Animation Vectorization and Compression

Bassam Helal

September 12, 2020

Abstract

Table of Contents

1 Introduction **5**

2 Background **6**
- 2.1 Raster Graphics
- 2.2 Vector Graphics
- 2.3 Image Compression
- 2.4 Video Graphics

3 Related Work **8**
- 3.1 Current Research
- 3.2 Current Applications
- 3.3 Limitations

4 Implementation **9**
- 4.1 Pipeline
  - 4.1.1 Input
  - 4.1.2 Processing
  - 4.1.3 Output
- 4.2 Toolchain
- 4.3 Approaches
  - 4.3.1 Color Quantization Approach
  - 4.3.2 Connected Component Labelling (CCL)
  - 4.3.3 Edge Based CCL Approach
  - 4.3.4 Pixel Based CCL Approach
- 4.4 Limitations & Drawbacks

5 Software Engineering **12**
- 5.1 Methodology
- 5.2 Schedule (Gantt Chart)
- 5.3 Risks
- 5.4 Testing?

6 Evaluation **13**
- 6.1 Accuracy
- 6.2 Compression

7 Results & Findings **14**
- 7.1 Results
- 7.2 Findings

8 Challenges **15**

9 Reflection **16**

10 Future Work **17**

11 Conclusion **18**

12 References **19**

13 Appendix? **20**

1 Introduction

- Project Description
- Motivation
- Existing Literature
- Limitations of the literature that we will attempt to tackle
- Aims and Objectives
- Discoveries and results we found after our attempts
- Section Signposting

2 Background

- Signposting

2.1 Raster Graphics

- Brief summary of what we will go over (signposting)
- Pixels, Bitmaps, Image Data Buffers: basically the bare must knows about Raster Graphics
- Advantages and popularity of Raster Graphics
- Disadvantages like upscaling, pixelation, size issues with high resolution
- Brief summary of everything we just mentioned

2.2 Vector Graphics

- Brief summary of what we will go over (signposting)
- Declarative vs Imperative ie WYSIWYG vs WYSIWYM, describing the image using primitive geometry
- Advantages of Vector especially in contrast to Raster (size and scalability), mention uses and popularity
- Disadvantages like portability and light technical stuff like gradients and photographs
- Brief summary of everything we just mentioned

We can make a small table comparing Raster & Vector Graphics for the attributes like size, scalability etc

2.3 Image Compression

- Brief summary of what we will go over (signposting)
- Fundamentals of Compression (type-agnostic) things like compression rate and lossy-ness or accuracy
- Lossless Image Compression (PNG, BMP etc)
- Lossy Image Compression (JPEG, GIF etc)
- Brief summary of everything we just mentioned

We can have an image comparing popular image file types with regards to compression

2.4 Video Graphics

This section may be quite short

- Brief summary of what we will go over (signposting)
- How video is stored and how it relates to Images (mostly the same)
- Common file types and how they compress and store video data (MP4, MKV, H265 etc)
- Evidence and proof of why video is expensive (and also popular)
- Brief summary of everything we just mentioned

3 Related Work

Here we throw a lot of the current stuff from both research and industry and show their flaws and limitations (which we may or may not have tackled)

3.1 Current Research

- Brief summary of what we will go over (signposting)
- New Machine Learning methods for Vectorization
- New Machine Learning methods for Compression
- Novel approaches and ideas within Vectorization such as mesh gradients etc (no ML)
- Brief summary of everything we just mentioned

3.2 Current Applications

- Brief summary of what we will go over (signposting)
- Adobe Illustrator, Logos, Topography, Medical stuff like X rays etc Potrace
- Animations and Games using Vector as the development method
- Current Image & Video compression algorithms such as PNG, H265 and H266 (no ML)
- Brief summary of everything we just mentioned

3.3 Limitations

- Brief summary of what we will go over (signposting)
- Why ML is absolute hot trash
- Show off the flaws with existing technologies while showing how we tackled (and hopefully beat) them
- Discuss inherent limitations of this process, such as issues with Vector Graphics and input data being varying etc
- Brief summary of everything we just mentioned

4 Implementation

This may get long and technical, we should try to keep things fairly high level as much as possible, we will also show both Math, pseudocode and pipeline of the overall process and avoid boring low level details

4.1 Pipeline

The very basic common high level pipeline demonstrating the input, the processing, and the output

4.1.1 Input

An animated cartoon video, particularly one with vectorizable qualities, which we will further detail

- Low or no use of gradients so it has discrete colors
- Low or no noise
- No 3D stuff like shadows and reflections

4.1.2 Processing

We have tried multiple processes which we further explain in the approaches section, but here we should go over the general idea of what is necessary so:

- Frame extraction
- Conversion of each Frame to an SVG which will have a single SVG Path element for each color
- SVG Frame Joining which will have some smart compression built-in

4.1.3 Output

An SVG Video that is of course highly scalable and has a lower size than the original if upscaled and with very minimal data loss

4.2 Toolchain

No need to spend too much time on this part, just quickly mention the tools that we use and why we chose them. This makes the reader brace themselves for what we did without actually telling them just yet.

4.3 Approaches

Here we will explain our story of the last few months, the many approaches and their failures. This section should implicitly show the reader that the problem is very hard, harder than initially thought to be. If we show the difficulty and the struggle it can explain any poor results and can further show overall contribution that we tried something difficult

4.3.1 Color Quantization Approach

From the general process of the library ImageTracer.js

- Explain Color Quantization
- Advantages (very accurate, fast, good for logos and web based stuff)
- Disadvantages (Too dependent on quantization number of colors, large size and poor speed with high number of colors)
- Show pictures of original vs some different results from this approach (to show why it fails)

4.3.2 Connected Component Labelling (CCL)

Using Connected Component Labelling (CCL) to create discrete but adjacent components

- Explain CCL
- Reasoning behind approach
- Possible disadvantages
- Signpost for the following approaches

4.3.3 Edge Based CCL Approach

Using OpenCV Canny Edge detection

- Explain OpenCV and Canny Edge Detection
- Reasoning behind approach (edges should form polygons which will be color regions) and the expected result
- Advantages (fast, easy, accurate for edge detection)
- Disadvantages (not all edges form polygons and 1 pixel incorrect can have huge effects)

4.3.4 Pixel Based CCL Approach

Using the color of each pixel to form regions with close enough or similar colors

- Explain pixel data (RGBA) and what it means to be close to a neighboring pixel
- Reasoning behind approach (an image can be a partition of several color regions)
- Advantages (surprisingly accurate, little data loss, huge size savings)
- Disadvantages (slow, complex, Anti-Aliasing and noise affects result so too many 1 pixel regions)

4.4 Limitations & Drawbacks

Here we explain the overall limitations of all the processes we used and really any future process because of the nature of Vectorization of Raster data

- Brief summary of what we will go over (signposting)
- Speed or performance (never realtime and requires HPC on GPUs for good speeds) and optimizations required
- Lossy-ness, the transformation of Raster to Vector is by nature lossy because we are undoing Rasterization artifacts
- Size, this can be an issue for low resolution images or for very complex images, rasterization is often better
- Brief summary of everything we just mentioned

5 Software Engineering

Just the SE stuff and how we applied it during development, briefly mention the changes Covid did to the development

5.1 Methodology

Extreme Lean Agile because life is unknown beyond 2 days in these strange times we live in :/
Here we can show how the poor results affected our schedule and how things shifted because of unexpected results, we can definitely show that this is common in real life and is a good representation and preparation of the real world which has unexpected delays and poor results all the time and how we overcame it all (less sleep and more coffee :D)

5.2 Schedule (Gantt Chart)

Followed it well but started falling apart towards the end, very accurate to the real world :D
Mention that we started dissertation somewhat early to avoid the risk of running out of time

5.3 Risks

Time Time Time!!!

5.4 Testing?

Do we need this section?
Unit Testing for accuracy and of course Manual Acceptance testing (to ensure images actually open, are not corrupt etc)

6 Evaluation

Two main angles, accuracy to the original and compression rate from the original As much as possible we need to emphasize that we have done something great, no lying, just selective and sometimes exaggerated benchmarks in order to further our claim that we have accomplished something meaningful

6.1 Accuracy

- Define accuracy (aka lossy-ness or how much data is lost from original)
- Quantitative methods (compare each pixel to itself), and compare with other compression methods like JPEG
- Qualitative methods (human eye side by side) and compare with other methods like JPEG

6.2 Compression

- Define compression rate (aka how smaller is the result compared to original or some other)
- Quantitative methods (compare against original, AND against others including upscaled and uncompressed)
- Qualitative methods (file transfer speeds of smaller sizes and how smaller files are better for users)

7 Results & Findings

Here we show both the results of our development but also our own personal findings that someone who wants to research this kind of thing should be aware of that I discovered

7.1 Results

TODO
The good, the bad and the ugly (with detailed explanations :D)

7.2 Findings

- This is a very difficult problem, more difficult than initially thought to be
- GPU acceleration is very hard but improves speed shockingly well
- Node sucks for concurrency and doesn't have true multi-threading
- Differing input will have different results because of the limitations of vectorization

8 Challenges

- Toolchain used was not very mature or helpful sometimes
- GPU programming is insanely difficult but yields huge benefits (leading to very difficult decisions)
- Optimizations to increase speed are also difficult and especially so on a non-multi-threaded environment like Node
- Reducing the number of human inputted arguments to create a fully autonomous deterministic system that is not ML based and produces very accurate results is very difficult

9 Reflection

- How would you do it differently if to start from scratch (Use JVM or Native and optimize for GPU early)
- Project Management (Fairly good given the circumstances)
- So much new knowledge about many different things
- Very difficult but very mentally stimulating and rewarding project that I really enjoyed

10 Future Work

- Use better tools that allow for native multi-threading and easier and direct access to GPU
- Better optimizations and better compression
- Explore having a mixed raster and vector solution to solve the areas where vector fails
- Buy better hardware for faster development :D

11 Conclusion

- Summarize everything we said
- Conclude with final thoughts about the good, the bad and how this can further improve the world etc

12 References

13 Appendix?