Saffron Type System

May 2, 2005

Outline

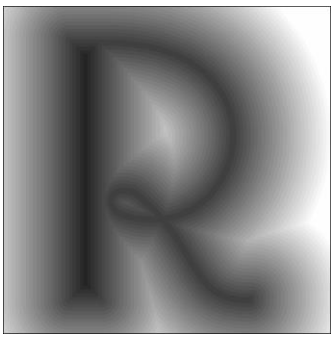
- How Saffron Works Today
  - Science and implementation
- Saffron Demonstration
- Limitations of the Current Approach
- Saffron Roadmap
  - Direct rendering
  - Hardware implementations
  - High quality small CJK
  - Enhancements and extensions
- Advantages over traditional approaches

Background

Distance Fields

- An informal definition
  - The 2D distance field of a shape represents, for any point in space, the signed minimum distance from that point to the edge (i.e., outline) of the shape

Shape

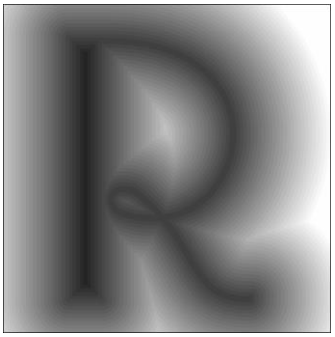


Shape's distance field

Advantages of Distance Fields

- Conceptual advantages over outlines
  - Represent more than the object outline
    - Object interior, exterior, and the outline
    - Represents an infinite number of offset surfaces

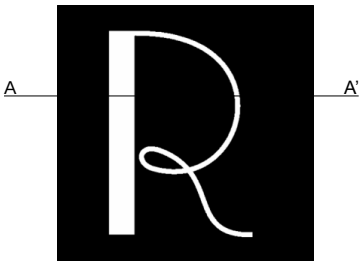
Shape

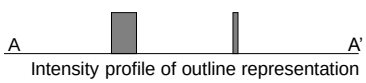


Shape's distance field

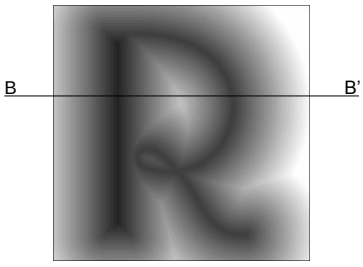
Advantages of Distance Fields

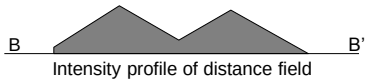
- Conceptual advantages over outlines
  - Gains in efficiency and quality because distance fields vary smoothly (C^0 continuous) and throughout space





Intensity profile of outline representation

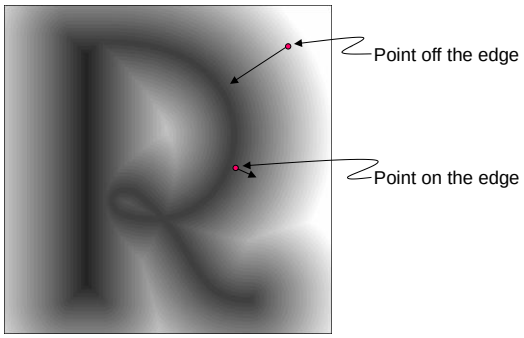




Intensity profile of distance field

Advantages of Distance Fields

- Conceptual advantages over outlines
  - Gradient of the distance field yields
    - Surface normal for points on the edge
    - Direction to the closest edge point for points off the edge

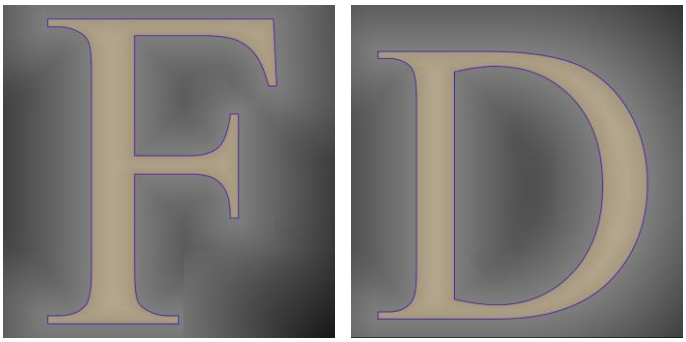


Advantages of Distance Fields

- Practical advantages
  - Smooth reconstruction of the edge
    - Continuous reconstruction of a smooth field rather than a continuous (i.e., analytic) or discrete (i.e., supersampled) reconstruction from a discrete representation
  - Trivial inside/outside and proximity testing
    - Using sign and magnitude of the distance field

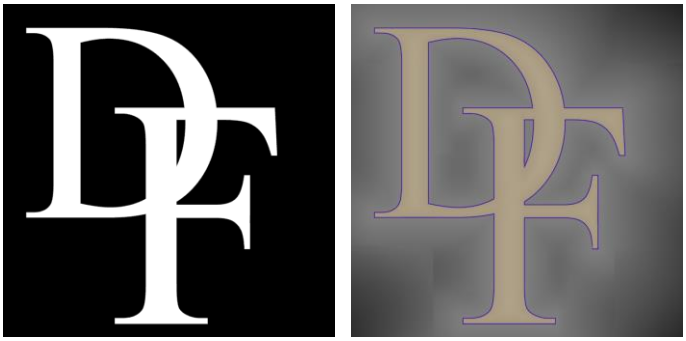
Advantages of Distance Fields

- Practical advantages
  - Fast and simple Boolean operations
    - Intersection: dist (A^B) = min(dist(A), dist(B))
    - Union: dist(AUB) = max(dist(A), dist(B))



Advantages of Distance Fields

- Practical advantages
  - Fast and simple Boolean operations
    - Intersection: dist (A^B) = min(dist(A), dist(B))
    - Union: dist(AUB) = max(dist(A), dist(B))

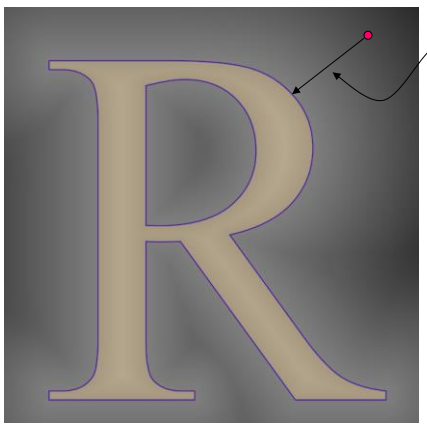


Advantages of Distance Fields

- Practical advantages
  - Fast and simple offsetting
    - Offset by d: $dist(A_{offset}) = dist(A) + d$

Advantages of Distance Fields

- Practical advantages
  - Enables geometric queries such as closest point
    - Using gradient and magnitude of the distance field

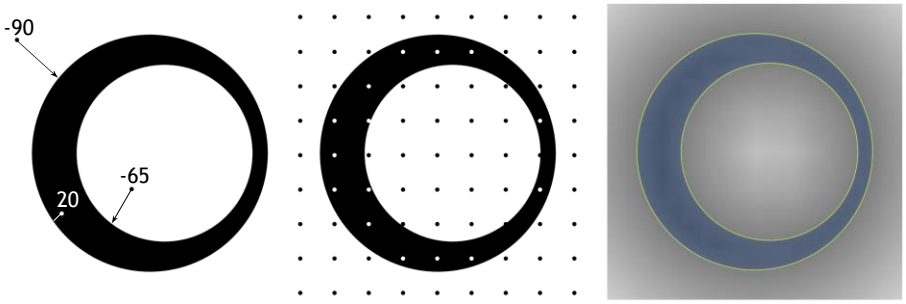


Distance Field Representations

- Analytic representation
  - For example, circle of radius R centered at (c_x, c_y)
    $d(x,y) = (R^2 - ((x-c_x)^2 + (y-c_y)^2))^{½}$
  - Problem
    - Unless the shape is simple, the analytic representation is difficult to determine and complex

Distance Field Representations

- Sampled Representation
  - Distance Maps: Regularly sampled distance fields



2D shape with sampled distances to the surface

Regularly sampled distance values

2D distance field

Distance Field Representations

- Sampled Representation
  - Drawbacks
    - Must sample at high enough rates to capture all the important detail such as
      - Corners
      - Voronoi region boundaries
    - Regular sampling ➔ if there is any detail present, the whole field must be sampled at high rates

Distance Field Representations

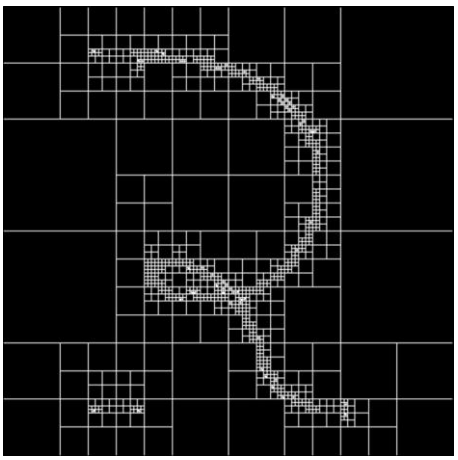
- Adaptively Sampled Distance Fields
  - Detail directed sampling
    - Sample at higher rates where there is significant variance in the distance field
    - Sample at lower rates where the distance field varies slowly or where accuracy is not needed
      - e.g., for points beyond a specified distance from a glyph's edge

Distance Field Representations

- Adaptively Sampled Distance Fields
  - Use detail-directed sampling of the distance field
  - Store sampled distances in a data structure for efficient processing
    - Have used hierarchical representations which have other advantages (e.g., level-of-detail)
  - Provide a reconstruction method for reconstructing the distance field from the sample points

Distance Field Representations

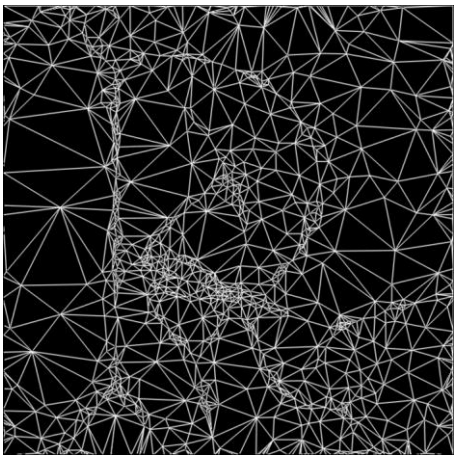
- Adaptively Sampled Distance Fields



Bilinear Cell ADF

Distance Field Representations

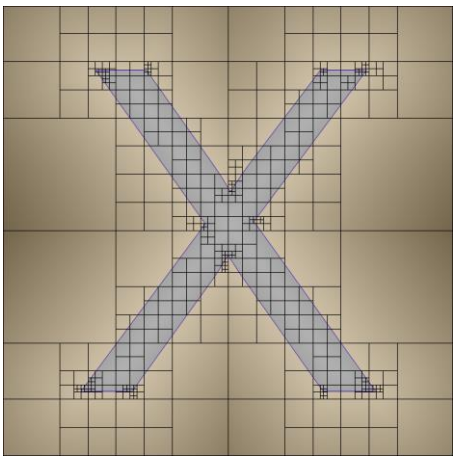
- Adaptively Sampled Distance Fields



Barycentric Cell ADF

Distance Field Representations

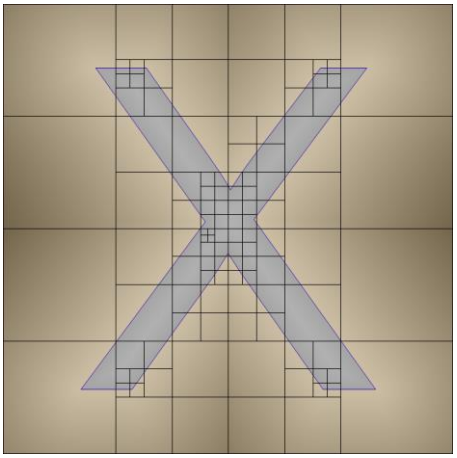
- Adaptively Sampled Distance Fields



Biquadratic Cell ADF

Distance Field Representations

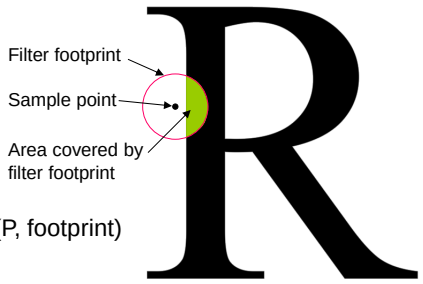
- Adaptively Sampled Distance Fields



Special Cell ADF

Distance Fields for Type

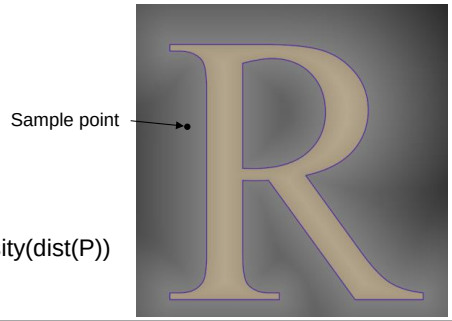
- Anti-aliasing
  - Traditional anti-aliasing estimates pixel intensity according to coverage
    - How much of the filter footprint is covered by the object
    - Requires expensive analytic filter or many supersamples to approximate the coverage at each sample point



Intensity = coverage(P, footprint)

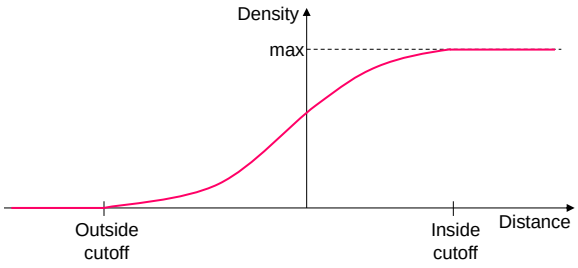
Distance Fields for Type

- Anti-aliasing
  - Distance-based anti-aliasing
    - Reconstruct distance at sample point and map to density
    - Requires only a single sample per sample point
    - Provides quality at least as good as exact analytic coverage-based methods and superior to supersampling



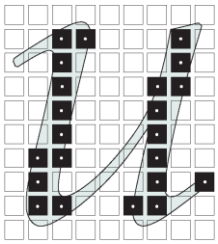
Distance Fields for Type

- Anti-aliasing
  - Distance to density mapping

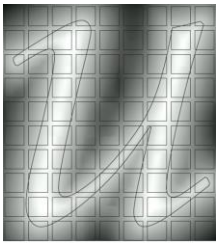


Distance Fields for Type

- Distance-based anti-aliasing
  - Quality is good because the distance field varies smoothly
    - Single distance sample captures proximity to the edge
    - Single coverage sample can miss the edge even when close



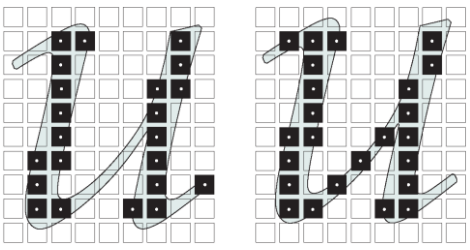
Sampled outline of a Palentino ‘u’



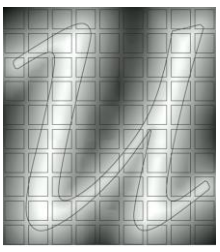
Distance field of a Palentino ‘u’

Distance Fields for Type

- Distance-based anti-aliasing
  - Distance field of a moving glyph doesn’t change much from frame to frame
    - Distance field provides good quality temporal anti-aliasing
    - Coverage via sampling can have discrete jumps in coverage between frames (blinking and popping)



Sampled values can vary significantly between frames when outlines are animated

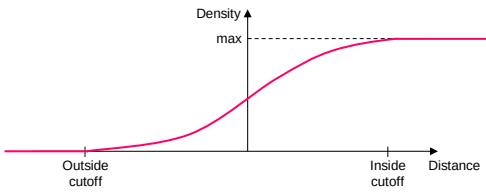


The distance field varies smoothly between frames

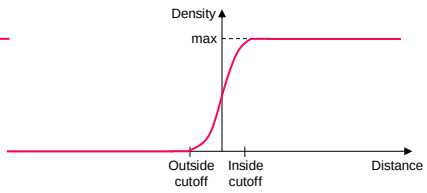
Distance Fields for Type

- Distance-based anti-aliasing
  - Continuous stroke modulation (CSM)
    - Vary the outside and inside cutoff values when mapping distance to density
    - Continuous variation of
      - Stroke weight (stroke thickness)
      - Edge sharpness

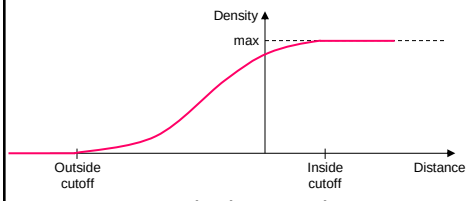
Continuous Stroke Modulation



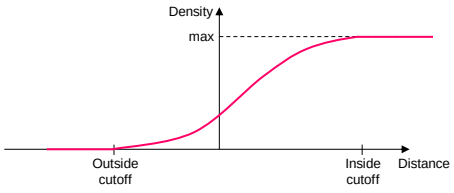
Softer edge



Sharper edge



Thicker stroke



Thinner stroke

Continuous Stroke Modulation

Continuous Stroke Modulation

Distance Fields for Type

- Good anti-aliasing is not sufficient for good quality type
  - Using only anti-aliasing results in
    - Fuzzy edges
    - Non-uniform stroke weights
    - Inconsistent cap-heights, x-heights, and other characteristic distances
  - Type looks best when
    - Edges have high contrast
      - Strong vertical stems and horizontal bars must be aligned to the pixel grid
    - Stroke weights are uniform
    - Characteristic distances are consistent

Distance Fields for Type

- Good anti-aliasing is not sufficient for good quality type
  - Saffron
    - Detects strong vertical and horizontal glyph edges automatically
    - Uses these edges to define cap-heights, x-heights, and other characteristic distances
    - Aligns these characteristic distances to the pixel grid during rendering

Distance Fields for Type

- Edge detection
  - Fortunately, edge detection is straightforward for distance fields
    - Render an image of the distance field from the ADF
    - Apply standard image processing techniques to detect vertical and horizontal edges in the distance field
      - For left edges:
        - » Filter image to detect left-to-right crossings of the distance field's zero-valued iso-surface
        - » Sum filtered values along columns of the distance image
        - » Strong left edges occur in columns where the summed values are large
  - Detecting edges from outlines is more complicated …

Distance Fields for Type

- Using the Distance Field Gradient
  - The gradient is used to detect stroke centerlines and the distance field is used to determine stroke thickness
  - Strokes can be thickened during rendering

Distance field of glyph

At stroke centers
- Discontinuous gradient
- Positive distance

Weak thin strokes

Corrected glyph after thin strokes were detected and strengthened

Distance Fields for Type

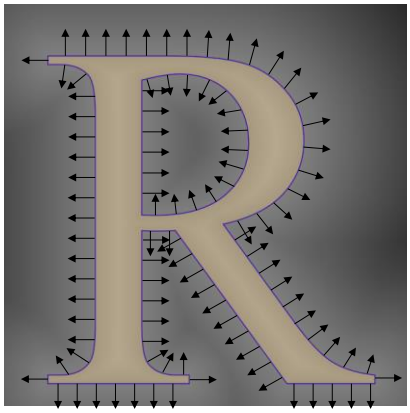
- Using the Distance Field Gradient
  - Detecting and modifying specific edges
    - e.g., Left, right, bottom, top

Weak left edges

Corrected glyph after left edges were detected and strengthened

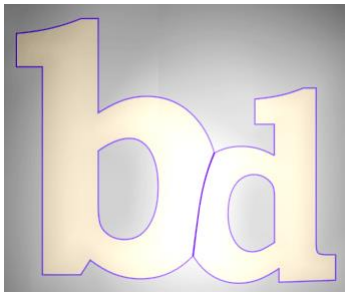
Distance Fields for Type

- Using the Distance Field Gradient
  - Use the gradient direction to
    - Detect specific offset regions (e.g., left, right, top, bottom)
    - Render directed shadows
    - Render motion blur



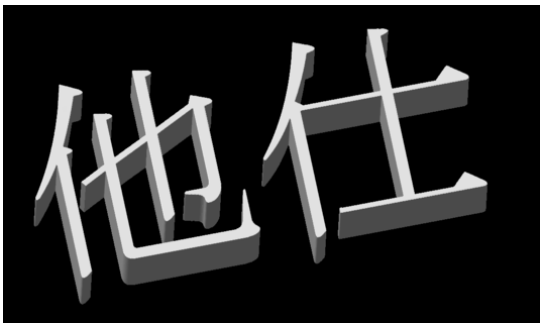
Distance Fields for Type

- Using the Distance Field
  - Animation effects
    - e.g., soft body deformation via implicit blends
    - e.g., particle systems



Distance Fields for Type

- Using the Distance Field
  - 3D type



Current Saffron Type System

Toolkit Approach

- The API is structured as a toolkit with 4 major functional blocks
  - Conversion of glyph outlines to ADFs
  - Rendering ADFs as density images
  - Determining alignment zones directly from ADFs, thereby enabling grid fitting during ADF rendering
  - A dual caching system for ADFs and density images

Additional Tools

- Validation tools
  - View ADF cells
  - View alignment zones
- Validation scripts for QA
  - Test all aspects of the system, including timing and memory requirements
- Example code demonstrating API usage
  - Fundamentals (e.g., ADF generation and rendering)
  - Grid fitting to the pixel and sub-pixel
  - Preserving metrics when grid fitting

Additional Tools

- Additional functionality
  - e.g., grid fitting for glyphs typeset at 90, 180, 270 degrees for all display modes (CRT, RGBv, BGRv, etc.)

Basic Data Flow and Processing

1. Initialize the system
2. Preprocess the required glyphs to determine their alignment zones
   - Initialize alignment zone detection for specified typeface
   - Detect the alignment zones for all required glyphs in the specified typeface
   - Terminate alignment zone detection for the specified typeface
3. Create an ADF cache and a density image cache

Basic Data Flow and Processing

4. Typeset and render each glyph
   - Query the density image cache for the rendered glyph
   - Upon a cache miss
     - Query the ADF cache for the glyph's ADF
     - Upon a cache miss
       - Generate the glyph's ADF from its outlines
       - Insert the glyph's ADF into the ADF cache
     - Render the glyph's ADF into a density image using the glyph's alignment zones for grid fitting
     - Insert the density image into the density image cache
   - Composite the glyph's density image into the display buffer
   - Advance the pen position using adjustments determined during grid fitting

Basic Data Flow and Processing

5. Destroy the caches
6. Terminate the system

API Structure

- Memory Allocation
- Fundamental Data Types
- Initialization and Termination of the System
- Glyph Outlines
- ADF Generation from Glyph Outlines
- Alignment Zone Detection
- Density Images
- Rendering
- Dual Caching System

Generating ADFs

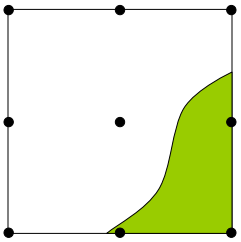
- ADF representation
  - Spatial hierarchy of cells
    - Current system uses a quadtree hierarchy of square cells
    - Each cell contains a set of sampled distance values and a reconstruction method
    - There are various cell representations and corresponding reconstruction methods

Generating ADFs

- Top down generation
  - Start with root cell
  - Recursively subdivide cells into smaller cells until either
    - A maximum generation level is reached
    - The difference between the reconstructed distance field in a cell and the true distance field in the cell is less than a specified error tolerance

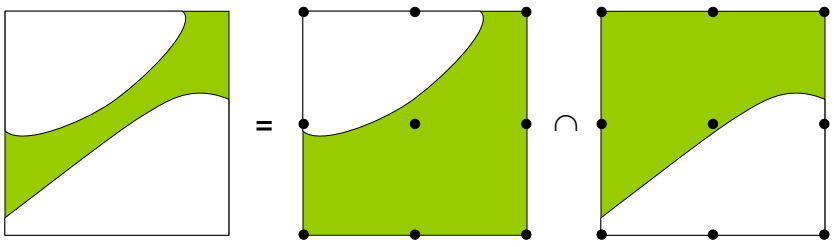
Special Cell ADFs

- Use a variety of cell types
  - Intermediate cells of the quadtree
    - Contain offsets to child cells and no distance values
  - Single section cells (cells closest to a single smooth section of the outline)
    - Contain 9 distance values (16 bit S.15 format) and use a biquadratic reconstruction method



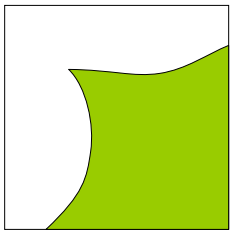
Special Cells

  - Two section cells (cells closest to two smooth sections)
    - Represent regions near thin stems and arcs
      - Distance field closest to two different sections of the outline
    - Contain 9 distance values for each section (18 total)
    - Reconstruct the distance field of each section separately using biquadratic interpolation
    - Reconstruct the distance field for the two section cell using CSG intersection of the two distance fields

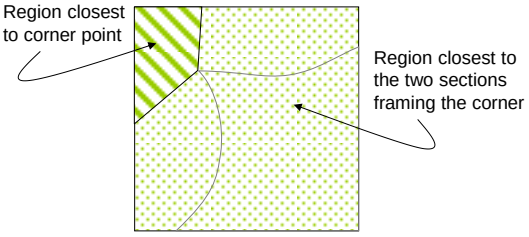


Special Cells

- Corner cells (cells closest to a corner of the outline)
  - Represent regions near corners of the glyph
    - Distance field closest to corners
  - Contain 9 distance values for each section framing the corner and 6 values representing corner data (24 total)
  - Use the corner data to determine whether a sample point is closest to the corner point or to the two sections
  - Reconstruction method depends on sample point location

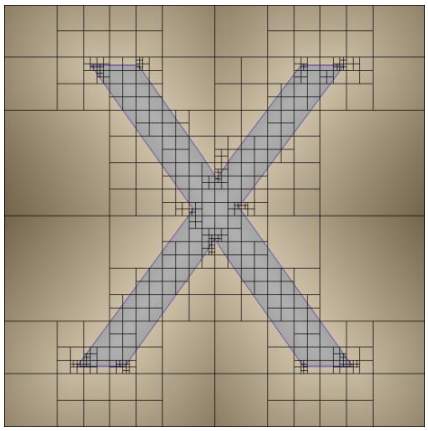


Cell containing a corner

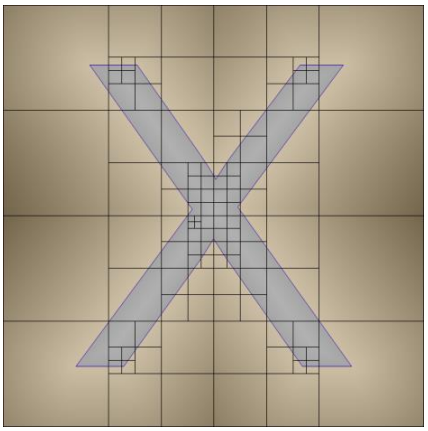


Distance reconstruction regions

Special Cells



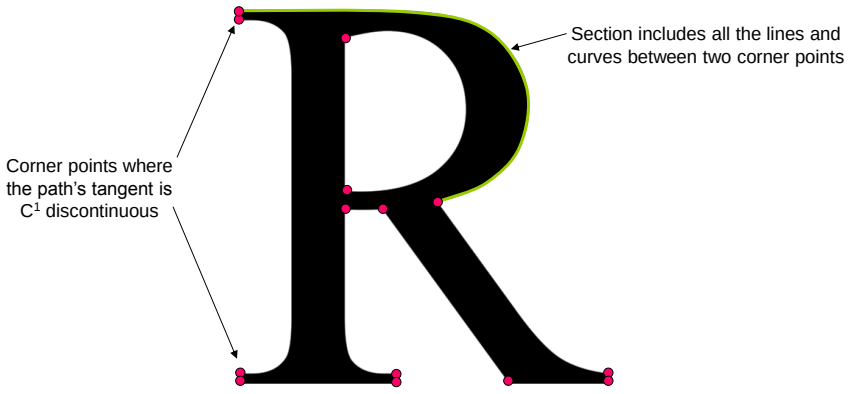
Biquadratic ADF



Special Cell ADF

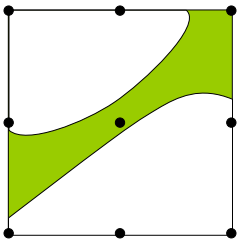
Special Cell ADF Generation

- Pre-process outlines to partition the path into sections and corners



Special Cell ADF Generation

- Use top down generation, starting with root cell
  - Recursively subdivide cells into smaller cells until either
    - A maximum generation level is reached
    - Cell distance field represents true distance field within a specified error tolerance
      - During distance computation, annotate distances according to their closest section or corner
      - For each cell, section and corner annotation is used to determine cell type (corner, one section, two section)



Alignment Zones

- Use of alignment zones
  - Determined directly from the ADF
  - Define strong vertical and horizontal edges of the glyph and characteristic distances of a typeface
    - Baseline to x-height and baseline to cap-height distances
    - Spacing between repeated vertical stems
    - Left edge of a glyph
  - Used to build the appropriate transformation needed for grid fitting each ADF to the pixel grid or pixel component grid

Alignment Zones

- Determining alignment zones
  - Three step process:
    - Initialize alignment zone detection for a specified typeface
      - Determine x-height and cap-height from particular glyphs of the typeface (i.e., 'z' and 'Z')
    - Detect alignment zones for each required glyph in the specified typeface
      - Use table to specify expected alignment zones for each glyph in Latin fonts
    - Terminate alignment zone detection for the specified typeface
  - Can be done once (offline) and stored with the glyph's ADFPath
    - Requires at most 64 bits per glyph

Rendering

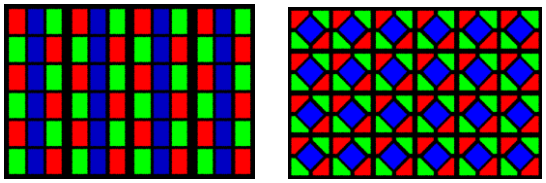
- Density images
  - Glyphs are rendered into density images
    - For each pixel or pixel component in the density image
      - Transform the pixel location to a sample point in the ADF
      - Reconstruct the distance field at the sample point
      - Map the reconstructed distance to a density value
  - Density images can be used by the application to blend a foreground color (e.g., a text color) with a background color or a background image to produce a colored image of the rendered glyph

Rendering

- Rendering Steps
  - Set the render state (e.g., point size, display mode …)
  - Call the render setup function to determine data used for rendering and typesetting
    - Render setup determines a matrix transformation that places the glyph in the display window
    - The matrix transformation represents any affine transformation, i.e., translation, rotation, scale, shear, reflection, etc.
  - Create a density image of the required size
  - Render the glyph into the density image

LCDs and Alternative Pixel Patterns

- Coverage-based anti-aliasing requires multiple samples per pixel component for supersampling
  - TrueType takes advantage of the RGB-striped LCD pixel pattern to reuse samples
  - Each different striping patterns requires different strategy to reuse samples (not hard, just inconvenient)
  - Irregular patterns can't reuse samples as easily



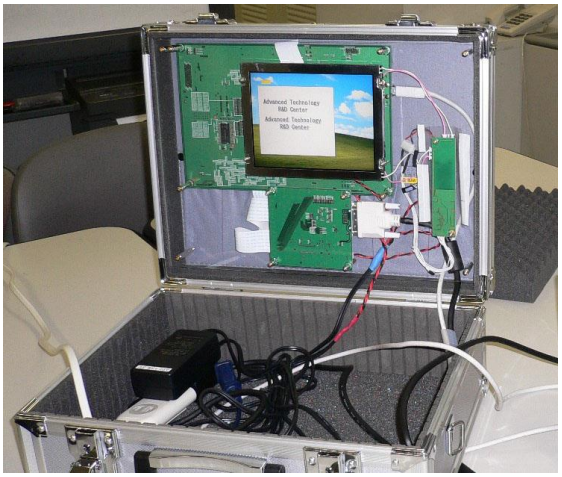
Two alternative pixel patterns used by Clairvoyante

LCDs and Alternative Pixel Patterns

- Distance-based anti-aliasing only requires one sample per pixel component
  - Different patterns do not affect efficiency
    - A simple modification to the transform from image coordinates to ADF coordinates is required
  - Library supports RGBv, BGRv, RGBh, BGRh
  - Effectiveness for alternative pixel patterns already demonstrated
  - CSM makes it easy to adjust stroke weight and contrast for new pixel patterns and to compensate for color fringing

Hardware Implementation

- Saffron – V0 (Biquadratic cells only)



Hardware Implementation

- Saffron – V0 (Biquadratic cells only)
  - 156K gates on an FPGA
    - Includes memory controllers for accessing fonts and frame buffers
    - Includes compositing (triple buffered)
  - Renders 100,000 glyphs per second at 100 MHz for 10 point, 96 dpi, LCD rendering (3 samples per pixel)
  - Current implementation has a single component pipeline
  - Designed for a 3 component pipeline (3x as fast at a cost of 20K-30K additional gates per pipeline)

Patents

- Detail-Directed Hierarchical Distance Fields: **U.S. Patent No. 6,396,492**
- Method for Antialiasing an Object Represented as a Two-Dimensional Distance Field in Image-Order
- Method for Antialiasing an Object Represented as a Two-Dimensional Distance Field in Object-Order: **Allowed**
- Method for Animating Two-Dimensional Objects
- Method for Converting Two-Dimensional Objects to Distance Fields
- Method for Converting a Two-Dimensional Distance Field to a Set of Boundary Descriptors
- Method for Converting Two-Dimensional Pen Strokes to Distance Fields
- Method for Generating a Two-Dimensional Distance Field within a Cell Associated with a Corner of a Two-Dimensional Object

Patents

- Method and Apparatus for Antialiasing a Set of Objects Represented as a Set of Two-Dimensional Distance Fields in Image-Order
- Method and Apparatus for Antialiasing a Set of Objects Represented as a Set of Two-Dimensional Distance Fields in Object-Order: **Allowed**
- Method for Generating a Composite Glyph and Rendering a Region of the Composite Glyph in Image-Order
- Method for Generating a Composite Glyph and Rendering a Region of the Composite Glyph in Object-Order
- Methods for Generating an Adaptively Sampled Distance Field of an Object with Specialized Cells
- Method and Apparatus for Rendering Cell-Based Distance Fields Using Texture Mapping: **Allowed**
- Method for Typesetting a Set of Glyphs Represented as a Set of Two-Dimensional Distance Fields

Patents

- Method, Apparatus, and System for Rendering Using a Progressive Cache
- Pipeline and Cache for Processing Data Progressively
- Modeling and Combining Multiple Graphics Objects
- System and Method for Generating Adaptively Sampled Distance Fields with Bounded Distance Trees
- Method for Traversing Quadtrees, Octrees, and N-Dimensional Bi-trees: **U.S. Patent No. 6,868,420**

Patents

- Distinction from prior art
  - Distance fields vs. images
  - Distance-based anti-aliasing vs. coverage-based anti-aliasing
  - Sampling discrete space vs. sampling continuous space
  - Derive alignment zones for grid fitting from distance field, not from hinted outlines
  - For typesetting, adjustments are made to advance widths using distance fields after iso-contour selection

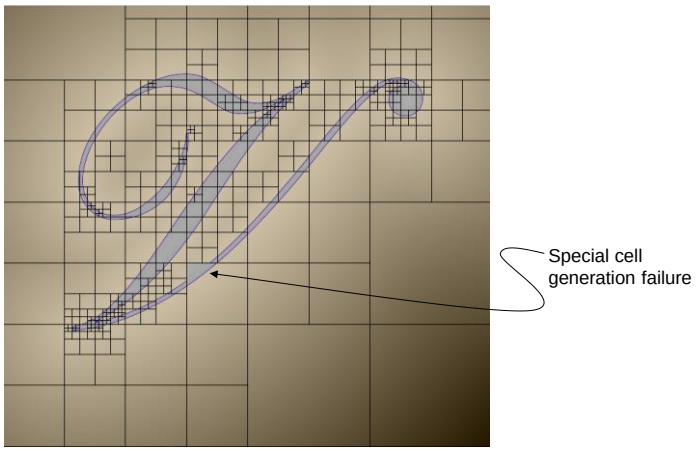
Limitations of Saffron – V1

Alignment Zones

- Current alignment zones include baseline, x-height or cap-height, and two vertical stems
- Some Latin glyphs would benefit from more alignment zones
  - Glyphs with 2 vertical or horizontal zones (e.g., e, F, m)
  - Glyphs with descenders (e.g., p, q, y)
- Need multiple alignment zones or new approach for best results in complex characters
  - e.g., CJK fonts

Issues and Unproven Features

- The predicate test used for special cell generation is not fail proof



Issues and Unproven Features

- Non-uniform scale in x and y distorts the filter region
- Some markets may require additional hint-level fine tuning to provide ultimate control
- Rendering bitmaps from ADFs untested/unproven
  - 2-bit rendering for eInk was successful

Memory and Processing

- Memory constrained devices
  - ADFs may be too big
  - 2D Distance field representation will always be bigger than 1D outline representation
- Processor constrained devices
  - Generation may be too slow

Saffron Roadmap

Saffron Roadmap

- Direct rendering
  - Solution for memory and processor constrained systems
- Hardware implementations
  - Special cell rendering
  - Direct rendering
  - Low power requirements?
  - Market potential? (Mitsubishi internal vs. broader markets)

Saffron Roadmap

- High quality small CJK
  - High quality stroke-based fonts with
    - Variable stroke weight
    - Various endcaps
      - Rounded, square, mitered, beveled, brushed
  - Alignment zones and grid fitting
  - Direct rendering
  - Automatic generation from outlines
  - Design tool

Saffron Roadmap

- Enhancements and extensions
  - CSM profile builder
  - Alignment zones for non-latin fonts
    - Hindu, Arabic
    - More general approach (e.g., multiple alignment zones) for CJK
- GPU implementation of direct rendering
- Extend library to support alternative pixel patterns
- Performance and size enhancements for Saffron – V1

Direct Rendering

Direct Rendering

- See separate presentation

Advantages Over Traditional Approaches

Advantages

- Sub-pixel rendering patent issues
  - Saffron's IP is comprehensive and patently-distinct from Microsoft's ClearType
  - Traditional, coverage-based anti-aliasing approaches encroach on Microsoft's extensive patent portfolio

Advantages

- Hinting
  - Saffron achieves high quality without hinting
  - Some markets are encumbered by hinting (e.g., Flash and SVG)
    - Markets with size constraints (bandwidth, disk space, memory space)
    - Dynamic environments where applications have no control over font quality (e.g., hints may not be available)
  - Hinting new fonts is labor intensive and expensive
- Sub-pixel hinting
  - Hinting fonts for sub-pixel rendering is even worse
  - Legacy bug in hinting of certain glyphs (e.g., Arial 'x')

Advantages

- Hardware market
  - Computationally clean rendering pipeline is straightforward to implement in silicon
  - No special casing of type vs. graphics
  - Hardware prototypes
    - Saffron – V0 hardware prototype exists
    - Hardware implementations of Saffron – V1 and direct rendering approach are under development

Advantages

- GPU implementation is straightforward
  - Potential markets
    - Desktop/laptop market (e.g., GPU-centric Longhorn)
    - Console and handheld game market
    - Embedded systems market
      - GPUs will be increasingly incorporated into next generation devices
      - GPUs on today's devices could support Saffron using Gouraud shaded texture mapped triangles

Advantages

- High quality stroke-based CJK
  - Can provide higher quality stroke-based fonts with a small memory footprint
  - Can provide much higher quality than current market offerings
    - Variable stroke weight
    - Designed endcaps

Advantages

- Alternative pixel patterns and displays
  - Saffron adapts easily to different sampling patterns
    - Saffron requires only one sample per pixel component for anti-aliasing
    - Saffron doesn't depend on the topology of the pattern
      - Kodak OLEDs
      - ClairVoyante Pentile
  - Continuous Stroke Modulation (CSM) provides OEMS with unprecedented control to tune type for their specific displays

Advantages

- Special effects
  - CSM control
  - Inexpensive thin and bold typefaces
  - Temporal anti-aliasing
  - Motion blur
  - Using the distance field for special effects such as soft-body deformation and particle systems
  - 3D type