg in a canoe, and likely many more that are not mentioned in the story. We obtain a fairly objective story of each of these mishaps, but there doesn't seem to be a strong emotional reaction by Kpomassie evoked by any of them. Understandably, his perception conforms to match that of the native people after living with them so long: dogs become regular wild animals that are useful but dangerous, and the elevated mortality rate and the methods of death are not strange. It is not only a lack of fear, but also a lack of hatred or sadness. This reminds me of the "mind of winter" (Stevens 1) that Wallace Stevens refers to in "The Snow Man", the mindset that allows Kpomassie to "not to think \ Of any misery in the sound of the wind" (Stevens 7-8) and to "behold … \ Nothing that is not there and the nothing that is" (Stevens 14-15). I can imagine something that is "not there" may refer to the imagination or the memory of what is lost; the "nothing that is" refers to the physical absence of a

person. In other words, it seems like the Greenland people must detach themselves from others, and in this way protect themselves from sadness and fear of losing others. And Kpomassie has learnt this detachment, evident by his emotionless stories.

This detachment can loosely be considered one of the reasons for the polygamous ways of the Greenlanders. Kpomassie discovers that the wife-swapping custom (and the girlfriend-swapping that likely derives from this) is the result of “motions for survival” (Kpomassie 230) rather than idleness, and there is an implicit social structure and rules. They do this to essential join families, increasing the chance of survival if one family suffers a casualty. There is the detachment from one’s own family in order to ensure stronger bonds overall; this sounds very similar to kingdoms marrying off their princesses to form alliances. A noble sacrifice.

We also see the evolution of detachment in eating habits. Thodoris mentions that there is a repetitive treatment of the bloody food eaten with bare hands – blood becomes prevalent as a sign of food and survival rather than that of death. There is so much blood, and occasionally it is human blood – but I believe that the connection to human mortality is largely lost. (What is lost with it?)

The final detachment is that of Kpomassie from Greenland and his fantasy of the Arctic. This is no minor feat; he knows that this is the height of his lifetime, and turning back may mean he may not have the chance to return. When he reaches Thule, and hears from Robert the story of his brother’s death and the story of Arnarnguangssaq, the frequency of Kpomassie’s interjections connecting the stories to those of his homeland become more frequent. His dream of listening to these stories of the “true” Eskimo people is realized, and then he detaches himself from the completed dream in order to not linger on it forever. His journey has completed its natural course like that of a lifetime, and there is little regret as he leaves.

The question then becomes, what does he feel at the end of his journey? Is it a renewed excitement, to reunite with his family? As a reader, I feel a bittersweet wholesomeness, a gentleness, which contrasts with many aspects of his adventure.

Jonathan Lam

Prof. Germano

HUM-324 Polar Imagination

9 / 10 / 21

Whale Magic

(Response to the Prologue and Chapter 1 from Lopez's *Arctic Dreams*)

I was quite captivated by the prologue to *Arctic Dreams*, much more than I was affected by Walton's accounts of the Arctic (i.e., Shelley's imaginative view). In *Frankenstein*, we are presented the scenery more in terms of generalities and expectations: being "nearly surrounded by ice, which closed in the ship on all sides" (Shelley, Letter 4), "vast and irregular plains of ice, which seemed to have no end" (Shelley, Letter 4), or "the sun is for ever visible, its broad disk just skirting the horizon and diffusing a perpetual splendour" (Shelley, Letter 1). As a reader, this seems no more than an intense winter, or somewhat farther north, but is not too far removed from my sense of reality. The latter is the most absurd detail to someone living in the temperate zone, but still not relayed in great detail compared to Lopez's treatment.

Walton inures himself to the the harshness of the Arctic in whaling expeditions. In the prologue, Lopez also provides a narrative of the whalers in the Arctic, but the everyday life is much more explicit – and, due to the nature of whaling, much more unsettling and surreal than those of *Frankenstein*: these creatures, unlike Victor Frankenstein's creation, are not of near-human proportions, and the lurking dangers are capable of causing widespread calamities. Even the plain numbers are too large to size up in most people's minds: the harvested whales from the *Cumbrian* provided "236 tons of oil [and] … more than four and a half tones of whalebone" (Lopez 2). In its mouth are "blades nearly 14 feet long" (Lopez 3) and the body "wrapped in blubber as much as 20 inches thick" (Lopez 3). Perhaps I am dwelling too long on this subject, but, as an engineer, it is hard not to size up an object when measurements appear. There's a quote from the movie *Boyhood* that captures this sentiment:

> “I mean, what makes you think that elves are any more magical than something like... like a whale? You know what I mean? What if I told you a story about how underneath the ocean, there was this giant sea mammal that used sonar and sang songs and it was so big that its heart was the size of a car and you could crawl through the arteries? I mean, you'd think that was pretty magical, right?” (*Boyhood*)

The quote is a response to a boy asking about real magic in the world, and is a fairly convincing statement that “magic” is that which is so far from comprehension that it doesn’t feel real or scientific. We also have the “singing” of the whales (Lopez mentions that, by the time of the *Cumbrian*’s journey, Europeans “were unaware that the Greenland right actually ‘sang’” (Lopez 8)), which adds to the whales’ mystery, and compounds the eerie “high note that eventually faded away to a very low note” (Lopez 5) indicating Arctic gales.

The most haunting image is that of the Greenland right who had gently pushed the *Cumbrian*. “Awakened by [the *Cumbrian*’s] approach, she swam slowly once around the ship and then put her head calmly to the bow and began to push. She pushed the ship backward for two minutes before the transfixed crew reacted with harpoons. The incident left the men unsettled. They flinched against such occasional eeriness in their work … They heard no whistling that year … – but they had not liked the whale pushing against them, as though urging them to go back” (Lopez 4). This particular gentle whale, kin of the playful dolphins, capable of dragging down ships but instead harmlessly nudging it, becomes “the carnage of wealth” (Lopez 6). Is this is an unending pain of hunters, and especially those who must hunt for a living? The innocence of a deer or a duck will not feed the hunter’s family, and so we might say that a hunter cannot be both empathetic and successful. But a whale is different story – it is so magical, and its power to drag or push a ship is as powerful as the immutable weather. One must wonder what exists in the mind of a godly object as it pushed the boat, and as it perished.

The lack of empathy is a recurring theme. There is the objective nature of whaling, but the interactions with the people are likewise. Lopez describes the fascination of the Inuit people to the

Europeans, and vice versa, but there is never the notion of care or respect. Survival of their own, lack of cross-pollination of ideas, cause both sides to condescend on the other. Just as whaling would eventually endanger the whale population, smallpox engendered by the Europeans spread unhindered through the Inuit villages, causing an estimated 90 percent decrease in their population (Lopez 10). Somehow, man has become one of the irregular forces of the Arctic that make it so inhospitable, causing genocides like the "ten-year series of late spring snowstorms [that] prevented lesser snow geese from ever laying their eggs" (Lopez 32), or the "spring storms that have swept hundreds of thousands of infant harp seals into the sea, where they have drowned" (Lopez 32), or "October rainstorm [that] created a layer of ground ice that, later, musk-oxen could not break open to feed" (Lopez 32). In this case, we have the "thousands of European men that arrived in boats and caused the unrestrained hunting of the Greenland right whale."

Much of Chapter 1 of *Arctic Dreams* is fascinating to me, but primarily in the sense of wanting to obtain more information, rather than in the awe-inspiring sense of the Prologue. It is useful to learn about how the sun moves (in much greater detail than given in *Frankenstein*) and how life conforms to the harsh year-round day. The departure from the norms of temperate life justify the sailors' wonderment.

(Do we need a bibliography for this kind of informal assignment?)

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

10/25/21

Response to *The Narrative of Arthur Gordon Pym*

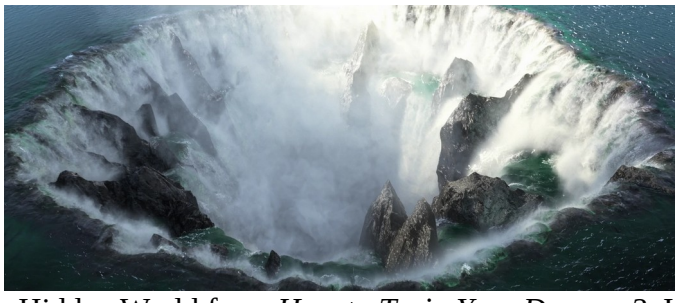
This response is a series of disparate observations more than a cohesive essay on a single topic.

The journey to the Antarctic is, unlike all of the journeys encountered thus far, not its original intention, but a series of chance circumstances, such as the sudden decision to avoid the Strait of Magellan. In the previous readings, whether the journeyer was in search of whales, reaching the pole, finding a NWP, or seeking forensic evidence of the ones who sought the NWP, the original goal was explicit and pole-related, and the ulterior motive of reaching the pole was not an unthinkable side-effect. However, simply by chance, there is the drifting southward on their first voyage, "from north to south, *not less than five-and-twenty* [120] *degrees!*" (Poe 90) – completely unmanned for the majority of their voyage. They then join a ship which happens to be traveling around the southern tip of Africa, and then towards the southern pole due to a shift in intentions. In the final few chapters, the last leg of the journey at the lowest latitudes is a life-or-death race effected by the native people.

This reminds me of various accounts from the scientists in Herzog's *Encounters at the End of of the World*, who mention that Antarctica is a place where eccentric people converge. I paraphrase here, due to not revisiting the transcript of the movie: they describe a convergence of people who want to jump off the margin of the map, and people who are not tied down, and thus fall down to the bottom of the Earth. Is Poe trying to imply something similar – that we are naturally drawn (if we survive a series of miracles) to the pole(s)? That what seems to be repeated misfortune is really more intentional, or at least less than random chance?

Poe's lack of emphasis on snow and the coldness of the Antarctic is notable. This may simply be an overlooked inaccuracy, but it feels out of place when Poe goes into so much detail about other scientific accuracies (e.g., the Galapagos tortoises, the filling of ships with grain, the biche de mer). Even if the Antarctic is visited during the summer (February), it may still be below freezing temperatures, far too cold for people not originally equipped to visit the poles. Moreover, while the closing Note chapter suggests that the cries of "Tekeli-li!" by the Tsalalians is made in response to whiteness, it is unclear how they are not continually averse to the whiteness of the snow that perpetually surrounds them.

I also wonder about the influence of other literature on *The Narrative of Arthur Gordon Pym*, and its on other literature. The first half of the book, which is a tale of unlikely survival on a shipwreck, is highly reminiscent of the 2001 novel *The Life of Pi*, containing similar themes such as cannibalism, drinking turtle blood, false illusion, and delirium. We have already learned that Symmes is a contemporary of Poe, and the dramatic conclusion of the book is highly influenced by the speculation of some dramatic disruption of the pattern of increasing cold in the more extreme latitudes. When reading the final few pages, the description put the image of the end of the world somehow reminded me of the misty opening to the "Hidden World" in *How to Train Your Dragon 3*, which may very well be a Symmesianism reference. In the image of that entrance (shown below), we have many of the same visual elements: the mist that obscures the mysterious bottom, the cataract opening into a chasm in the middle of a warm ocean, the darkness of the ocean and rocks as compared to the white mist.



The entrance to the Hidden World from *How to Train Your Dragon 3*. Image courtesy of https://howtotrainyourdragon.fandom.com/wiki/Hidden_World.

Also: a general commentary on the book (not pole-related)

I think the book is much less convincing by the fact that there is a Preface and ending Notes chapter that attest to the veracity of the events in the book, and by the awfully convenient rescues of the narrator (and of his similarly-fortunate crewmate Peters). The other aspect that confuses me is the bifurcation of the story into two misfortune-bound ship voyages: the first by the *Grampus*, and the second by the *Jane*. It is only really the latter voyage that concerns the Antarctic and thus this course; the first half of the book seems to be entirely disjoint and provides little to the latter half.

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

9 / 18 / 21

The impracticality of exploration, and the class of the polar celebrity writer

Response to Spufford's *I May Be Some Time: Ice and the Imagination*

While originally under the pretense of searching for the fabled North-West Passage, an exploratory expedition of the North becomes more a status symbol than an attempt to provide any practical advancement to society. Those that explore are glorified by those that come before them (both those that do and don't return successfully), forming a framework for climbing the social ladder: their return is marked by treasure and hysterical celebration (Spufford 56), their writings are virtually guaranteed publication (Spufford 52), and they get access to the finer gardens of society (Spufford 52).

Contrast this with the whalers by profession, who in some sense condescend to the explorers. While the whalers may also have a stake in the wealth and recognition, they had at least the grounding of an occupation. Of the men who were purely explorers, "most of them knew nothing about polar exploration when they set out to do it. The English were uniquely unprepared for the job" (Spufford 5). Lopez also acknowledges this, as the whaling crews generally "viewed the British discovery expeditions … as a pompous exercise in state politics, of little or no practical value" (Lopez 10). The simple fact is that even if its existence of the North-West Passage is proven, the intermediate icelands would be impassable anyways, which invalidates a claim of commercial benefit. To the working whalers, pure exploration is the prerogative of those who can afford the risk of adventure.

The explorers who return have an influential role akin to famous writers or poets. Spufford lays out several written works written by these famous explorers, including playful verses from Parry (Spufford 51) and Coleridge (Spufford 53). It's not a surprise that *Frankenstein*'s Captain Walton was a poet before he turned to exploration. These explorers "could expect to see their narratives accepted by

prestigious publishers [and] there was regular coverage of expeditions in serious reviews" (Spufford 52), which is part of the aforementioned framework that allows the explorers to spread their recognition with a positive feedback loop. In addition to having a platform to spread their narratives, Spufford also describes a certain characteristic of the writings that make them so popular. Bewick's *History of British Birds* exemplifies this: despite being a children's book about birds, it is able to capture the awe of many British people and become a classic (even making its way into *Jane Eyre*). While it doesn't fall into the genre of fiction (nor do the accounts of the other explorers), it is a suggestive tone mixed with the readers' imagination that pushed it into the realm of the imaginary – "from being the language of pious geography, albeit heightened and intensified, Bewick's words *become* here the language of romance and fantasy" (Spufford 13). As we already know from our previous readings, the "hype" of polar exploration is carried on by the narratives of explorers, but here we see that even a descriptive work of ornithological writing is sufficient to stimulate the imagination.

While much of the polar curiosity formed by this emerging body of literature is evocative and harmless, it arguably hurts the science. It is difficult to contest the words of explorers, and the scientific method is (apparently) not well understood. Spufford brings about the prominent example of Symmes and his theory of an "inner world" and "layers" within the earth with a polar entrance. Of course, this is crackpot science with no regard to logical methods (e.g., the misunderstanding that the lack of a counterexample is not equivalent to a proof), but it was able to gain the attention of a body of people and some votes in Congress to pursue his speculation. It is a conspiracy theory founded on the imprecise descriptions by explorers (and unrefuted due to lack of scientific evidence).

Lastly, I find the mention of a rebuttals against the Arctic fever a welcome relief. This is a logical skepticism not encountered in the earlier readings. We have the mockery of the hysterical return celebrations by Cruikshank (Spufford 56); Smith's logic that "if you can revere a geographical concept, you must also be able to slander one" (Spufford 55); and the *Symzonia* in response to Symmes (Spufford 72). The disdain of the whalers is another form of healthy doubt.

Jonathan Lam

Prof. Germano

HUM324 – Polar Imagination

11 / 01 / 2021

Response to *Land of Wondrous Cold*

I was reading through our classmates' responses, and the idea of the "narrative" of the explorations and of climate change, as opposed to some sort of bland history or jumbled collection of events, is reflected in several of the responses. Tony, for example, mentions the reappearance of the Erebus and Terror. I was confused when initially reading this, and upon a quick search, it turns out that it was indeed the same ship as in the doomed Franklin expedition. (Aside: I am not sure why we haven't seen earlier mention of the Erebus and Terror and their previous Antarctic expeditions. When confirming that the two ships were the same ones used in the later Franklin expedition, I wasn't sure if I had found the correct one, as both ships were originally warships as mentioned later in the book. Wikipedia also lists five ships named HMS Erebus and nine ships named HMS Terror, adding to the confusion. There are also volcanoes named Erebus and Terror on the Antarctic continent.) This seems not too surprising, as the world of famous Arctic explorers and Arctic exploration ships is small: for example, Crozier was a member of the Ross Antarctic expeditions and also second-in-command on the Franklin expedition, and Ross (both James and John) were sent to search for the lost Franklin Expedition. As Tony also mentions in his response, this gives the ship a character, a history. Without this information when reading the previous accounts of the Franklin expedition, my view of the ships were that they were ordinary exploration ships, perhaps something like whaling ships, ill-prepared for the ice. Armed with this knowledge, we now know that the Franklin explorers had the assurance of two of the finest exploration-purposed ships in the world, with the confidence of surviving a longer- and similarly-ice-laden trip to the Antarctic. I can imagine that this may have given the Franklin expeditioners too much confidence in their ship to think about learning the ways of the Inuit people for

survival, in the same way that the Titanic was “unsinkable.” This extra knowledge also more strongly contrasts with the dinky little boat and six-man crew led by Amundsen in 1906 that was the first to actually traverse the North-West Passage.

The idea of the nonfictional narrative is also mentioned in Sanjana’s response, in which she mentions D’Urville’s wife Adelie, and the similarity to the tragedy of Sir and Lady Franklin. I too noticed this, and thought this to be the first instance of romance encountered in our readings thus far; Lady Franklin’s is also the story of a tragic love, but it is not presented to be romantic in the way that D’Urville and Adelie kept a secret promise to each other. This gives rise to his study of the natural sciences to prevent an onset of madness – without the description of the romance, I believe that the latter fact would be much less potent.

The piece of the narrative that I found to be most capturing was the moments when the discovery capture the botanist, Hooker. Cindy also mentions the enrapturing image of the sheltered flower that Hooker finds. However, I find equally if not more fascinating the moments when the seeds of continental drift theory begin to materialize in his brain, when “the world of Kerguelen opened before him like a book” (Wood 51), also on the same island. This is the tip of the iceberg into a much deeper theory – literally on a global scale – which is significant in the theory of global climate change patterns (as Wood explains through much of the book). Similar moments have have also presented themselves to Hooker when he notices the likeness of plant species on spatially distant bodies of land, which is the foundational moment for the theory of continental drift, and (via association with Darwin) also important evidence for the theory of evolution.

Looking at these three pieces in the narrative – the background on the Terror and Erebus, D’Urville’s romance, and the origin of continental drift – none are strictly necessary to tell the narrative of Arctic exploration, or of climate change. The previous accounts have gotten away with omitting these details, instead stating statistics about the water level change if all the glaciers were to melt, and of the dates and writings and skeletons left behind by the Franklin expedition. Providing these details

of what the characters knew or saw, rather than simply the aftermath of their expeditions, makes for a much more tantalizing narrative. By collecting and connecting these narratives, we can make better inferences about what the explorers felt or were motivated by, and we can even get closer to reliving their moments of joy or discovery.

01/01/2022

Notes on the revision of an essay on life and death for the Inuit People in the Arctic;

To Prof. Germano:

I agree with your feedback. I have become an expository rather than an argumentative person, and usually I explain things in a way that conveys my excitement more than the point. This is undoubtedly the source of the fuzzy direction and an overly conversational tone in the paper.

Pruning is the first obvious solution to this, but it is often not so obvious whether a particular paragraph is or is not ultimately useful to the reader in understanding the final point by illustrating some new aspect. In other words, it is difficult to determine whether the information contained is something that I should be telling the reader or something they should be figuring out themselves (something I have also found trouble with when tutoring computer science). I removed the section on infanticide and canicide, trivial footnotes, and filler transitions that existed for the sake of transition. I left much of the other information intact, figuring that much of the ostensibly-cursory information is ripe for context and interpretation; however, I believe that a finer pruning (i.e., further pruning with little content loss) can be performed with much more effort.

There was no reordering – one of the things that I believe was very well done in the first draft was the logical buildup of context. I felt no improvement due to any reorderings I attempted.

There was a thorough rewording due to general style issues. Many passive wording, past tense, overly conversational tone, misspellings and grammatical issues, etc. were cleaned up. This is especially true in the exposition and conclusion of the essay. I guess this cleanup should be a given of any revision.

These changes so far help remove clutter but don't actively promote an argument. I had better refined my argument between the previous draft and the time of my presentation. The conclusion to the previous draft was more or less along the lines of "now, you've seen what was and what is now. Consider it all and become more compassionate." I've taken it to the slightly-less-abstract, based on your comment about the cheapening of death and the functional view of belief systems that I emphasized during the in-class presentation: what does it mean to redefine death? It's still very open-ended but much more focused.

Lastly, apologies for the tardiness. I've been stressing over general indecision and confusion about my Master's, so this has been a somewhat lower priority until I could get my head straighter.

Happy new year! (Fingers crossed that we don't start this semester online!)
Jonathan Lam

Jonathan Lam
Prof. Germano
HUM324 – Polar Imagination
Jan 1, 2022

Shifts in the suicidal imagination of Arctic native peoples

We began our studies of the polar regions with Frankenstein's creature, leading Victor Frankenstein in a malicious game of cat-and-mouse to the Arctic. Victor's desperation and emaciation are increased by the wear and tear of the climate, and he perishes. His purpose defeated, the monster too ends his life in a spectacular and almost ritualistic manner (by pyre) at the North Pole.

While *Frankenstein* is a work of fiction, with polar elements added apparently for dramatic effect, death is a pervasive theme in the Arctic and Antarctic regions. The cruel Arctic environment is uninhabitable for the majority of civilization, leaving an untouched wilderness whose image early explorers have romanticized. The poles are elusive, and, naturally, major national and colonial powers may wish to stake their claim in it for national pride and economic trade routes, expending much effort (and accruing many casualties) during the golden age of polar exploration in the mid-nineteenth century. It can be argued that the closeness of death (by natural, extreme means) gives the polar regions its allure – to cosmopolitan and tropically-situated outsiders, at least.

The same line of reasoning would not make sense for the scarce native population of the Arctic. This broadly includes people who live above the Arctic circle, who for millennia lived without support of resources from the temperate regions. Examples of people of this region include northern Canada, Alaska, Greenland (collectively, the *Inuit* peoples), and Siberia (the *Yupik* peoples). The tantalizing closeness of death and beauty of the blank landscape for those in temperate regions with fanciful imaginations becomes the weariness of daily life, in a similar way to the drone of a daily corporate life. Yet even with millennia of adaptation, being "well-suited" for the Arctic up until the industrialized era was a treacherous lifestyle of accidents and natural disasters. Even after the colonization of the northern regions by industrial powers, including modern housing and healthcare facilities, the death rate is still alarmingly high – much of which has to do with factors outside of the environment.

This essay will focus on the ways in which the Inuit and Yupik peoples conceptualize death, deal with the elevated risk of death in an oppressive climate, and how colonialism has shifted these traditional views[1]. It will be a synthesis of a number of anthropological studies conducted in the mid-to-late twentieth-century on the topics of traditions of death and patterns of death among the Inuit. The interest in this apparently morbid topic stems from the idea of a resignation-to-death by Inuit elders – a stoic suicide – as recorded in Kpomassie's encounters in Greenland; a stoic image that has captured the imagination of many regarding the nature of the native people. This is only one of a number of "death rituals" that appears to be common across the native peoples of the North (but not common in

1 A word of caution: The Kpomassie story, and the many subsequent stories and interpretations from second-hand accounts in anthropological studies, have gained the author's interest in this anthropological aspect. However, the author is inexperienced in this sort of examination, and is prone to errors such as overgeneralizing results that may be particular to a specific Arctic native people to all Arctic native peoples – this fallacy is known and noted in much of the research in this subject matter (Andersen and Poppel, 2002). Similarly, a topic with as much social and psychological implication as suicide is breached by the author for the first time. The individual effects of a suicide on a person may be grossly understated for the purposes of this paper, which focuses more on the communal effect.

frequency, and ever becoming more rare, and thus somewhat cloaked in mystery) that piqued my interest in the subject. The imagery of Frankenstein's monster's resolution to suicide picks up a shade of reality.

AN OUTSIDER'S PERCEPTION OF THE INUIT, AND SENICIDE[2]

We can learn a lot about the nature (and allure) of mortality in the Arctic through the adventure of a single man in *An African in Greenland*, as Tété-Michel Kpomassie tells the story of his lone journey to discover what he imagines to be an incredibly benevolent[3] people, the Inuit of Greenland. His curiosity stems from a library book, and his drive stems from an aversity to the strict customs of his tribe in Togo. From the pictures in the book he sees "these men plump and smiling. Strange clothes made of animal skins covered their bodies, and all you could see under their big hoods fringed with thick animal fur were their happy, open, honest faces" (Kpomassie, p. 46). They are happy, and the children are free-willed, despite living in a desolate land without trees and by eating raw meat. This is in contrast to his own circumstances, where, despite the abundance of the land and relative ease of everyday survival, he is unhappy because strict cultural rules would force him to become a priest against his wishes. The question that he seeks to answer is, *How can people live with so little and be so happy?* or, conversely, *Why does our life of abundance feel so miserably constrained?*

Kpomassie moves slowly upwards through Greenland, moving at a rate constrained by the climate that allows him not only to observe, but also to be changed by the people around him. By the end of his journey, the native people see him as one of them, a true Greenlander, of the same personality that he had sought in the first place. He quickly discovers the carefree, good-natured personality of the southern Greenlanders at K'akortoq – so carefree that the lack of apparent purpose in their actions perplexes him. The seemingly senseless polygamy, the ever-refilling coffee and house visits, the unlocked doors at night, the rampant alcoholism, the unemployment and the lack of urgency cause him to wonder where the "true Eskimos" – the ones he had read about in the book – have gone. These people have that good nature and lack of strictness he was looking for, but they are fed by Danish governmental subsidies rather than hunted seals.

While at K'akortoq, Kpomassie visits the senior people's home and a group of very lively elders receives him. This is the first encounter with traditional Inuit culture – and yet he sadly notes that they are quarantined away rather than being united with their families. As if the traditional culture were being locked away in a modern, safe box – to keep them safe, or perhaps to keep the youth safe from them.

Kpomassie then tells of the foreign idea of suicide among the elderly, speculating that these elderly are contained in the senior home to prevent them from committing suicide. We get a typical ancient version of the story:

> "In the old days, both in Greenland and among the other Eskimos, the old people, so as not to encumber a migration, would elect to remain behind and die slowly in the abandoned igloos. It

2 The killing of elderly. Alternatively, senilicide.

3 I attempt to summarize the general fascination of the Inuit people in a single word. Kpomassie doesn't use the word "benevolent"; other attempts at summarizing this feeling may be "gentle" or "good-natured."

> was a spontaneous, stoic, unforced decision, and one which to them seemed noble" (Kpomassie, p. 101).

as well as a typical modern version of the story:

> "Today, the old sometimes commit suicide. An old man may be driven to such an extremity when he is *gamapok*, angry. Angry with himself. He goes out and never comes back. This happens particularly in winter: he leaves the house and walks a long way out on the frozen sea without heed for the places where the ice is soft, then all of a sudden – just as he had hoped – he sinks and is swallowed up. Sometimes he tells his family, and they do nothing to stop him. The old man has made up his mind and will not budge! Those who kill themselves in this way have often been great hunters. Diminished by old age and feeling themselves a burden to everyone, they don't take easily to their changed condition" (Kpomassie, p. 101).

While Kpomassie seems to have little reaction to such a story, only mentioning it in passing as a possible reason as to why the elders were in the senior home rather than living with their family, this captures my imagination as a tragedy.

In the case of the former: in a life-and-death scenario, the young are more valuable than the old for the simple logic of procreation and physical strength. But the elderly are an accumulation of knowledge, providing long-term stability and wisdom – losing them may be just as valuable. Moreover, it creates a certainty of slow death, which requires immense determination on behalf of both the abandoners and the abandoned; once the choice is made, there is no turning back, and only a death by starvation or cold awaits. It brings about a very raw emotion: sacrifice for the greater good as the ultimate means of survival[4]. This act is heroic, and this scene of Inuit senicide has captured the public imagination.

However, the latter case seems to be a case of heroics rather than heroism, with pride is at stake rather than survival. In his stubbornness, does the old man view himself as in the former case, his death serving to prolong the preservation of the living? Even if not the case, both stories have the aspect of a *violent death*[5] against oneself, and especially a violent death by the harshness of winter.

As Kpomassie continues north from K'akortoq, the environment becomes increasingly hostile, and death ever more present. Besides the obvious modes of death by freezing or drowning (or freezing after a dip in the water), our narrator experiences second-hand: drunk parents falling on and suffocating their baby, babies devoured by dogs, puppies devoured by dogs, a middle-aged man devoured by dogs, an execution of dogs suspected of manslaughter (dogs feature quite prominently in this matter of death), able-bodied men falling through thin ice and drowning, a carelessly drunk murder due to a relationship dispute, an accidental gunshot through the leg in a canoe, as well as many near-death scenarios.

---

4 Another powerful instance of this story, one necessitated by relatively modern times, is that of the cleanup effort following the Fukushima nuclear reactor meltdown incident in Japan: a crew of retired, elderly Japanese citizens volunteered themselves for the cleanup effort to save the younger generation, facing a danger that we know no protection against.

5 The term "violent death" is used loosely; the National Violent Death Reporting System (NVDRS) defines violent death to be "a death that results from the intentional use of physical force or power, threatened or actual, against oneself, another person, or a group or community" (Centers for Disease Control and Prevention). I use it here to mean the complement of non-violent deaths, i.e., a death not by disease or old age, including accidental deaths and deaths by harsh environmental conditions, and especially referring to deaths by intent. "Violent death" is used in a similar way in the literature, such as (Pika 1993).

Kpomassie appears to show little empathy for these deaths; this increased indifference to seeing the dead is a reasonable, or at least understandable, reaction to the increased rates of death. In her discussion of suicide rates among the Inuit, Stevenson notes that "the truth of the yearly suicide rates makes their death inevitable. Inuit are tentatively imagined at the walking dead, and yet even that thought is proscribed: It is a 'horrible thought'" (Stevenson et al. 2012) – replace the "yearly suicide rates" with the "general observed mortality rate"[6] to match Kpomassie's diverse observances of death. Luckily for Kpomassie, the stories he hears appear to largely be secondhand, which increases his distance from the dead; for seeing firsthand these deaths may either more rapidly increase one's tolerance to the atrocities of death, or may cause a traumatic aversion to it.

Although Kpomassie has a background that is very different from a metropolitan reader such as myself, he does have the perspective of an outsider. The inferences that are made from these observations, without a deep understanding of the culture, undeniably have deeper roots. Keeping this in mind, we may attempt to draw some conclusions. Firstly, there must be a hardiness or indifference or other form of emotional protection in the Inuit people in order to persist among high rates of mortality. Thus, we may be overly reductive and partially answer the titular question by claiming that the significance of death to the Inuit is simply "less." We can also be overly reductive and say that the great perceived benevolence[7] of the Inuit is precisely this fortitude: the ability to stay positive in the face of great natural danger. While these statements may be partially true, they are only in the vacuous sense: inasmuch as they are oversimplified and ambiguous.

In summary, Kpomassie's accounts include a wide variety of methods of death and touch upon many of the major themes explored in the literature about mortality of the native people of the Arctic, including (but not limited to): senicide, alcoholism, romance and family, and death of dogs.

MULTIPLE INTERPRETATIONS OF (LIFE AFTER) DEATH

To improve our understanding of the meaning of death for the Inuit people, it is instructive to take a deeper dive into the traditions and beliefs surrounding death.

Many similarities can be drawn between the beliefs of different Inuit peoples. Walsh and O'Neill summarize the beliefs of a number of North American Inuit groups (Walsh and O'Neil 2018). Generally, the physical ritual is to cover the body and abandon it outside[8], and there is a brief period of mourning. A common trait is the breaking ("killing") of some of the person's possessions, so that they can take those items with them to the next life (Mackenzie Delta Inuit, Copper Inuit, Netsilik Inuit, and Caribou Inuit). Walsh and O'Neil describe the crude burial and the short mourning process mainly to be the result of two factors: a matter of pragmatism, and the fact that all of the Inuit groups believed in some kind of life after death. The exact nature of the realm of the dead varies – the Baffinland Inuit, Iglulik Inuit, believe in a series of underworlds and an "overworld" (the Baffinland Inuit called this *Qudlivun,* a realm in the sky)[9]. The Iglulik believe that certain classes of death may route to different

6 This is not an unfair substitution to make, as we will soon see. The general mortality rate of the Inuit people, as well as the rate of suicide, are both high. The latter is perhaps the more shocking, however.

7 See Footnote 3.

8 A practical consideration, as as burial is difficult in frozen ground.

9 One should be careful not to inject the connotations with Christian beliefs of Heaven and Hell with these terms. Logistically, the Inuit people probably had little to no interaction with Christian beliefs, so it would be incorrect to tie

destinations – death by childbirth, violent death, or suicide would admit entrance to the sky realm. The Netsilik Inuit believe in the existence of three realms: a joyful sky realm for those with a violent death and for hunters, *Agneriartarfik*; a joyful underworld, *Aglermiut*; and another underworld, not as joyful.

Alongside the belief of an afterlife underground or in the sky, many Inuit also believe in the idea of a recycling of souls, a reincarnation – we will focus on this interpretation. The moment one dies, they awaken as a newborn, either in this world or a mirror one. The exact reconciliation of these two beliefs is somewhat unclear; Walsh and O'Neil summarize the phenomenon of the soul as "one form traveling to the underworld and another potentially being recycled back into the world of the living" (Walsh and O'Neil 2018), implying some uncertainty on their part as well. The fact remains that, with afterworld or reincarnation, death is not an ending.

The other resounding theme is the apprehension of the dead causing harm to the living. This may involve some superstitions such as not leaving items with the body that can cause harm (Caribou Inuit), or orienting the body facing east (Netsilik Inuit). The harm being done to the living is not taken to be a form of bitterness or ill-will; it is taken to be the "perceived opposition between anything associated with the dead and the health and the living" (Walsh and O'Neil 2018), as if the dead and the living were naturally averse to one another as an axiom of the universe.

Willerslev corroborates many of these traditions as he studies the Chukchi people of Siberia[10]. He illustrates the beliefs with an analogy:

> "The Chukchi cosmos can perhaps best be described as a hall-of-mirrors world: Each thing is paired with almost endless doubles of itself, which extend in all directions and continually reflect and echo one another. For example, the much feared evil spirits, the *ke'let* … are said to live in camps and villages, travel about the country on sledges, and go hunting for prey as do human beings. The game they hunt, however, is the souls of men, which they call their 'little seals' or 'reindeer.' From the viewpoint of a human being, the ke'let have monstrous and terrifying features, such as hanging eyes, half-formed bodies, and large mouths full of teeth. Yet, from the viewpoint of the ke'let themselves, they are the ones who are human, and they regard the human shamans who can attack and kill them as ke'let – that is, as evil spirits" (Willerslev, 2009).

The "hall-of-mirrors world" metaphor is introduced with the example of the ke'let evil spirits but also is true of the dead. For the Chukchi, there is a combination of the afterworld and reincarnation in a sort of "mirror realm" – when one dies, they are immediately reborn into a realm both alike and reversed from ours. The inhabitants of that mirror realm have their souls and bodies flipped inside out, and they see our realm of living as a mirror realm of the dead, one that they will reincarnate to when they die. It is unclear whether these inhabitants of the mirror realm are exactly the ke'let or if they are a different form, but they share the aspect that the dead are the "other" in the perspective of the living, and vice versa. There is a symbiosis: what we see to be reindeer or seals may be their people, and the living

---

them to the same origin. Also, the cyclical nature of Inuit souls differs from the Christian belief, in which Heaven and Hell are eternal resting places.

10 Walsh and O'Neil also note the relation of their work to Willerslev's notions about Siberian peoples. Conversely, Willerslev is one of the editors of the book in which Walsh and O'Neil's paper is published. Willerslev's observation of the Chukchi's cultural thoughts on death stem from the same purpose as mine: to understand better what deeper meaning there might be to voluntary death.

must hunt to survive; and, like people, the dead game are instantly reborn in the other world. This is the "perceived opposition" between life and death mentioned in Walsh and O'Neil – a necessary and constant exchange between the life and the dead, not out of ill-will, but simply because it is a natural order like that of hunter and prey.

The realm of the dead (i.e., the mirror world) is believed to have power over life in the living realm, and this is true from the perspective of both realms. This is because the number of souls is conserved in this system; by controlling how many are born or die in one realm, the other realm is affected inversely. (This interpretation provides a logical (if not provable) mechanism for which sacrifice benefits the living.) Thus there is a constant deference to the other side, a sort of respectful and ritual kowtow of reciprocation. As we have seen earlier, there is also the ritual "killing" of possessions, both alive (in the form of sacrificing livestock) and non-living (by breaking such possessions), as these possessions are to accompany their owner in the next life. These sacrifices can be seen as a way to appease the other side (sacrifice "is made not only to the gods but against the gods" (Evans-Prichard 1954)), so that they don't exert their power.

Thus death carries a functional role within this worldview, by recycling old souls into new souls and keeping the system in equilibrium. If, hypothetically, the people in one realm were to live forever, then the other world would be starved and the young will cease to be born, whose result will be cataclysmic on both realms. In other words, the belief is that one is linked with an realm that struggles to survive and requires death in your world in order for it to flourish; and the inhabitants of that realm also sentient and aware of the reciprocal effect of our realm on it. Moreover, we are stuck in an inextricable and transparent contract with the mirror realm – transparent in the same way that scientists are unsure of how quantum entanglement binds two entangled particles, the state of one particle affects the other, even at great or indeterminate distances[11].

The importance of sacrifices cannot be understated. In order to maintain the balance, there must be a healthy flow from the living to the dead, and vice versa. Just as humans create tools to ease various aspects of their lives, sacrifices are a tool that humans devised to have some control over the life-death cycle, seen as a way to prevent others from dying. As Willerslev notes, livestock (usually reindeer) may be sacrificed for a human. If the living are poor on reindeer, then a surrogate may be used; typically, a *zyozyat* (reindeer sausage) or even a wooden image of the zyozyat may be ritually sacrificed using by stabbing. Of course, the greater the sacrifice, the more likely the ones in the other realm will be appeased; the surrogates may only act as something like artificial sweeteners are to sugar: symbolizing the attempt at providing sugar, but without the original benefit of providing energy. The dead can recognize this sacrifice as a request for postponement but will not allow it infinitely.

Willerslev claims there is a fundamental difference between "ordinary" suicide among the Chukchi people and the ritual suicide that is voluntary death. The latter is ritualized and usually involves advance notice, whereas the former is typically more sporadic. He then reasons that the latter cannot be primarily motivated by the a matter of efficiency, for two reasons: ritual suicide is still carried out today

11 Note that quantum entanglement can be broken the moment one of the entangled particles interacts with the environment, as suggested in a Quora thread "How can an entangled particle pair be 'disentangled'?" (Kattel 2019). One such form of interaction is measurement (as in the famous metaphor of Schrödinger's cat). We can actually use this to further our analogy with the mirror world: one may argue that the moment that one is able to truly observe the mirror world, then the entanglement would collapse, possibly throwing reality into an unstable state. The same can be applied to many religious phenomena that can only be felt but not proven (scientifically), and thus lie outside the realm of science except in its (reportedly) perceived effects.

when the Russian government will provide necessary living aids; and ethnographic records have shown that ritual suicide occurred not only for the elderly, but also for able-bodied youth (mostly those in distress). A possible justification for this logical gap lies in the mirror world: by offering oneself up as a sacrifice – the ultimate sacrifice, when compared to other sacrifices, which act only as inferior surrogates – the long-term stability of the two realms is protected.

Sacrifices for the sake of survival are not limited to the elderly, as there have been some instances of infanticide and canicide in Inuit cultures as well (although much rarer than senicide). Infanticide, like senicide, prunes off weaker members and aids the stability of the group (Schrire and Steiger 1974). However, it may not be purely a matter of efficiency; the same argument can be made as in the case of senicide, that this has the secondary benefit of returning souls to the mirror realm. Ritual canicide relates directly to this theory: while animals have been sacrificed as a means of curing the dead in other cultures, the mirror world provides a semi-concrete mechanism for this cure.

If we return to Kpomassie's stories, it is now a matter of deciding whether the second, modern story is that of a ritual or regular suicide. While it is not very ritualistic, it does not fall into the patterns of the aggravated suicide of the youth, for example in its means (walking off into the snow) and is told by Kpomassie to be a story of the elderly only, and typically of the hunters, who participate most in this life-death exchange. It would not be too far off to say that, albeit hidden by their stubbornness, these angry elderly truly believe that they are helping the young in their fatal decision.

COLONIAL HEALTHCARE AND TEENAGE SUICIDE: THE MODERN CONUNDRUM

It is no surprise that life in the Arctic regions is still treacherous, but certain aspects of life have been made more difficult after the subsumption of Arctic lands by industrialized southern nations, beginning in the second half of the twentieth century. In particular: Alaska became a U.S. colony, the Arctic regions of North America came under Canadian control, Greenland under Danish control, and Siberia under Russian control. This has some benefits for the local peoples, such as governmental aid for basic survival needs, but the negative effects arguably outweight the positive ones.

We can begin our survey into modern mortality with the modes of death that have been invariant since ancient times. The rate of accidental death (Day and Lanier 2003) and infant mortality (Aaen-Larsen et al. 2003) will likely always remain relatively high, given the harsh environment. Similarly, the psychological effects of the year-long day, which Kpomassie encounters in the form of a Greenlander becoming homicidal and amnesic, and which have caused noticeable seasonal patterns of violent death (Pika 1993), will not change as long as the Earth continues on its orbit and humans can sense light.

Some of the important objective sources of change directly related to globalization include climate change, government policies, resource development (such as oil drilling), animal rights groups (which may interfere with traditional hunting efforts), contaminants[12], increased cultural exposure with the south, imposition of religion (especially Christianity), and imposition of new education systems (Anderson and Poppel 2002). A combination of these effects has led to a health crisis among the Inuit

12 The contamination even includes the radioactive type: a U.S. military plane crashed into Greenland with hydrogen bomb nuclear warheads onboard in 1968, leading to genetic deformities in wildlife and a spike in cancer rates in the surrounding areas. (Ehrlich 2002).

people: an increase in access to unhealthy foods and changes to the environment caused a marked change in the lifestyle. Anderson and Poppel state that between 1945 and 1996, the proportion of the population engaged in hunting and fishing had decreased from 66 percent to 25 percent. This was accompanied with rising trends of crime and drug abuse, and a general feeling of hopelessness in the face of the ever-present global market economy. If we consider the changes in mortality rates of Alaska natives between 1979 and 1998, we find that the major causes of death pivot from being concentrated in the infectious diseases to primarily heart disease and cancer (Day and Lanier 2003), a result of a more unhealthy lifestyle made easier by globalization with lower intake of the local diet of fresh meat (Tynes and Haldorsen 2007). According to Day and Lanier, unintentional mortality in Alaska occurs at a rate 3.9 times that of whites in the U.S.; suicide is the fourth leading cause of death (7.5% of all deaths), 4.2 times the rate of whites in the U.S.; and homicide is at a rate 3.3 times that of whites in the U.S. Alcoholism has also become a systemic problem among the Inuit people; the mortality rate due to alcoholism in Alaska was fourteen times the national average in 1968 (Fleshman 1972), and this may be closely related to suicide and accidental death rates.

With this context, the suicide pandemic among the youth is not surprising. Adding to the list of large-scale lifestyle changes mentioned above, there are a number of more subtle effects that further isolate the youth. An article aptly titled “The weight on our shoulders is too much, and we are falling” describes several of these effects, such as: the unskeptical absorption of Western media containing unrealistic romantic expectations and an entirely different system of idealist, monogamous “voluntarist individualism” than the traditional arranged marriage system; a general racism and poverty; and segregation of the youth from elderly via the education system (Kral 2013). This has led to the widespread feeling of isolation between the youth and their family or romantic partners.

With so many social pressures, one might hope that increased access to nationalized healthcare after colonialization would mitigate the suicide crisis. However, the system is not effective nor understanding. There are long separations from family (both in space and time), and there are several stories of losing contact with elderly family members, only to discover that they had been unceremoniously died on the journey or during the treatment (Stevenson 2012). Moreover, the methods for treating suicidal patients, as conducted via a survey of healthcare workers in the Arctic, tend to be procedural rather than personal (Trout and Wexler 2020). The healthcare system appears to cause more to scramble away from it than to seek it out; and the patients that go through this system may suffer additional mental strains from cultural separation.

Stevenson examines the Canadian healthcare for the Inuit, and reaches the same conclusion as Trout and Wexler that the approach to care is not appropriate for treating the youth. She describes the perspective of legislators on suicide as “at once prohibited (*life is sacred, thou shall not kill thyself*) and awaited (*but, of course, we know you will*)” (Stevenson 2012). For the purposes of the law, death should be prohibited, and all of the standard necessary actions should be put into place to aid the suicidal people; when considering whether “whether it would be better to have a ‘dead Eskimo’ or a ‘disturbed,’ living Eskimo,” it is easy to always prefer the latter, and much more difficult to consider each case and its effect on the community. Stevenson asserts that this is the curse of stereotypical modern medicine: it is a biopolitical plan to keep as many people alive as possible, interfacing with patients using the face of anonymous care. As with much of politics, the purported benefits lie far from the perceived effects.

Needless to say, it feels as though everything is wrong. *The weight on our shoulders is too much, and we are falling*.

In their desperate isolation, the morbid trend of suicide may even feel as a unifying thread, as illustrated in the common feeling:

> "Some Inuit described seeing their dead friends visit them, usually at night, asking to join them in death … Suicide for some has become a shared response to distress, and … a way of belonging and identifying with similar others" (Kral 2013).

The motif of having the dead come to visit is not unheard of in popular culture, but it is also reciprocated in the traditional mirror world analogy: while the realms of the living and dead are usually kept separate, the inhabitants of the mirror realm may be able to enter the current world and look for souls to take when certain taboos are broken: "suicide is when the dead rope the living in" (Willerslev 2009). One may imagine that, amid high rates of suicide, the pressure from the many in the mirror realm is too much to resist.

Despite being a point of unification, this youth suicide is by no means accepted with pride. It is the opposite of the ritualized voluntary death of the elders.

> "This trend (of rising suicide rates) is deplored by the Alaska Natives I have encountered. The high rate of suicide does not imply a positive social sanction. Many of the patients expressed a feeling of shame and feared nonacceptance in their village following their shooting … [Historically, voluntary death of the elderly] was undertaken after reflection and sometimes consultation with family members, who might assist in the final act. This form of suicide was positively sanctioned and had a cohesive effect upon the community" (Kost-Grant 1983).

Indeed, it appears as though the meaning of self-death has changed from that of voluntary death. While it may be a way for the youth to escape and unify with the dead, it has a highly negative effect on the community, and sometimes this may be its primary purpose:

> "In fact, there is a general saying among the Chukchi that 'young people kill themselves to harm their kin,' meaning that suicide among youngsters often has the aim of hurting the feelings of others, most notably parents or lovers" (Willerslev 2009).

Why has the intent changed? One factor that may have contributed is that the ritualized death (when carried out by a family member) is outlawed as homicide (e.g., in Russia) or otherwise antithetical to a public health system (as in Stevenson's examination), and is thus much rarer (for fear of prosecution). The youth, then, may be more influenced by negative connotations with suicide from outside sources rather than local traditions. Voluntary death having thus been outlawed, suicide moves from the ritual space to the self-initiative. This is a lens into Willerslev's categorical separation of suicide and voluntary self-death, in particular the sporadic and non-ritualized aspect of the former. Other studies have also taken to this view, suggesting that ritualization also has the effect of regularizing the behavior, and this deregularization combined with the devastation of traditional cultures has caused "a situation of overwhelming traumatic self-annihilating behavior" (Bogoyavlensky and Volshonsky 1997).

In a sense, we can say that national health system such as the Canadian one, in their attempt to conserve and control life, has the contradictory effect of preventing controlled manners of death and forcing unsanctioned, unregulated death.

The concept of control is a recurring theme in Stevenson's work, and ties back to the belief of the mirror world. Recall that there is the belief that the realm of the dead has power over life in the realm of the living. Conversely, the land of the living has some power over the land of the dead, by virtue of the fact that the number of souls are limited. By means of sacrifices, the land of the living can expedite the process of recycling souls, and keep the mirror worlds in a healthy equilibrium. But a blanket restriction on preventable death via government policy is a wrench thrown into the equilibrium.

We consider two final examples relevant to the interpretation of death present in Stevenson's work. The first is the metaphorical death, the type that does not involve the cessation of a heartbeat. Stevenson cites the following exchange between a social worker and an Inuit family, upon returning their daughter after several years of tuberculosis treatment.

> "On arrival to the house … the social worker … knocks on the door, says to the Eskimo couple who lived there … 'Here is your daughter we brought her back for you,' to which they replied 'We don't have a daughter. We never had one.' 'Well yes you did but that was a long time ago.' And they say, 'Yes, but she died. The white man took her away and she died. We've never heard of her since'" (Pearson 1973).

There is a myth that if you touch a baby bird that falls out of its nest, then its mother will reject it. While that myth is unsubstantiated, the daughter in this story is very much dead to her parents. It is not simply a matter of racism (because the doctors are white). Stevenson explains the above example as a problem of forced cooperation (compliance) of the Inuit with an uncaring health system and how removing an individual from the traditional social system causes one to be not recognized as alive anymore. But we can also explain it in the lens of the second example, which lies in the analysis of the Inuit word for "survival." According to Stevenson,

> "The word *annaktujuniq* literally means the state of one who escapes from sickness, hunger, danger. And in another context, the base *annaktuq* is used to describe an animal or quarry that gets away, escapes death. Survival is linked to escape. The intimacy of the other's death, the death that one escaped, is crucial – the other's death is imagined as one's own" (Stevenson 2012).

If we consider the act of survival to be the act of staying alive, this description is a fitting synopsis of previous interpretations. Living, or survival, is a constantly dynamic action: the act of being chased by death. Every life is intimately linked with a corresponding death; and death (in moderation) is a cause for celebration for its part in this cycle. Artificial restrictions on death and a stagnancy of lifestyle (such as in the case of the disowned daughter above) diminish how alive one is. One of the many possible facets of suicide among the Arctic youth may be to get closer to the act of cycle of life and death; in their isolation, suicide the only known avenue.

LIFE AND DEATH IN THE ARCTIC[13]

I am as ever tempted to chalk up all of the surprising aspects of historical death phenomena, especially infanticide and senicide which can be thought of as having economic effects, to simple maxims of pragmatism, as expressed by Balikci when surveying prior work on infanticide among Inuit peoples:

> "A basic postulate, 'life is hard,' and its corollary, 'the insupportability of unproductive members of society,' govern these acts" (Balikci 1967).

The "life is hard" postulate extends past reasons of efficiency; the simple fact of higher mortality and colonial oppression is an oversimplified but all-encompassing reason for suicide. I am also tempted to treat traditional beliefs of the afterlife in the traditional way: held in disbelief until there is sufficient evidence to prove it; Taylor, for example, says that his hypothesis about the secondary nature of canicide as a means to ward off malevolent spirits in the huskies "can probably never be proven with certainty" (Taylor 1993). I continue with the same hesitancy, that of a scientist: all of our evidence is given in the form of retrospective, uncontrolled studies.

That being said, that is the view with which I entered into the study of Arctic mortality and thoughts on death, captured by the single image of an hardened Inuit grandmother pondering her final moments in the abandoned igloo as she sends off the youth for survival. Having considered the evidence, it is safe to conclude that many of the deaths are more than matters of efficiency, despite their superficial appearance. It is similarly overly simplified and *dangerous* to reduce the problems of the contemporary youth to that basic postulate, "life is hard" – the work of Trout, Wexler, and Stevenson has the potential to save these lives now and in the future by better understanding the root causes, and to disregard it by being reductive will needlessly allow anguished, socially-motivated suicide.

By studying the Inuit traditions and beliefs surrounding death, we find that death is a complex, multifaceted framework rather than a single event. Understanding this framework gives us a better facility to understand its changes over time.

First of all, it does not always correspond literally to physical death. In most cases, physical death is associated with a rebirth in another world, and thus the soul remains alive, or at most momentarily dead. On the other hand, we have also seen instances in which one who is physically alive is dead in the view of the community, and thus lost from this cycle of life.

Secondly, this fits within a larger animistic framework. The cycle of life and death is not limited to humans, but also the ecosystem of prey, pack animals, and symbolic substitutes. However, not all deaths are considered equal, and in particular human deaths are optimal for the cycling of human souls.

Thirdly, that there is an aspect of living or surviving that is escape; and, conversely, to live in a world without escape means barely living. Given to a number of factors related to the industrialization of the Arctic regions, leading to more sedentary lives and the destruction of traditional hunter ways of living, one may regard the amount of life within the Inuit communities to have decreased. This also ties in with establishing dynamic equilibrium with the mirror world.

---

13 The title of this section borrows the title of the final chapter of Hester Blum's *The News at the End of the Earth*. Now we mention life as well as death, for the interpretations we've encountered have linked them together as cyclical and continuous; part of the same fabric.

These beliefs surrounding death, while not provably real in a scientific sense, are very real in their effects. That is why the topic is important for study: the risks, decisions, and emotions people generate are deeply tied to their beliefs. If the belief system under consideration is that of life and death, then we expect (and observe) the effects to be equally grave. For the sake of this essay, we focus on one particular interpretation of death – the "mirror realm" of the Chukchi peoples, as described by Willerslev – and take note of its effects. Picking another interpretation will lead to a different analysis.

Changes in beliefs also lead to changes in effects. The meaning of suicide and death has degenerated unpleasantly among the youth in Arctic regions in modern times. Rather than the voluntary death achieved by their elders that was unifying and respected, the youth suicide epidemic is a denial to colonial control and its maladies. Suicide has become disgraced and often spiteful against family and loved ones or against oppressive social situations, which has the opposite effect it used to cause. Voluntary death by a family member is illegal, the elders are too distant, and the healthcare system only has a superficial meaning of care. The concept of death has been cheapened – reduced to worldly urges and a dull survival without consideration of the mirror world.

With this understanding, we return to our role as outsiders.

We return to that initial fascination with the Inuit tradition of voluntary death that allures Kpomassie, myself, and many others, as well as that general "benevolence." In the photos of the Inuit people smiling when they had so little food, because they did not fear death, because they knew that their soul does not end even if their life perishes? That if the life was scarce in this world, that it would be bountiful in the mirror world?

We return as outsiders in globalized healthcare and climate change. Stevenson proposes that "if we let suicide remain a wound rather than a problem to be solved through cooperation, we can experience the suicidal imaginations, its desires and negations, rather than circumscribe it with meaning" (Stevenson 2012). No simplistic, procedural methods aimed at the effects will improve the root cause of confused belief systems. In the same that the tackling the polar imagination has led us to gain valuable insight into why and how we chase the delusions about Arctic and Antarctic regions (and inconsiderately destroy it with climate change), we must now tackle the suicidal imagination.

WORKS CITED

Aaen-Larsen, Birger, and Peter Bjerregaard. "Changes in Causes of Death and Mortality Rates among Children in Greenland from 1987-91 to 1992-99." *Scandinavian Journal of Public Health*, vol. 31, no. 3, Sage Publications, Ltd., 2003, pp. 187–93, http://www.jstor.org/stable/45137626.

Andersen, Thomas, and Birger Poppel. "Living Conditions in the Arctic." *Social Indicators Research*, vol. 58, no. 1/3, Springer, 2002, pp. 191–216, http://www.jstor.org/stable/27527008.

Balikci, Asen. "Female Infanticide on the Arctic Coast." *Man*, vol. 2, no. 4, [Wiley, Royal Anthropological Institute of Great Britain and Ireland], 1967, pp. 615–25, https://doi.org/10.2307/2799344.

Bogoyavlensky, D. D., and Michael Volshonsky. “Native Peoples of Kamchatka: Epidemiological Transition and Violent Death.” *Arctic Anthropology*, vol. 34, no. 1, University of Wisconsin Press, 1997, pp. 57–67, http://www.jstor.org/stable/40316424.

Day, Gretchen Ehrsam, and Anne P. Lanier. “Alaska Native Mortality, 1979-1998.” *Public Health Reports (1974-)*, vol. 118, no. 6, Association of Schools of Public Health, 2003, pp. 518–30, http://www.jstor.org/stable/4598895.

Ehrlich, Gretel. *This Cold Heaven: Seven Seasons in Greenland*. HarperCollins UK, 2002.

Evans-Pritchard, E. E. “The Meaning of Sacrifice Among the Nuer.” *The Journal of the Royal Anthropological Institute of Great Britain and Ireland*, vol. 84, no. 1/2, [Royal Anthropological Institute of Great Britain and Ireland, Wiley], 1954, pp. 21–33, https://doi.org/10.2307/2843998.

Fleshman, J. Kenneth. “Disease prevalence in the Alaskan Arctic and Subarctic.” *Acta Socio-Medica Scandinavica. Supplement*, vol. 6, Sage Publications, Ltd., 1972, pp. 217–26, http://www.jstor.org/stable/45200030.

Kattel, Pradip. “How can an entangled particle pair be ‘disentangled’?” *Quora*. 2019. Web. 10 Nov. 2021.

Kost-Grant, Brian L. “Self-Inflicted Gunshot Wounds among Alaska Natives.” *Public Health Reports (1974-)*, vol. 98, no. 1, Association of Schools of Public Health, 1983, pp. 72–78, http://www.jstor.org/stable/4627352.

Kpomassie, Tete-Michel. *An African in Greenland*. San Diego: Harcourt Brace Jovanovich, 1983. Internet resource.

Kral, Michael J. “The Weight on Our Shoulders Is Too Much, and We Are Falling”: Suicide and Culture Change Among Inuit Male Youth.” *The Return of the Sun: Suicide and Reclamation Among Inuit of Arctic Canada.* Oxford University Press, 18. Oxford Scholarship Online. Date Accessed 15 Nov. 2021. https://oxford.universitypressscholarship.com/view/10.1093/oso/9780190269333.001.0001/oso-9780190269333-chapter-3.

“NVDRS Frequently Asked Questions.” *Centers for Disease Control and Prevention*, Centers for Disease Control and Prevention, 28 Sept. 2021, https://www.cdc.gov/violenceprevention/datasources/nvdrs/faqs.html.

Pika, Alexander, et al. “The Spatial-Temporal Dynamic of Violent Death among the Native Peoples of Northern Russia.” *Arctic Anthropology*, vol. 30, no. 2, University of Wisconsin Press, 1993, pp. 61–76, http://www.jstor.org/stable/40316338.

Schrire, Carmel, and William Lee Steiger. “A Matter of Life and Death: An Investigation Into the Practice of Female Infanticide in the Arctic.” *Man*, vol. 9, no. 2, [Wiley, Royal Anthropological Institute of Great Britain and Ireland], 1974, pp. 161–84, https://doi.org/10.2307/2800072.

Stevenson, Lisa. “The Psychic Life of Biopolitics: Survival, Cooperation, and Inuit Community.” *American Ethnologist*, vol. 39, no. 3, Wiley, 2012, pp. 592–613, http://www.jstor.org/stable/23250787.

Taylor, J. Garth. “Canicide in Labrador: Function and Meaning of an Inuit Killing Ritual.” *Études/Inuit/Studies*, vol. 17, no. 1, Association Inuksiutiit Katimajiit Inc., 1993, pp. 3–13, http://www.jstor.org/stable/42869857.

Trout, Lucas, and Lisa Wexler. “Arctic Suicide, Social Medicine, and the Purview of Care in Global Mental Health.” *Health and Human Rights*, vol. 22, no. 1, [Harvard School of Public Health/François-Xavier Bagnoud Center for Health and Human Rights, The President and Fellows of Harvard College], 2020, pp. 77–90, https://www.jstor.org/stable/26923476.

Tynes, Tore, and Tor Haldorsen. “Mortality in the Sami Population of North Norway, 1970-98.” *Scandinavian Journal of Public Health*, vol. 35, no. 3, Sage Publications, Ltd., 2007, pp. 306–12, http://www.jstor.org/stable/45149857.

Walsh, Matthew J., and Sean O’Neill. "Death, Rebirth, Objects, and Time in North American Traditional Inuit Societies." Mirrors of Passing: Unlocking the Mysteries of Death, Materiality, and Time (2018): 123.

Willerslev, Rane. “The Optimal Sacrifice: A Study of Voluntary Death among the Siberian Chukchi.” American Ethnologist, vol. 36, no. 4, [Wiley, American Anthropological Association], 2009, pp. 693–704, http://www.jstor.org/stable/40389836.

The life activity:
what it means to kill and die in the Arctic

Jonathan Lam

Hello, everyone! I'm presenting my paper topic, about death in the Arctic, particularly among the Inuit people and other native people living above the Arctic circle.

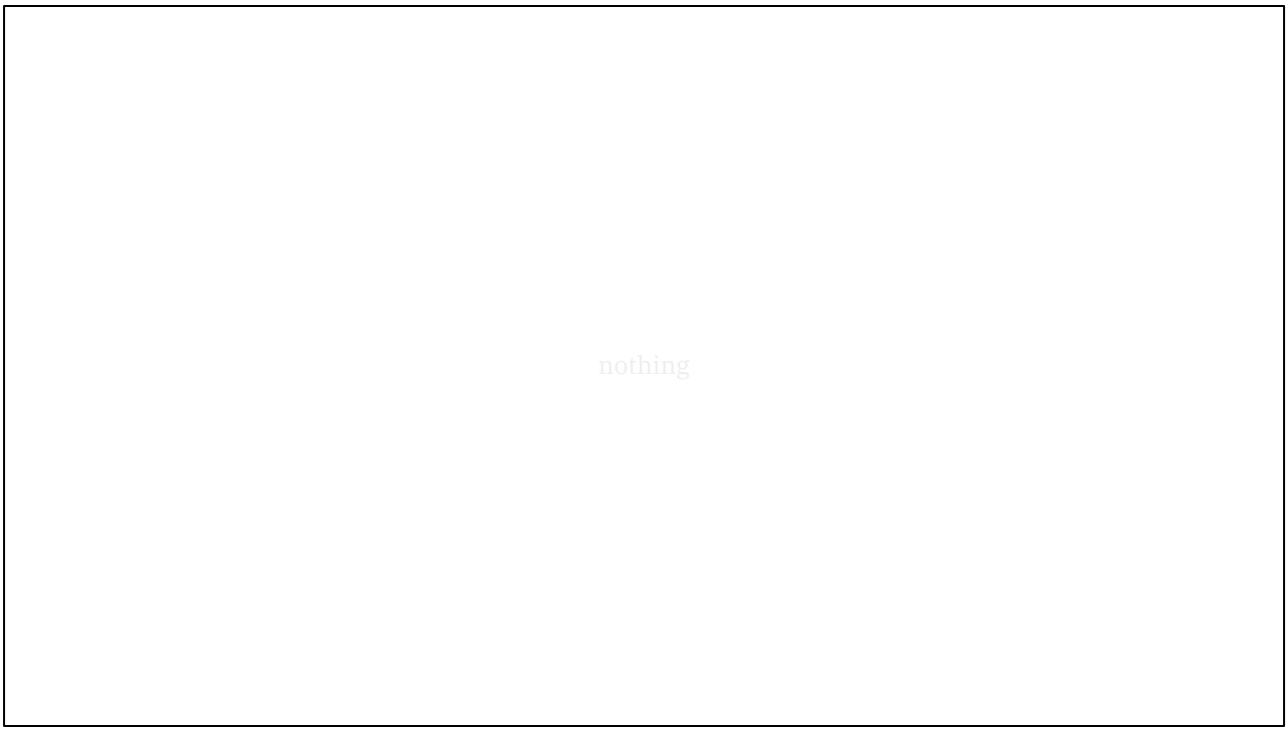
It sounds like a very broad and grandiose topic, to talk about death. And it is. It stemmed from a particular image from one of our readings, but became something very large. Therefore, I'd like to take some time to frame this discussion first, with a very big question.

What do you believe happens after death?

What do you believe happens after death? Or, what are your beliefs of life-after-death?

I don't expect anyone to answer this right now. But take some time, think about it for a minute, because chances are you don't think about this every day. I'll take 30 seconds to pause here.

I'm not expecting anyone's answer to this, but this will help warm up this discussion. If someone wants to share, then feel free. But I will go ahead and share my own. And my own belief of what happens after death is…



Nothing. We evolved from some primordial soup, a genetic algorithm that is a function of chemistry, physics, and probability. Consciousness, life, and what we may call a soul, to me, may be thought of as an advanced survival mechanism. To reason about our lives allows us to make really good decisions about survival. But, once we're done with living, and the electricity stops running in your brain, the consciousness stops, just like turning off a computer.

And maybe this is a depressing thought, because you might think, well, there's no point to living -- everything's going to be gone at some point. And this was really a struggle I had around those angsty years of middle school to early high school. I think for around a year or two, every night in bed I would play through scenarios of death: either imminent death or terminal illness. Inescapable death. Usually involving family. And then thinking about how I would react afterwards, when they had passed, and there was no way to reach them.

Eventually I overcame this, but it was a few years of thinking about it. What got me over it was just that school got too busy around sophomore year of high school, and I don't think I've ever really caught a break since then. But in a way, it's crowded out any thoughts about death, and I've become much accepting of this nothingness -- if something were to happen, I think now, the best thing to do is to keep doing all the other things that are to be done.

At this point, I'm wondering if anyone else would like to share their beliefs on life after death.

How does the way you think about death affect the way you live?

This sets us up for the second question, and the one that is really more important: how is this belief linked into your life? How did it form from your circumstances, and how does it affect your decisions?

So, rather than a explanatory view of death, where we try to rationalize any particular view, this is more functional and practical, and something we can work with. Also, we'll probably never get a global consensus on what truly happens after death.

In my case, I think that confronting the hypothetical for some time, and eventually realizing that the best way to stop worrying about an untacklable problem like death, is to simply immerse yourself in other things. That probably changed the way I think about worrying things. It might also be why I am such a procrastinator.

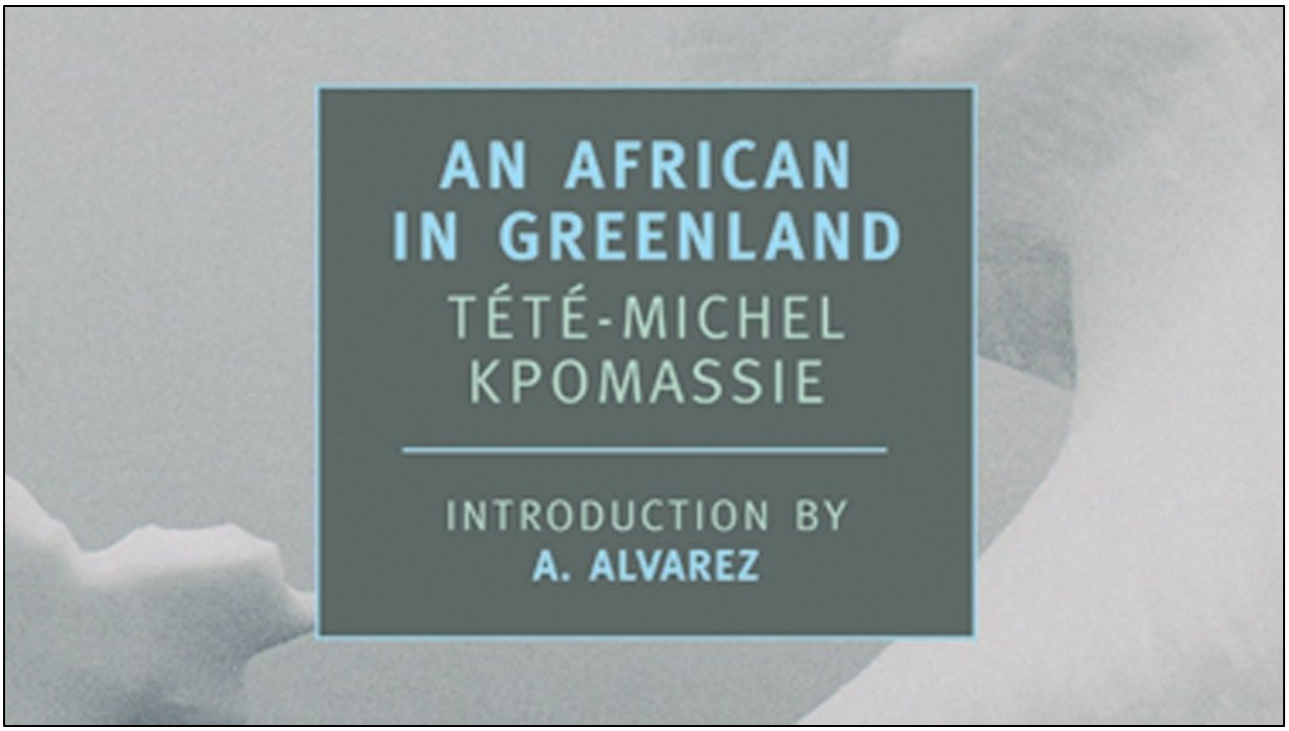
“If you live each day as if it were your last, someday you'll be right. Every morning I looked in the mirror and asked myself: If today were the last day of my life, would I want to do what I do today?”

Steve Jobs

As another example, take this well-known quote from Steve Jobs. [READ QUOTE]

We can pick up some aspect of an interpretation of death even from this quote. It implies that we should attempt to achieve as much as possible in this lifetime, otherwise we may feel unfulfilled at the end of our life journey, which seems fairly reasonable. But if we consider alternative interpretations, it may be less reasonable. For example, take the hypothetical belief that after one dies, they are transferred to an afterworld where they never age. In this case, one may “strategically” die young in the ordinary world so that they live forever young in the afterlife. If a long life is their goal in their first life, then they will be punished for that in the second. Or, imagine a world where the ones who are most fulfilled in their first life are given poor chances in the afterlife. If this were the case, Steve Jobs’ advice would be harmful rather than helpful.

As mentioned before, we will never know the true nature of life after death. But by now, hopefully you can see the lens with which we will try to view the Inuit people, who must have formed a different perspective on life after death, given their tough situation.



That brings us to this book. There is a lot of death in this book. To name a few: freezing, drowning, starvation, drunk parents falling on and suffocating their baby, babies devoured by dogs, puppies devoured by dogs, a middle-aged man devoured by dogs, an execution of dogs suspected of manslaughter, able-bodied men falling through thin ice and dying of drowning or freezing, a carelessly drunk murder due to a relationship dispute, an accidental gunshot through the leg in a canoe, and many more that I have probably missed.

While this seems very random, the themes of death in this book seem to be fairly widespread -- accidental deaths are still much more common in Arctic regions than sub-Arctic regions of the same nations. The point is that death is in close proximity to the Greenlanders, and I wonder how they must think about it.

> “In the old days, both in Greenland and among the other Eskimos, the old people, so as not to encumber a migration, would elect to remain behind and die slowly in the abandoned igloos. It was a spontaneous, stoic, unforced decision, and one which to them seemed noble.”
>
> Tété-Michel Kpomassie

But, despite the ubiquity of accidental death, and the need to cope with it, still the most striking incident of death was in the shown quote: that of the voluntary death of elders. [READ QUOTE]

To me, there is a very strong emotion here: sacrifice of a human being that you care for, in order for the preservation of the youth. There is immense trust on both sides: both of the family leaving the elder behind, and the elder, facing that scenario of imminent, unstoppable death, as they wait out the rest of their time, alone.

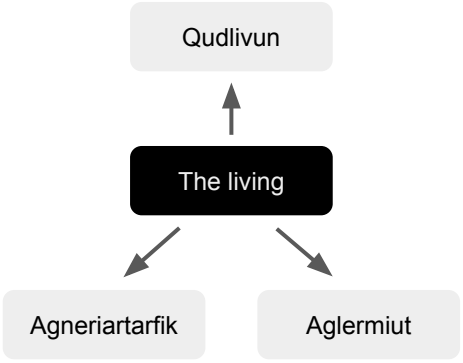
Smith, Craig S. "A Lost Art in the Arctic: Igloo Making." *The New York Times*, 21 Jun. 2017. Web.

…

It's hard to imagine what goes through their head at this time. We might say that it's only a matter of efficiency and survival, but it turns out that voluntary death among the Inuit people is not limited to situations of dire survival.

Afterlives
of the Netsilik Inuit

(Walsh and O'Neill)



It is time to talk about what the Inuit people think about life death. It's difficult to avoid overgeneralizing, but there do seem to be a number of patterns. A report by Walsh and O'Neill talk about the death traditions and beliefs of the various Inuit groups among northern Canada. There is a general trend of burial by simply abandoning the body, covered by a simple covering. This is a matter of pragmatics -- usually, it is too difficult to bury due to the frozen ground. The body may also be buried with some of the person's broken possessions, such as weapons, so that they are also "killed" and pass with the person to the other world.

What is the afterlife? The specifics seem to vary between Inuit groups, but there seems to be a general consensus on life after death. The Netsilik Inuit, for example, have the idea of three afterlives: one in the sky, the most honorable, for hunters and those who suffer honorable deaths; then there are two underworlds, one which is similarly joyful, and the other which is for the less honorable, and more negative.

This may be a fairly easy to understand belief of the afterlife, one that lines up closely with Heaven and Hell. There are of course subtle differences, such as what qualifies one to enter one realm or the other, the existence of two "joyful" realms, and the "killing" of possessions -- all which can have additional interpretations drawn from them.

The “mirror realm”
of the Chukchi of Siberia

(Willerslev)

But the previous interpretation is not the only one. It also seems that a common theme among the Inuit people is a recycling of souls, i.e., a rebirth into the current world. One of these beliefs is described in detail by Willerslev.

“The Chukchi cosmos can perhaps best be described as a hall-of- mirrors world: Each thing is paired with almost endless doubles of itself, which extend in all directions and continually reflect and echo one another. For example, the much feared evil spirits, the ke'let (sing, ke'le) are said to live in camps and villages, travel about the country on sledges, and go hunting for prey as do human beings. The game they hunt, however, is the souls of men, which they call their "little seals" or "reindeer". From the viewpoint of a human being, the ke'let have monstrous and terrifying features, such as hanging eyes, half-formed bodies, and large mouths full of teeth. Yet, from the viewpoint of the ke'let themselves, they are the ones who are human, and they regard the human shamans who can attack and kill them as ke'let - that is, as evil spirits” (Willerslev)

I’ll ask you, the audience, what to think about this. This is actually the most important slide in the whole presentation, indecorous as it is. What kind of conclusions can you draw from this interpretation?

This example introduces the mirror world in terms of evil spirits, but it also applies to the living and the dead. The living and dead are said to occupy two parallel realms, and death in one means rebirth in the other. The two are thus locked in tandem with a fixed number of souls. Moreover, each realm sees itself as the living and the other as

the dead, and everything in the other realm is opposite the current realm -- e.g., the people are inside out. Each world is also known to exert some control over the lives of the other; only if the living die will people in the dead realm be born.

Thus there is both a reverence and fear of the dead; we must offer periodic sacrifices so that their world is plentiful, and they would not try to limit the life in the living realm. Willerslev notes that the voluntary death is considered the “optimal sacrifice,” worth more than livestock or other substitutes, and that this sacrifice is an additional reason for self-sacrifice: to preserve the equilibrium between the two realms.

Suicide:
then and now

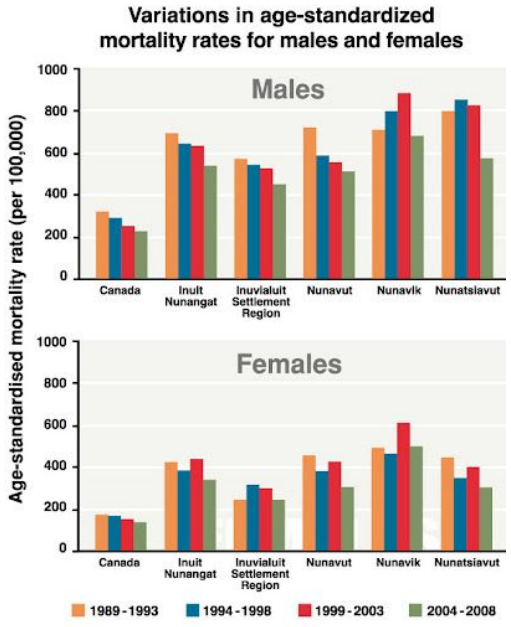
We turn our attention to how these interpretations manifest today. You may remember that alongside that quote from before, about the elders committing suicide. It is accompanied by the modern equivalent.

> “Today, the old sometimes commit suicide. An old man may be driven to such an extremity when he is *gamapok*, angry. Angry with himself. He goes out and never comes back. Sometimes he tells his family, and they do nothing to stop him. The old man has made up his mind and will not budge! Those who kill themselves in this way have often been great hunters. Diminished by old age and feeling themselves a burden to everyone, they don’t take easily to their changed condition.”
>
> Tété-Michel Kpomassie

[READ QUOTE]

But this sounds like more of a grumpy old man than a matter of sacrifice. It’s lost a lot of the nobleness. This is reflected in more than just one way in the Inuit community.

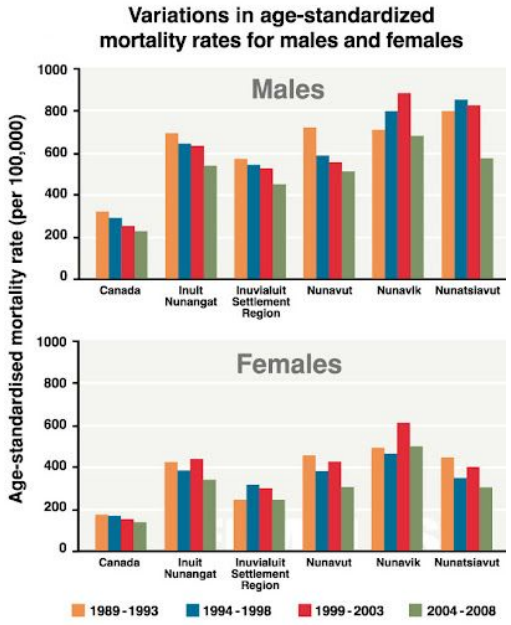
But is there still a vestige of that nobleness? Perhaps the old man thinks that he is a burden, and that killing himself is in some way helping the youth. This would make sense within the “mirror realm” framework.

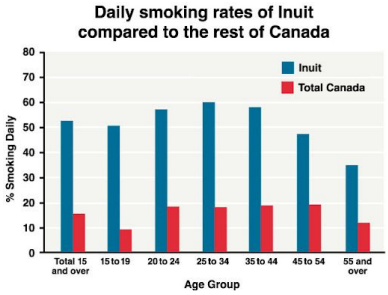


"Health Indicators for Inuit Nunangat." *Canadian Environmental Health Atlas,* Canadian Institutes of Health Research, 2021. Web.

As we slowly transition over to the pressing modern issue of youth suicide, let us first consider mortality rates in general. There are a number of studies that study mortality rate of Inuit people, especially in Canada and Alaska in the second half of the twentieth century. The picture is not pretty. We can very roughly attribute many negative health consequences to a more sedentary lifestyle, access to unhealthy foods, climate change and pollution, and systematic racism -- broadly speaking, the general symptoms of colonization. New education and healthcare systems are imposed that cause increased familial separation and loss of cultural knowledge, causing increased feelings of separation amongst the youth. The situation is similar among several groups in Siberia.

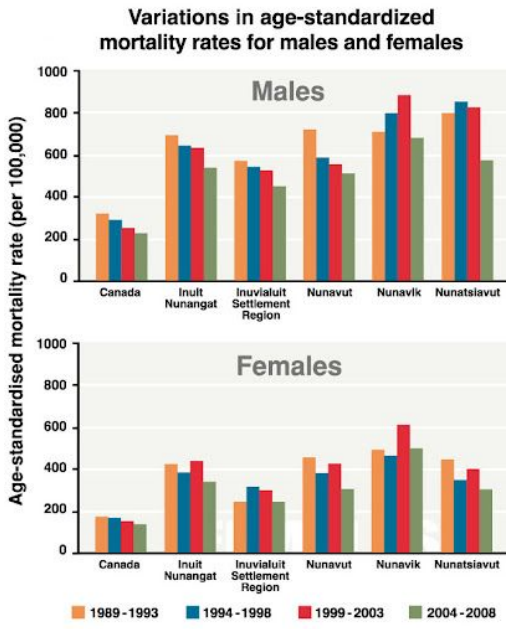
Thus, it is not surprising that mortality is still very high in Inuit societies. If we consider the following charts showing mortality rates of Inuit groups in Canada from 1989 to 2008, we can see that the mortality rate is consistently many times higher than that of Canada in general.

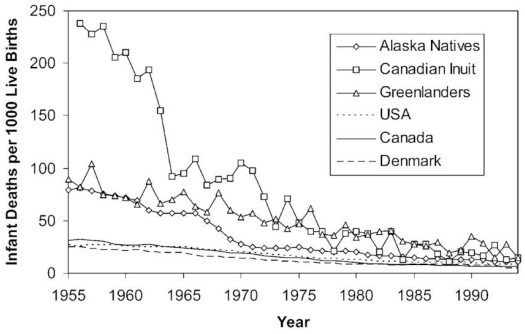


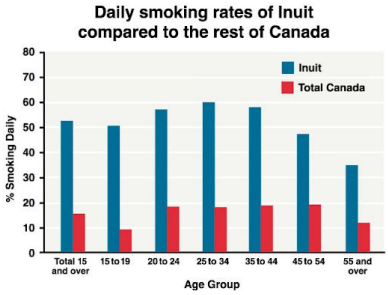


"Health Indicators for Inuit Nunangat." *Canadian Environmental Health Atlas,* Canadian Institutes of Health Research, 2021. Web.

This second chart is similarly not painting a good picture. High rates of tobacco and alcoholism have rocked Inuit societies.





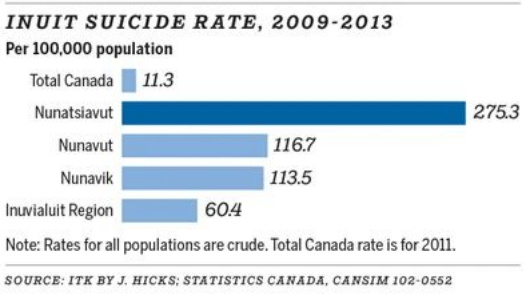


"Health Indicators for Inuit Nunangat." *Canadian Environmental Health Atlas,* Canadian Institutes of Health Research, 2021. Web.
Bjerregaard, Peter, et al. "Indigenous health in the Arctic: an overview of the circumpolar Inuit population." *Scandinavian journal of public health* 32.5 (2004): 390-395.

And in this chart in the top right, we see infant mortality rates among several Inuit groups in Alaska, Canada, and Greenland when compared to national averages. Luckily, modern medicine is able to help with some cases of mortality, such as infant mortality, but it is still noticeably higher than the national averages.

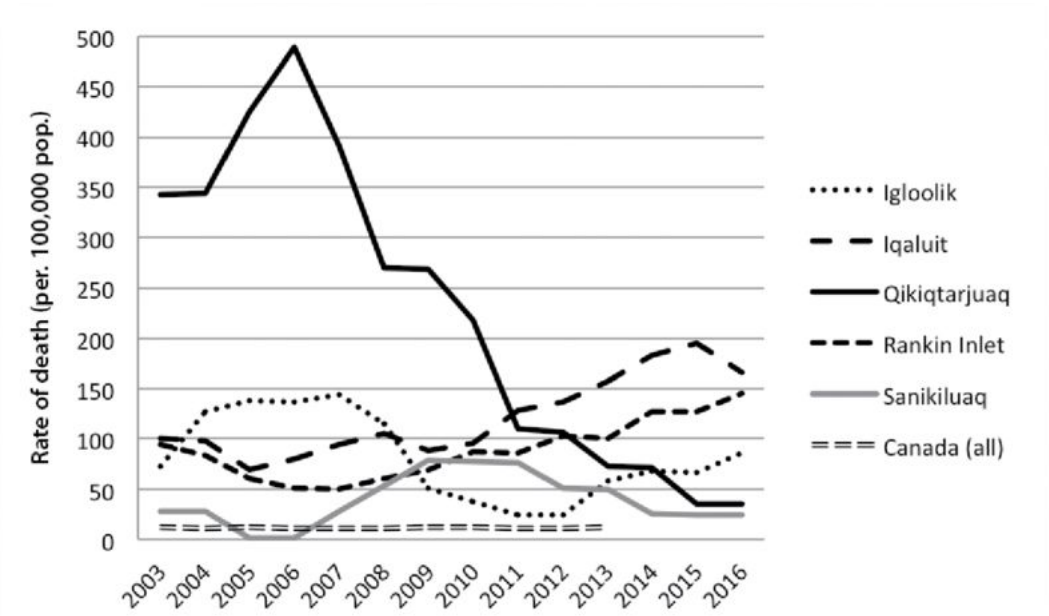
Just for context, some additional statistics on mortality from Day and Lanier about mortality rates in Alaska in the late twentieth century:

- unintentional mortality 3.9 times national average
- suicide 4.2 times national average (and fourth leading cause of death)
- homicide 3.3 times national average
- alcoholism mortality 14 times national average (Fleshman, 1968)



Crawford, Blair. "Fix 'unacceptable' social inequity to reduce Inuit suicide rates, report urges." *Ottawa Citizen*. 4 July 2016. Web. https://ottawacitizen.com/health/family-child/fix-unacceptable-social-inequity-to-reduce-inuit-suicide-rates-report-urges

We show some numbers related to suicide from only a decade ago. While most of the studies on Inuit mortality have mostly been centered around the mid-to-later parts of the twentieth century, suicide rates among the Inuit are still very high.

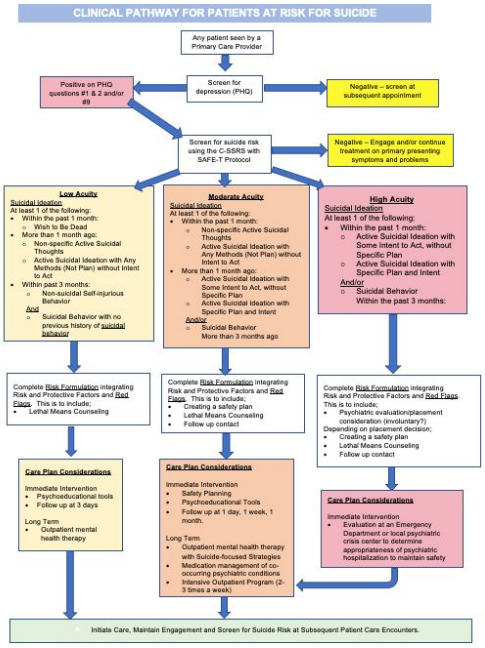


Hicks, Jack. "A critical analysis of myth-perpetuating research on suicide prevention." *Northern Public Affairs* 5.3 (2018): 44-49.

As another view, also from Hicks, we see the change over time from 2003 to 2016. Only one of the Inuit groups shows a marked decline in suicide rates; but all of them appear to be several times higher than the national suicide rate in Canada.

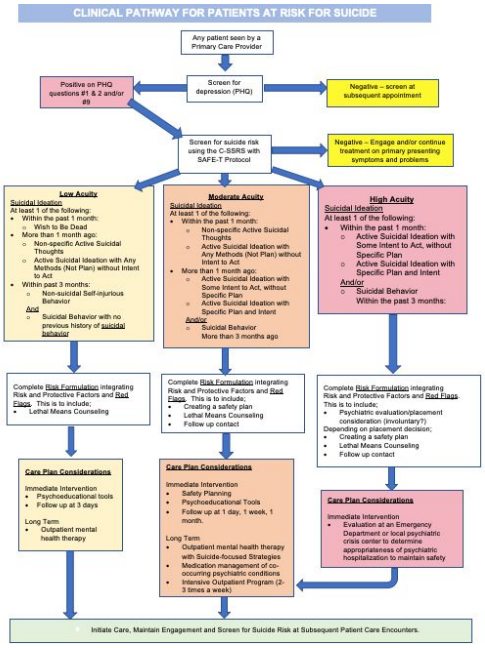
It is important to note that these suicides are of a different nature than that of the voluntary death. Willerslev in particular is very particular to separate the two. The modern suicides are of a much more spontaneous manner, and are usually spiteful: towards failed romantic relationships, towards family, or towards the systematic oppression. Some report that they see the dead coaxing them over to the other side, and that there is a greater unity in death than in life.

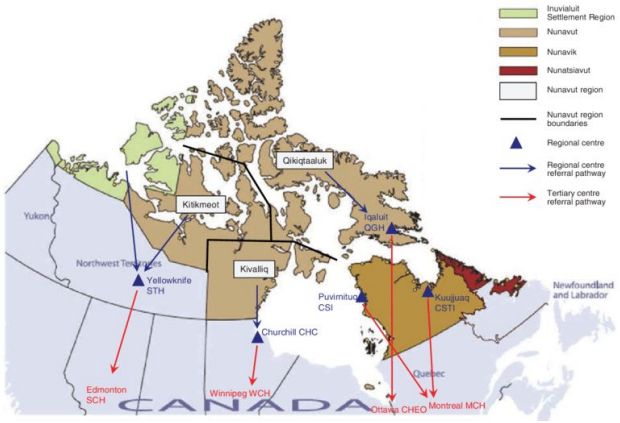
I can imagine this in the framework of the mirror world model in two ways. Firstly, the dead inviting over the living when they are not careful is a recurring theme. Secondly, the thought that not only is there life after death, but that the mirror world is opposite the current one in many aspects: if the current world is oppressive and limited, then the other one is united and free. Death in this model, dishonorable as it may be, is an escape to that opposite world.



"Suicide prevention in primary care settings." *Montana DPHHS*. Montana.gov. 2021. Web.

Some researchers attempt to drill down into the causes of these absurdly high suicide rates. Stevenson focuses on the colonial healthcare system as a system that doesn't work. She mentions that a lot of the healthcare workers turn to procedural flowcharts such as this one, rather than attempting to better understand the person and root cause.





"Suicide prevention in primary care settings." *Montana DPHHS*. Montana.gov. 2021. Web.
Banerji, Anna, et al. "Hospital admissions for lower respiratory tract infections among infants in the Canadian Arctic: a cohort study." *CMAJ* open 4.4 (2016): E615.

Another aspect is the isolation caused by modern medicine. Pictured here are the common referrals for medical facilities in Canada. They are expectedly sparse in the northern regions, and this causes hundreds to thousands of miles of separation from family. There are also horror stories of family members never returning from treatment, or perhaps returning months later and not being accepted by the family. Stevenson notes the following exchange between a social worker and the family of a child who had just returned from years of treatment for tuberculosis:

"On arrival to the house … the social worker … knocks on the door, says to the Eskimo couple who lived there … 'Here is your daughter we brought her back for you,' to which they replied 'We don't have a daughter. We never had one.' 'Well yes you did but that was a long time ago.' And they say, 'Yes, but she died. The white man took her away and she died. We've never heard of her since'" (Pearson 1973).

This may be considered the "curse of modern, anonymized healthcare." It clashes too strongly with the other aspects of Inuit belief systems on death. We may summarize our thoughts with another quote from Stevenson:

"The word *annaktujuniq* literally means the state of one who escapes from sickness, hunger, danger. And in another context, the base annaktuq is used to describe an animal or quarry that gets away, escapes death. Survival is linked to escape. The intimacy of the other's death, the death that one escaped, is crucial – the other's death is imagined as one's own" (Stevenson)

From this, we may be able to see why modern healthcare is the antagonist of the Inuit death tradition. To survive or live in the Inuit tradition does not necessarily align with the physical definition of being alive. But we can consider that not being in this cycle of being chased, of living in life-and-death situations, may be as well as considered dead. To be in a stagnant state for so many years, may be no better than recycling the soul in the mirror world -- what one may call the “life activity.”

“If we let suicide remain a wound rather than a problem to be solved through cooperation, we can experience the suicidal imagination, its desires and its negations, rather than circumscribe it with meaning.”

Lisa Stevenson

I’ll leave you guys on this note from Stevenson.

The life activity:

what it means to kill and die in the Arctic

Jonathan Lam

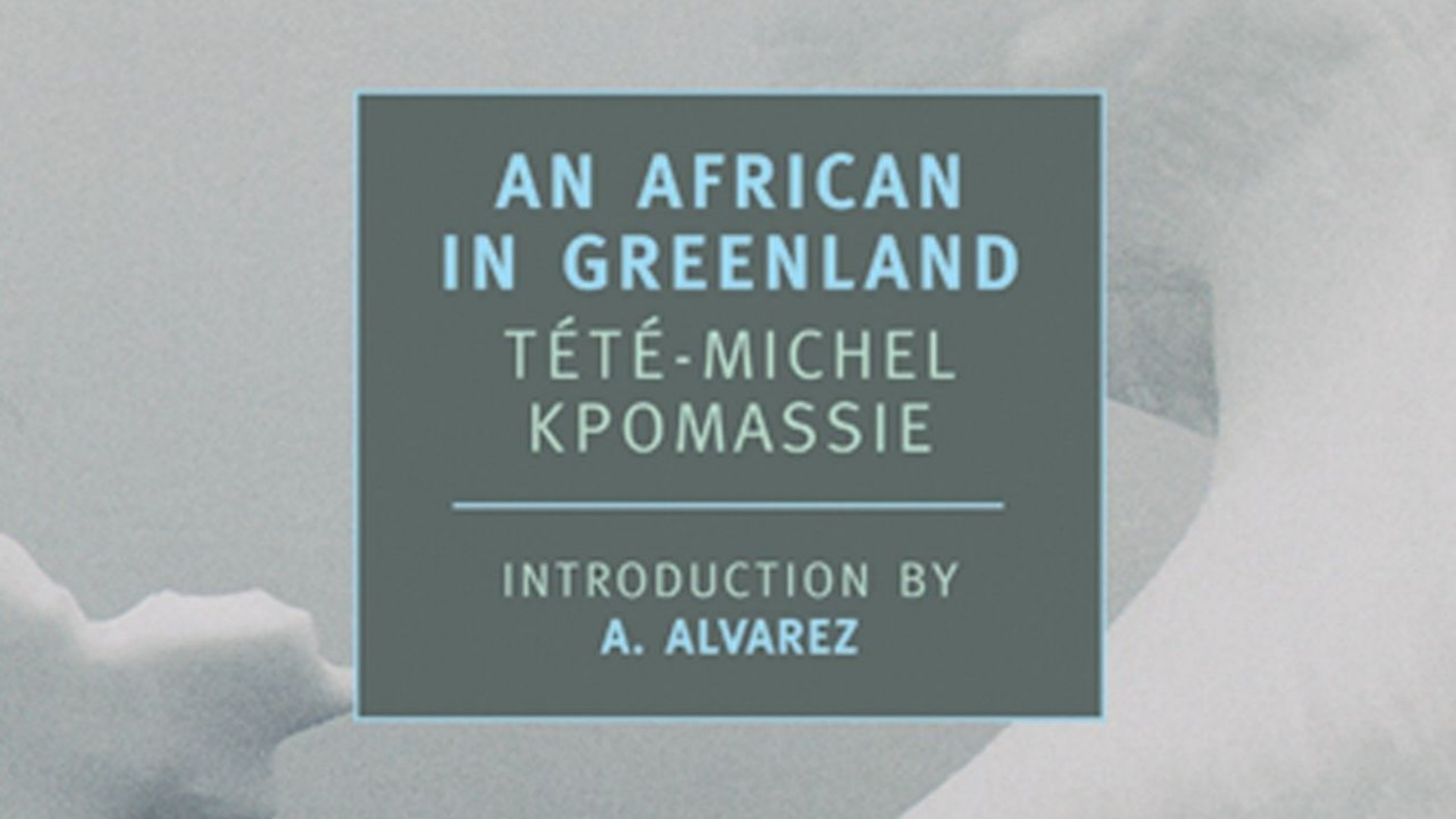
What do you believe happens after death?

nothing

How does the way you think about death affect the way you live?

“If you live each day as if it were your last, someday you'll be right. Every morning I looked in the mirror and asked myself: If today were the last day of my life, would I want to do what I do today?”

Steve Jobs


AN AFRICAN
IN GREENLAND
TÉTÉ-MICHEL
KPOMASSIE
INTRODUCTION BY
A. ALVAREZ

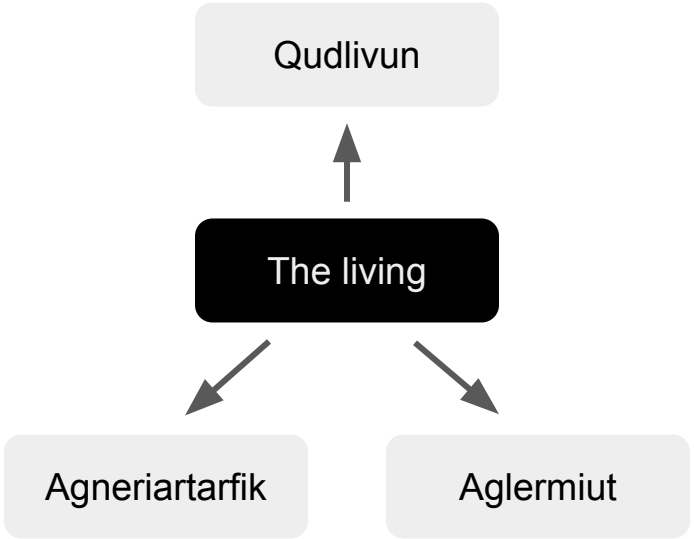
“In the old days, both in Greenland and among the other Eskimos, the old people, so as not to encumber a migration, would elect to remain behind and die slowly in the abandoned igloos. It was a spontaneous, stoic, unforced decision, and one which to them seemed noble.”

Tété-Michel Kpomassie

Smith, Craig S. “A Lost Art in the Arctic: Igloo Making.” *The New York Times*. 21 Jun. 2017. Web.

Afterlives
of the Netsilik Inuit

(Walsh and O’Neill)



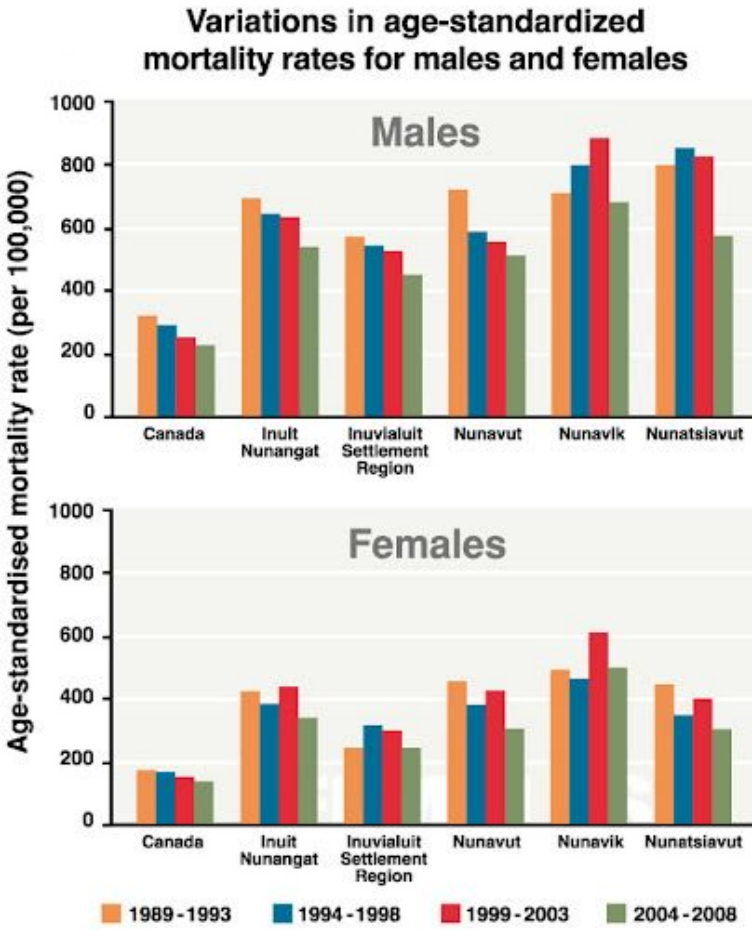
The “mirror realm”
of the Chukchi of Siberia

(Willerslev)

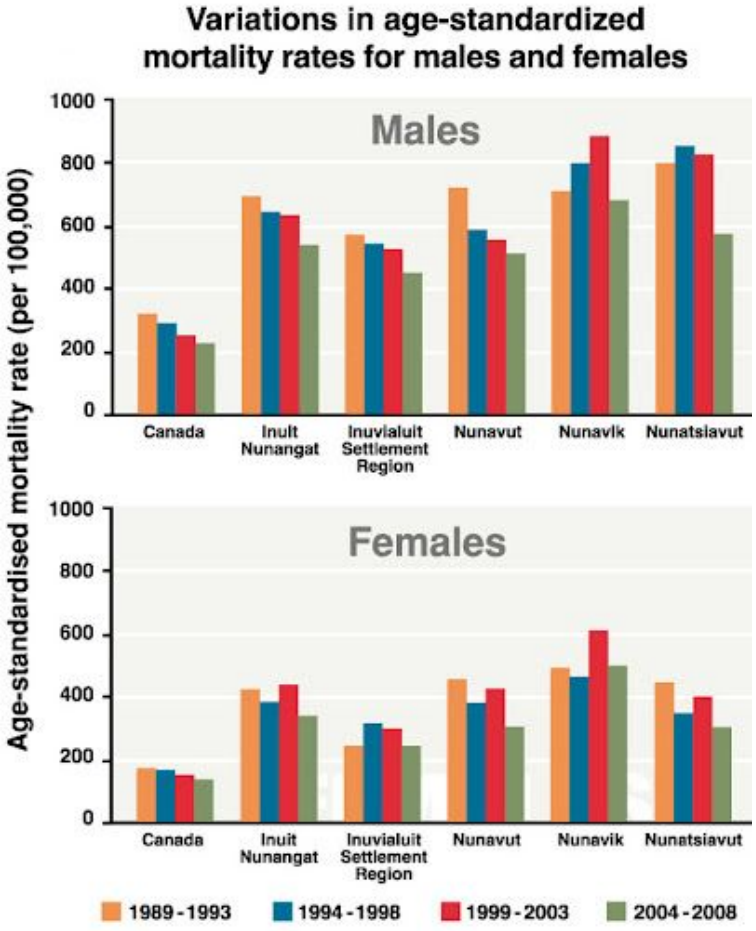
Suicide:
then and now

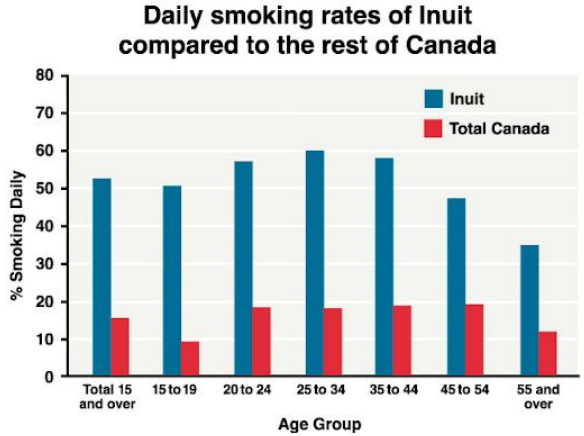
“Today, the old sometimes commit suicide. An old man may be driven to such an extremity when he is *gamapok*, angry. Angry with himself. He goes out and never comes back. Sometimes he tells his family, and they do nothing to stop him. The old man has made up his mind and will not budge! Those who kill themselves in this way have often been great hunters. Diminished by old age and feeling themselves a burden to everyone, they don’t take easily to their changed condition.”

Tété-Michel Kpomassie

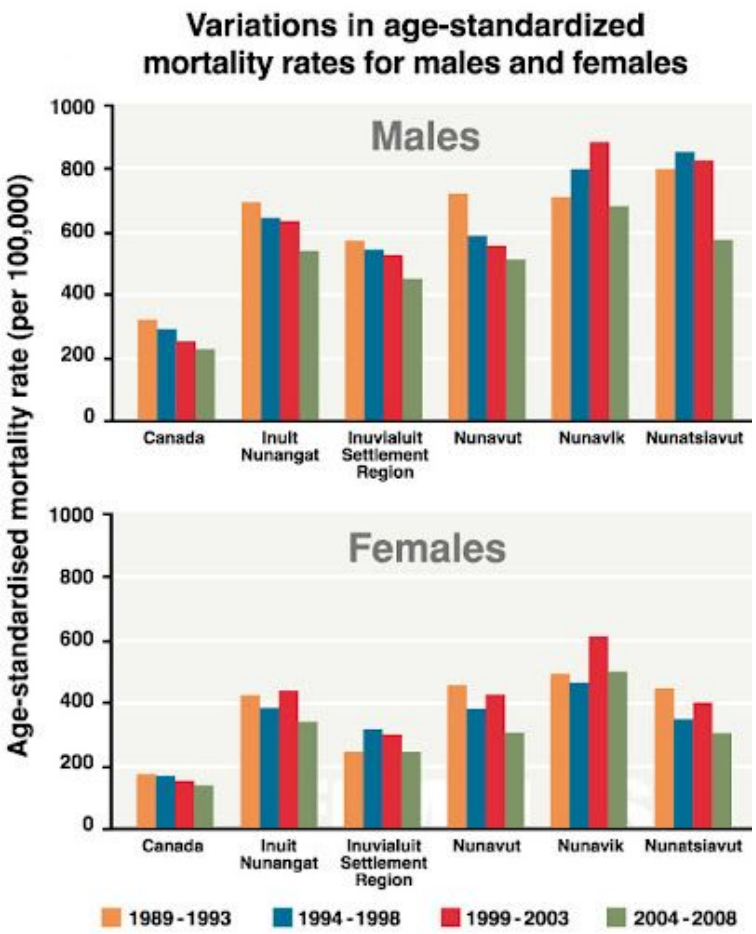


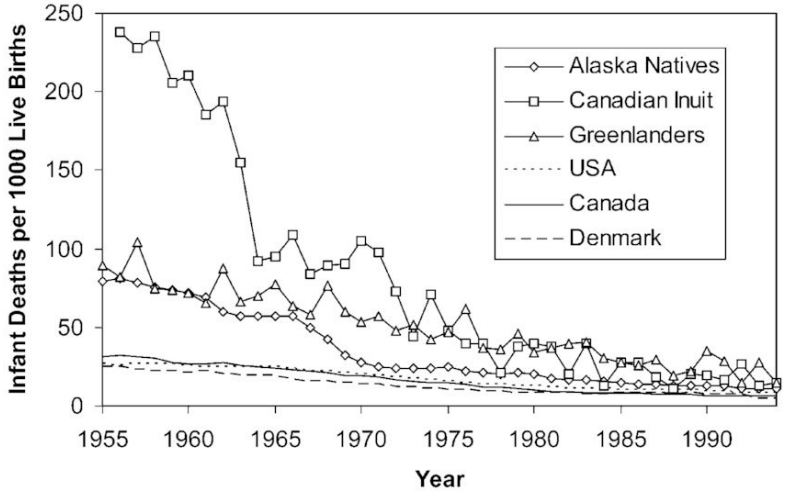
"Health Indicators for Inuit Nunangat." *Canadian Environmental Health Atlas,* Canadian Institutes of Health Research, 2021. Web.

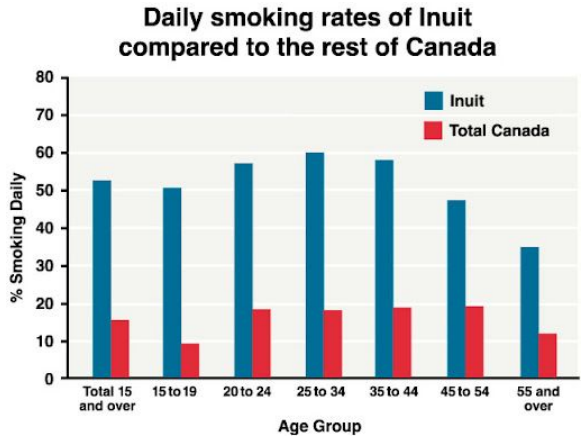




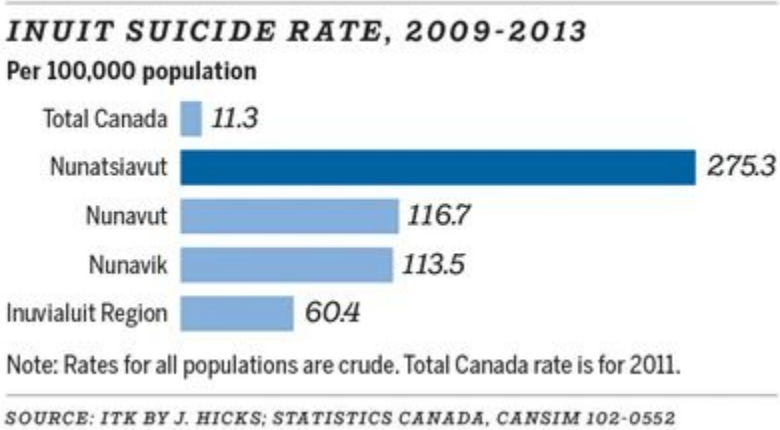
"Health Indicators for Inuit Nunangat." *Canadian Environmental Health Atlas,* Canadian Institutes of Health Research, 2021. Web.



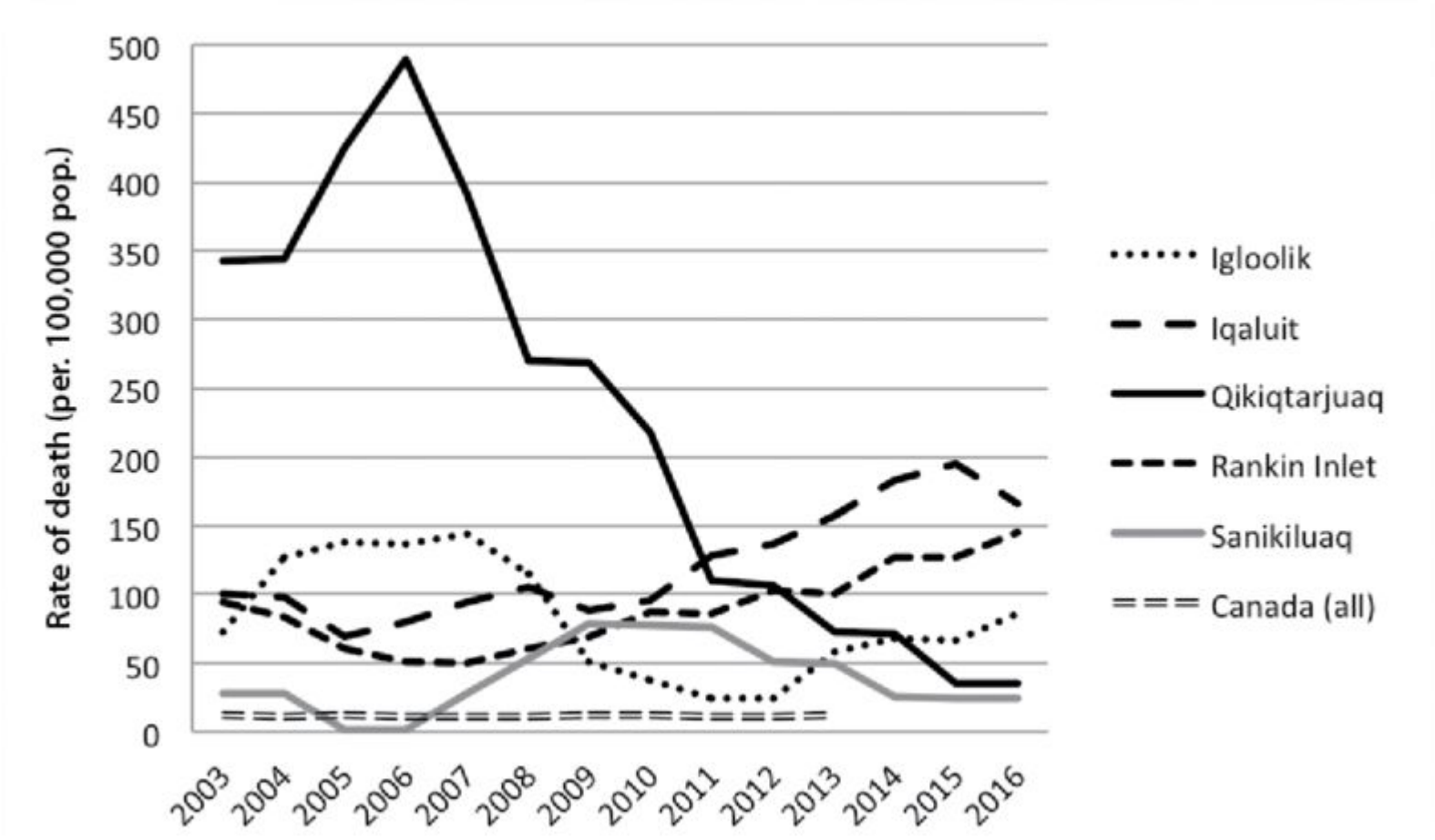




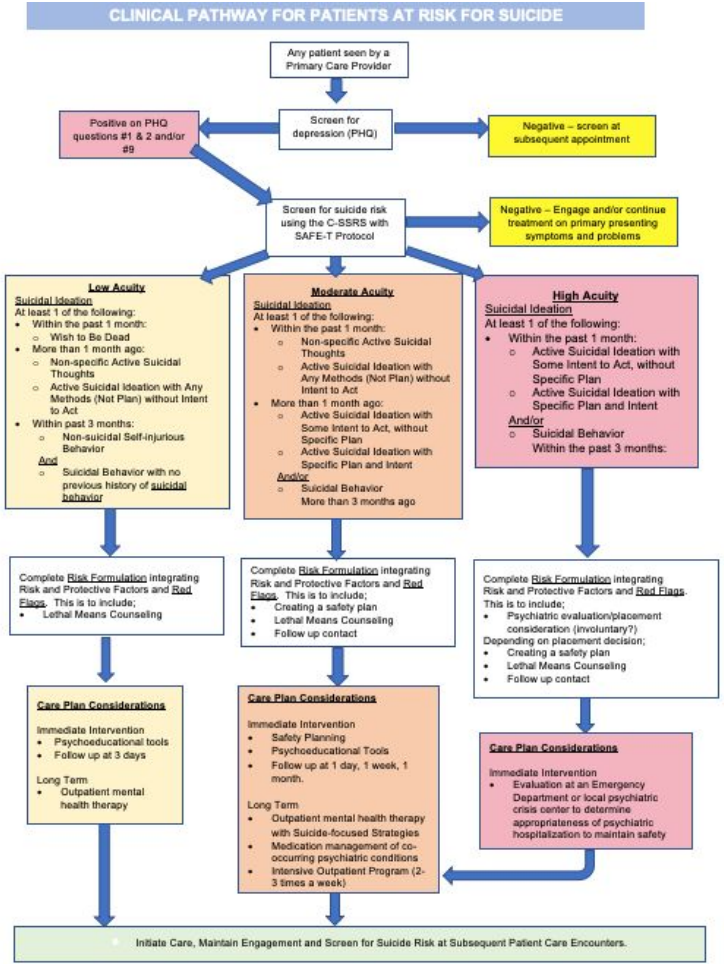
"Health Indicators for Inuit Nunangat." *Canadian Environmental Health Atlas,* Canadian Institutes of Health Research, 2021. Web.
Bjerregaard, Peter, et al. "Indigenous health in the Arctic: an overview of the circumpolar Inuit population." *Scandinavian journal of public health* 32.5 (2004): 390-395.



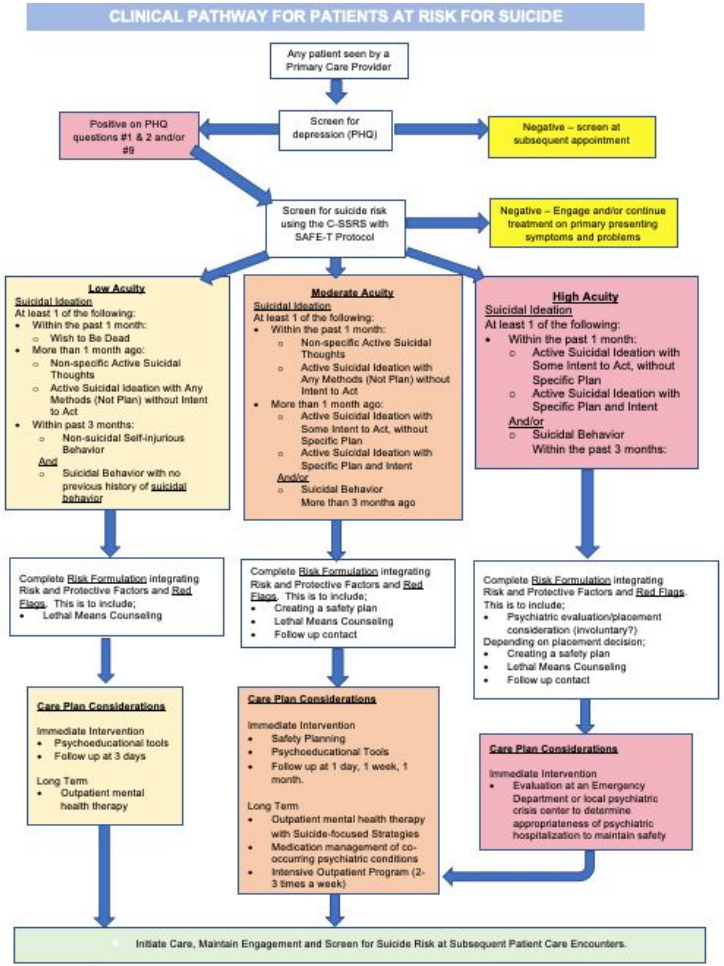
Crawford, Blair. “Fix 'unacceptable' social inequity to reduce Inuit suicide rates, report urges.” *Ottawa Citizen*. 4 July 2016. Web. https://ottawacitizen.com/health/family-child/fix-unacceptable-social-inequity-to-reduce-inuit-suicide-rates-report-urges

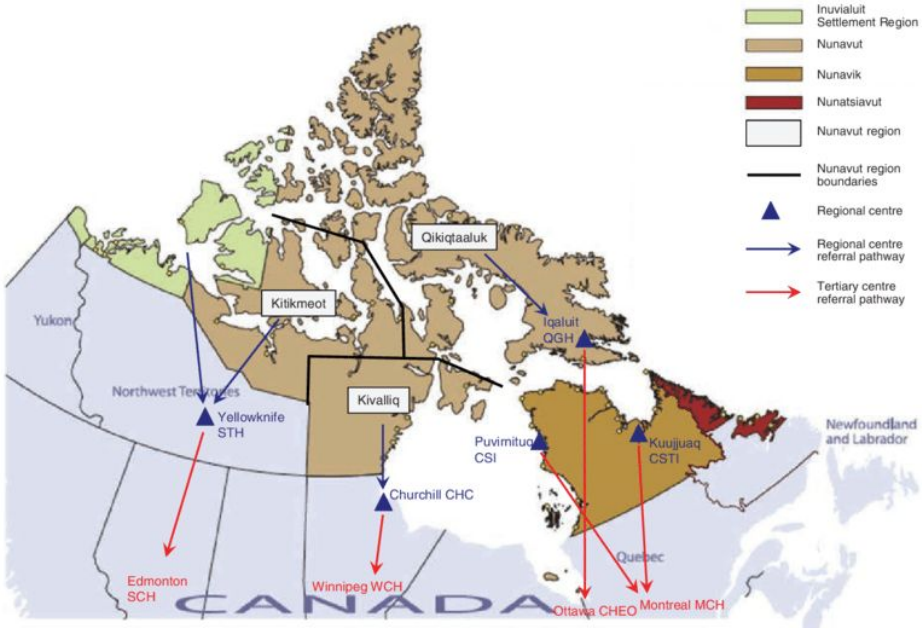


Hicks, Jack. "A critical analysis of myth-perpetuating research on suicide prevention." *Northern Public Affairs* 5.3 (2018): 44-49.



"Suicide prevention in primary care settings." *Montana DPHHS*. Montana.gov. 2021. Web.





"Suicide prevention in primary care settings." *Montana DPHHS*. Montana.gov. 2021. Web.
Banerji, Anna, et al. "Hospital admissions for lower respiratory tract infections among infants in the Canadian Arctic: a cohort study." *CMAJ* open 4.4 (2016): E615.

“If we let suicide remain a wound rather than a problem to be solved through cooperation, we can experience the suicidal imagination, its desires and its negations, rather than circumscribe it with meaning.”

Lisa Stevenson

Thesis planning

Jonathan Lam

9/13/21

Contents

1 Introduction

1.1 Problem/Motivation

For practical and historical reasons, programming languages have remained in a textual form that is difficult for beginners to learn. While modern programming languages are arguably more user-friendly than older languages (with more English-like syntax, more consistent syntax) and IDEs have much better linting and static analysis support (e.g., Microsoft's Language Server Protocol (LSP)), all practical modern programming languages are still focused on the same syntax-based textual format as back when FORTRAN was invented in the 1950's.

This was useful out of necessity in the past, and because programming was in the land of geeks who could program directly in hexidecimal machine instructions and write their own device drivers [citation: Linus Torvalds]. However, programming and software development as become a commonplace tool not only for the dedicated programmer, but also for engineers, scientists, artists, and professionals in many unrelated fields. Thus, improving the quality of programming education is becoming more important as the need for quality programs and programmers is improved.

As a programming tutor for several years, and mostly-self-taught programmer for even more years, I have had experience with dozens of programming students and their struggles. Of course, syntax is a very common concern; while it offers efficiency for veteran programs and IDE aids are very effective, the apparent difficulty of syntax can easily overwhelm the beginner programmer [citation: Hedy programming language], which can take attention off of the greater ideas. This effect is compounded when being immediately exposed to a number of different programming tools at the same time (e.g., the introductory course at my school pushes Git + Bash + C on the programmer in the first two weeks). The general cause is that source code is (and always has been) textual and non-structural, which is prone to errors. The textual source code representation has other problems, such

as overwhelming the user by a wall of plain text (syntax highlighting, "zen mode," and code folding only mitigate this problem slightly), and being a fundamentally linear format (which may not be the most intuitive since the programming language features fundamentally form a graph).

Besides syntax and other text-related difficulties, there are a number of other common difficulties shared amongst students related to programming languages and software development (without taking a class about software development principles), such as:

- *Translating ideas to code*: Taking an abstract set of specifications and implementing them in a controlled fashion. Untrained programmers can be frozen with "where should I start?" or miss implementation of many of the specifications without a clear definition of them. This can be allieved with a structured top-down approach, proper documentation, and specification of goals.

- *Clear and consistent documentation*: There is the quip floating around the Internet [citation?]: "Before the weekend, only God and I know what this code does. Now only God knows." We often get caught up with the implementation of some thought (the translation of ideas to code) that we forget what the original idea is. Having working with students, and even looking at the code of my coworkers in a professional environment, this is too often true, and the solution is very simple: a structured approach to documentation. Literate programming [citation: Knuth], "self-documentation," or "implicit documentation" are similar terms used to describe a simple solution, as well as generally clean code.

- *Comprehending error messages*: Error messages (at compile- or run-time) are often linked with the call stack or line numbers, which is a structural feature that may not reveal the intent of the code at that particular point. Of course, meaningful function names are useful to make error messages more meaningful for you or someone up the stack.

- *Inability to translate ideas between programming languages*: Certain languages embody certain programming paradigms better than others. Often scientific programming students get stuck in the imperative programming style (e.g., if first learning C) and produce spaghetti code in languages such as Python, which is much better suited for FP concepts (e.g., building meaningful composite operations out of `map` or `filter` primitives).

I propose that a programming model based on intentional programming may be useful to ease some of the common complaints about current programming education, as well as encourage general software engineering best practices and build practical code.

1.2 Intentional programming

Intentional programming [citation: The Death of Computer Languages, the Rise of Intentional Programming] was a concept engendered in the 1995 at Microsoft Research by Charles Simonyi, the creator of the WSIWYG editor and Microsoft Word. Simonyi suggested that the rate of programming language design was greatly slowed, and that there were still major unsolved problems when developing code. These problems include language non-intercompatibility, language non-intracompatibility, the mixing of intent (functional description) with implementation, the lack of "natural" notation, and comments being unused by the computer. Moreover, enough programming languages have proliferated that we are able to identify "genes" (features or "memes," using the original meaning of the term) carried throughout the languages and making them (un)desirable – such as garbage collection, object-oriented design, type systems, and special keywords or control-flow constructs – that we may shift our focus from programming in languages to programming in these generic language features.

The solution is to raise software to a level of abstraction that encodes functional intents (*what are you trying to do?*) separately of implementation (*what language construct should I use to do it, and what syntax would that entail?*). No programming language is perfectly intuitive to every user, and no syntax is perfectly natural for every user. Thus, by having a user encode their functional intent in terms of their own vocabulary, using their own predefined grammar. This gives rise to the idea of programming with "intents" as the building block of a program – blocks of presudo-code with a well-defined purpose and API that are expressed in the user's own notation.

Benefits of this system include the abolishment of predefined syntax (we are now working at the abstract-syntax tree (AST) level), more natural editing of code using a GUI (akin to how a WSYWYG document editor allows easier formatting of rich text than a plaintext editor), self-documenting code, and simpler mechanisms for structural editing of code (e.g., refactoring).

At Microsoft, Simonyi was able to demonstrate this idea by creating an IP IDE [citation]. Simonyi left Microsoft to found Intentional Software [citation], which developed tools implementing the intentional programming paradigm and improve programmer productivity. Since the creation of In-

tentional Software, there has not been much literature on intentional programming. At the time of writing, it is difficult to find much information about the company or its products, which seem to be never released except to some of the company's partners [citation]. Intentional Software was acquired by Microsoft in 2017.

TODO: do some digging about Intentional Software – did it ever produce any software?

1.3 Intentional programming for education

Intentional programming and variants have been attempted for use in the industry, but not aimed at students in programming. Intentional Software is highly geared towards domain experts who wish to incorporate more of their nontechnical specifications in the software system.

There are a number of benefits that a level of metaprogramming may bring to handling the problems with learning programming discussed earlier. This may include:

- *Code annotations*: The simplest implication of code generation is that it will force users to include documentation about each statement or small group of statements. Very rarely is a full description of the code included within the source itself – annotations are often included separately because text-based source code is not a good medium for explanation. This can be implemented in several ways: we can decide that every element of a program should fall under some annotated intent.

- *Eliminating syntax errors*: As mentioned earlier, we are working on the level of abstract syntax trees, and syntax errors will become difficult. In most cases, syntax is essentially abolished in favor of user-defined phrases.

- *Top-down programming*: Specifying a functional API before filling in the details (implementation details). This ensures proper documentation and well-specified APIs, and the code is always in a "complete" (compileable) state. This goes hand-in-hand with test-driven development (TDD).

- *Attention*: "Code zooming" (folding) on the level of (functional) intents, rather than (structural) blocks, reduces cognitive overload by only showing information related to the current intent, or a small number of nested intents.

- *Polyglot programming*: Different programming languages have different strengths in different domains, but it may be overwhelming to learn the proper syntax for each language.

- *Natural representations*: This is perhaps the most powerful aspect of intentional programming (more specifically, language-oriented programming), and which allows for quicker comprehension and improved recall than plaintext documents. This manifests itself in several ways:

  - *Personalized vocabulary*: The user expresses and idea in a phrase that is easy to understand. These phrases have placeholders for their arguments, which allow for an arbitrary combination of prefix, infix, or postfix notation without worrying about syntax ambiguities (since we eliminate parsing). Library writers are not bound by the terseness or syntax of the language, and are instead encouraged to express intent APIs in a natural prose-like manner. Users are also encouraged to freely alias intents using more intuitive phrasing. This also allows for seamless internationalization.
  - *Graphical program layout*: The program projection (view) and navigation is less cluttered than a plaintext document. Visual representations may also allow for non-linear (i.e., 2-D or even 3-D) representations of a program (like Scratch).
  - *Data projections*: Data that are well-suited to visual representations (e.g., equations, tables) are not limited to textual representations in a graphical editor.

- *Learn by example*: As opposed to learn by error, students may be able to express a higher-level idea using the IP representation and generate code in the target language(s). They can gain an undestanding of the language's syntax by examining the generated code rather than learning it by trial and error or by reading documentation.

1.4 Concerns

One of the major issues is that this system should not make programming more confusing or less intuitive. What we aim to do is encode generic traits inside an intermediate representation (pseudocode) that does not have those traits, and is smart enough to generate correct code. The problem with arbitrarily abstracting programming languages is that we need a precise formal specification in order to actually generate working code [citation:

`https://qr.ae/pGc3tj`, `https://qr.ae/pGc3tf`] – this is what a programming language is. Yet we can define high-level software by using many layers of abstraction. Thus, this is *not* a project about arbitrary code synthesis using AI or other generative methods; this is a framework to guide the programmer towards building cohesive, well-documented code as a means towards correctly implementing intent.

Another concern is that this software should generate correct code. By loosening the restraints of formal language and working on a level that is more general than any one programming language, we may produce code that is not syntactically or semantically correct. A considerable amount of knowledge about the underlying source code is necessary to generate correct code, and this limits the ability for the system to be language-independent.

One last major concern is that we cannot enforce that the user's intent lines up with the implementation. This whole project is an essay towards improving the chances of this happening, but there is no enforcement. There needs to be a better metric towards accomplishing this.

A major goal of this project is determining how to correct these issues. A possibility is to add API-driven constraints ("duck typing") to the language as a very basic type checking – however, this is circular as there is no way to check the correctness or completeness of the duck typing.

One might argues that everything in intentional programming is achievable with well-documented code (e.g., following good documentation). This is true, but this is the same of any compiler or generative programming – we are always compiling code down from a more intuitive format to a more obtuse format. In most older languages, we compiled languages down to assembly. Newer languages often transpile down to a reasonably-low level language such as C. In our case, we aim to transpile down to arbitrary languages, as well as drivers are written for that language.

A related concern is that assessment of this method is highly based on usability. While there are certain specific and measurable desired features (see features section, below), the effects of these features are unknown.

The concerns can be summarized with the following questions:

- How can we ensure that we produce correct code when loosening language formality? (And without introducing much complexity?) In other words, how can we decrease the formality of language while maintaining a precise specification?

- How can we enforce that the intent implementations match the users' true intentions?

- How can we assess this system (to test whether it actually aids students' learning process?)

2 Related work

Several visual programming languages exist to help aid programming education, especially for younger students. Notable examples include MIT's Scratch [citation] and App Inventor languages [citation], which are visual block-based programming languages. With block-based programming languages, syntax errors are scarce if not impossible. These languages are somewhat limited in their capabilities, and thus are not well-suited to "real-world" programming [citation].

The Hedy programming language [citation] has the intent of reducing cognitive load on students, citing the fact that the short term memory can be overwhelmed quickly by unimportant things like syntax. It brands itself as "gradual programming": there are progressively-difficult "levels" of the Hedy language, each adding new concepts, syntax, and functions on top of the previous level. This reduces the cognitive load at each step, and allows the user to proceed through the increasing difficulty like a leveled game. Hedy also notes that most programming education recommends trial-and-error learning, which may be discouraging to students who may prefer a more intuitive or guided approach.

There are many programming education websites currently that teach elementary programming in a rather standard manner – by teaching new skills gradually and having students replicate these in exercises. Khan Academy, Udemy, and Codecademy are examples of this. Another example Exercism, which focused mostly on providing mentored exercises without the standard learning track (the learning track was only added recently), which focuses more on a trial-and-error approach. Websites like these are a common entry point for self-taught programmers, but they face the problems discussed with standard programming education.

Intentional programming and metaprogramming are related to a number of other software-engineering practices and principles, such as (but not limited to): generative programming [citation], literate programming [citation], test-driven development (TDD) [citation], top-down design [citation], and language-oriented programming [citation]. It heavily involves the use of a domain-specific language (DSL) in its representation; Lisp is historically known for its metaprogramming capabilities [citation] and used as the basis for this project's DSL.

Domain-oriented programming is probably the closest idea to this that has been put into practice. See JetBrains MPS (Meta Programming System) – it uses the same jargon as Simonyi, such as "intentions" and "projections."

TODO: cite works talking about the faults in programming education

TODO: talk about error messages in programming languages

3 Features and functional requirements

This section is split into two subsections. General features indicate features that embody some general software engineering principle (and are thus difficult to measure directly). These will be argued intuitively. The other section includes measurable features.

3.1 General features

TODO: desired features: write something about intents?

3.1.1 Intents can express any language feature

TODO: intents can thus model language "genes", such as scoping, types, etc., even when it doesn't have it itself; we're dealing with syntactic modifications at this point, which may require some knowledge outside of our system (which the target language compiler should be able to detect)

3.1.2 Intents separate "intent" from implementation

TODO: intents may be implemented as one of several language features, e.g., control flow operators, special keywords, procedures, macros (code replacement), etc.

3.1.3 Automatic test framework

TODO: automatically generate unit tests for each intent based on its api

TODO: automatically generate assertions for its inputs based on its api

3.1.4 Gradual programming with default options

TODO: default values on intents (similar to default parameter values) used to ease use of intents

3.1.5 Language interoperability

TODO: specialized intents for I/O between programs written in different languages (a binary exchange fixpoint)

TODO: allow multiple target languages in the same project

3.1.6 Learning ecosystem

TODO: learn by example (a.o.t. learn by error), where examples all follow the intentional programming paradigm

TODO: package-like system with centralized system of intents

TODO: need some concept of fully-qualified intent names, and versioning; one way is to make each intent version have a different name so that a particular version is fixed

3.2 Measurable features

3.2.1 Single hierarchical namespace

TODO: only namespace for most use cases (except for those writing the low-level drivers) is the intent namespace – can group intents using a hierarchy and use fully-qualified names; simplifies model for the programmer

3.2.2 Single-file project, multiple views

TODO: replaces the traditional file/directory structure of a code project, which may confuse beginners

TODO: users can open several intent-level "views" into the project; see also: zooming capability

3.2.3 Target compilation information is associated with intents

TODO: if errors/messages are encountered during target code compilation, the generated code is linked to the intent

3.2.4 Literate programming

TODO: everything is annotated (enforced)

TODO: the API is annotated

3.2.5 Plain-english syntax and language-oriented design

TODO: talk about Lisp; however Lisp is hard to use (only prefix operators), so we want a more flexible syntax. Each intent is a small arbitrary (English) phrase with placeholders

TODO: not limited to english – easily translatable to other languages (most languages' syntaxes are English-only)

TODO: talk about recall [citation about recall?]

TODO: for teaching purposes, impedes cheating (since everyone's self-defined language is slightly different)

3.2.6 Structural correctness

TODO: scratch

TODO: paredit

TODO: we are basically working on the AST level

3.2.7 Structural completeness

TODO: top-down design

3.2.8 Syntactically-correct generated target code

TODO: cannot always be semantically correct due to additional constraints

TODO: structural correctness and completeness in the IR (correct AST) should translate to syntactically-correct code (but not semantic correctness)

3.2.9 Semantic "zoom" (code folding)

TODO: reduce cognitive overload

TODO: semantic (intent-based) a.o.t. structural (block-based) code folding

TODO: folded code can be replaced with its high-level description

3.2.10 Semantic error reporting

TODO: intents have access to the intent-stack, can be used to generate semantic (intent-based) exception messages rather than purely structural messages (call stack dumps)

3.2.11 Text-based intermediate representation

TODO: useful for text-based versioning/collaboration systems (e.g., Git)
TODO: easy to read, based on Lisp

3.2.12 Target language static analyses

TODO: implement the language server protocol (LSP) client
TODO: editing is much more feasible if we see the errors that are generated as we are editing, rather than when compiling the target code

4 (Proposed) Architecture

TODO: give a rundown of how this may be built

5 Analysis

TODO: talk about potential methods of evaluating this method: e.g., usability studies for UX
TODO: performance analysis: see if this generates correct code, whether it is "good quality code," and if the code generation is fast
TODO: talk about whether this matches the guarantees (in the features)

6 Concerns

TODO: what if a user edits the text file and leaves it in an inconsistent state?
TODO: going from a source file to an IR representation?