Practical Analytic 2D Signed Distance Field Generation

Wasim Abbas Chris Doran Rich Evans Roberto Lopez Mendez
ARM Ltd

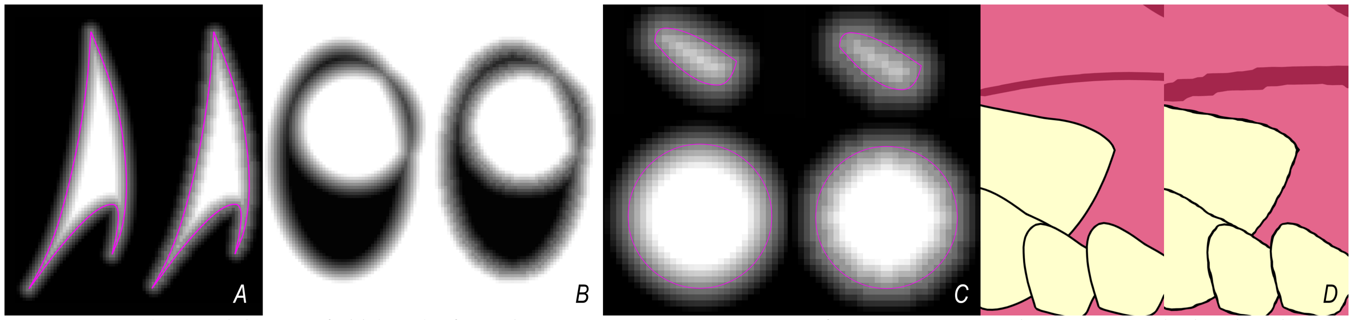

Figure 1: *A) Signed distance field (SDF) of a path containing 3 Bezier curves. Left image is generated using our method and Right is generated using 8SSEDT. B) And C) showing different paths where left in each is generated using our method and right in each is generated using 8SSEDT. D) A comparison of SVG Tiger rendered from SDF generated from the two methods.*

Abstract

In this talk, we present a novel technique to generate Signed Distance Fields (SDF) from vector paths. Unlike existing methods, instead of first rasterizing a path to a bitmap and then deriving the SDF, we can calculate the minimum distance for each pixel to the nearest segment directly from a path description comprised of line segments and Bezier curves. Our analytical method differs from previous works by computing distance in a canonical quadratic space. We have tested our code in Skia to accelerate SDF text rendering and we have found that our method is higher quality and more than 70% faster. Higher quality SDFs are achieved by sampling vector data at the SDF resolution required without losing quality to prior rasterization steps.

In this talk we present our approach and compare it to related work. We describe the algorithm in detail and how it was implemented in Google's Skia library. We present quality and performance results.

Keywords: SDF, Path, SVG, 8SSEDT, OpenGL ES

Concepts: **• Computing methodologies ~ Computer Graphics;** *Rendering*;

1 Overview

A vector drawing is represented as a sequence of path commands, such as an SVG document. A path contains one or more segments and there can be one or more contours in each path. A typical system for rendering paths would first generate an approximate SDF texture for each path and then combine them on a GPU. Our method accelerates the generation of accurate SDF from paths. We calculate distances to the true outline of the segments. In other methods such as [Danielsson 8SSEDT], a path is first rasterized to a pixel grid and then distances are calculated between pixels. The true outline of the path is lost when rasterized.

To analytically calculate distance to a Bezier segment we need to solve a 3rd degree equation. Computing this directly at each pixel is impractical for efficient operation. We present a set of transformations for a Bezier curve onto sections of a canonical parabola $y = x^2$. Calculating distances in this quadratic space is much simpler and more efficient, enabling efficient analytical SDF generation. A neat future of working with the canonical curve is that the cubic is already in standard form, and we can write down closed form solutions straight off.

Given the size of the SDF required, we traverse the bounding box of each path segment and calculate the distance of each pixel within the bounding box to the segment. Bounding boxes of some of the segments may overlap. This results in multiple distances for some pixels within the final image. We only save the minimal distance to each segment.

At the same time when distances are calculated, for each pixel a winding score is also accumulated. This is used in a second pass to resolve the minimum distances into 8-bit unsigned integer range.

2 Related Work

There are multiple existing techniques for creating high quality images from vector paths, including CPU and mixed CPU-GPU approaches which use texture-mapping graphics hardware to accelerate operation. We focused on evaluating the practicality and efficiency of our technique in Skia, developed by Google. Skia is used in the Chrome/Chromium browser and Android mobile operating system and uses a mixture of CPU and GPU

SIGGRAPH '16 Talks, July 24-28, 2016, Anaheim, CA,
ACM 978-1-4503-4282-7/16/07.
http://dx.doi.org/10.1145/2897839.2927417

algorithms to render paths. It employs SDFs for efficient text rendering. Use of SDFs in font rendering was introduced by multiple authors around the same time, including [Green 2007]. Skia uses [Danielsson 8SSEDT], one of the better-known algorithms for SDF generation, to generate SDF from font glyphs for rendering text. In other work, [Ronald N. Perry 2000] present a method for generating adaptive SDFs. Both of these methods require first rasterizing the path and so approximate the distance calculation. More accurate results were obtained by [Behdad 2011] which uses the vector path during SDF generation on the GPU. This gives better quality compared to the approximate SDF methods but at a very high runtime cost.

2 Results

We have tested our code in Skia. Skia uses 8SSEDT to generate SDF from paths rasterized with Freetype2 library. We have inserted our method in Skia replacing Freetype2 rasterization and 8SSEDT steps with our algorithm. Our algorithm can generate SDF for all paths in tiger.svg on Galaxy S6 in 960ms compared to 8SSEDT code path of 1660ms. This gives 73% improvements for this use case. Full implementation of our method and the formulas used are available in www.desmos.com at https://www.desmos.com/calculator/wxqfhychxu.

2 Future

Our current implementation is a CPU only method. It is possible to implement our algorithm on a GPU to calculate SDF with a canonical parabola lookup table (LUT). For each path segment a bounding box will be rendered using programmable hardware using the LUT as a texture. Given two same size Bezier curves with different shapes, it will occupy different regions with varying sizes of the LUT. This is why the LUT needs to be generated such that there is enough resolution at the origin of the parabola. It is also possible to store the parabola in question at several different scales in the LUT texture, and use the size of the curve in canonical-space to select which scale to use.

Acknowledgements

To Sam Martin for reviewing this abstract and Joel Liang for Skia implementation.

References

CHRIS GREEN. 2007. Valve. Improved Alpha-Tested Magnification for Vector Textures and Special Effects. http://www.valvesoftware.com/publications/2007/SIGGRAPH2007_AlphaTestedMagnification.pdf.

BEHDAD ESFAHBOD, 2011. Glypy. https://github.com/behdad/glyphy.

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones MERL – Mitsubishi Electric Research Laboratory

STEFAN GUSTAVSON AND ROBIN STRAND, 2011. Anti- Aliased Euclidean distance transform.

PER-ERIK DANIELSSON, 1980. Euclidean distance mapping. In Computer Graphics and Image Processing 14 (1980), 227–248.