

Saffron 3.1

Continuous Stroke Modulation

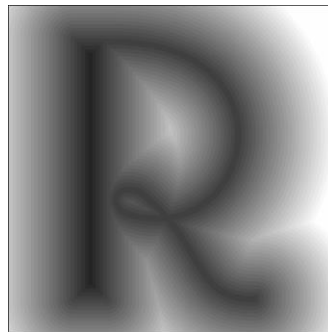
January 26, 2007

Distance Fields

- What is a distance field?
 - 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

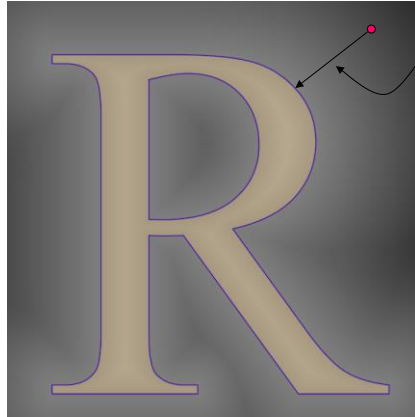


Shape



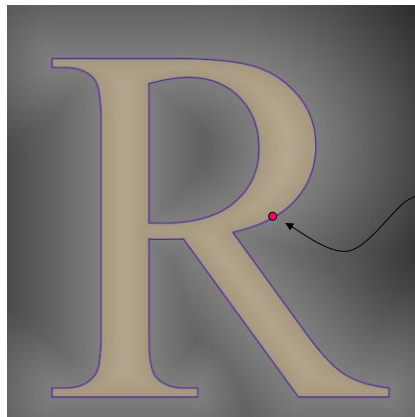
Shape's distance field

Example #1



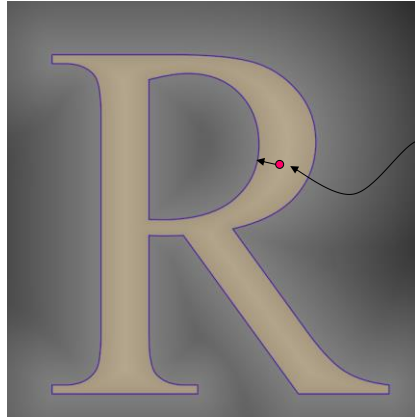
Distance to closest point on outline (signed distance is negative)

Example #2



Point lies on outline (signed distance is 0)

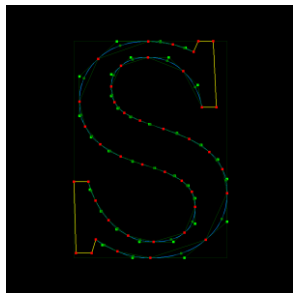
Example #3



Point lies inside outline
(signed distance is positive)

Glyph Rendering

- Rasterize glyph outline using image samples



Outline description



Image pixels

Glyph Rendering

- For each image sample
 - Compute distance from sample to closest point on outline

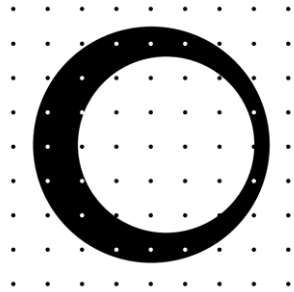
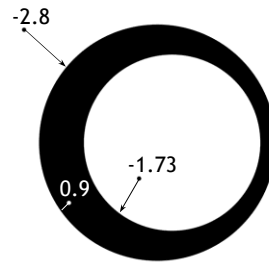


Image samples



Distances from image samples
to closest points on outline

Distances and Densities

- -2.8, -1.73, and 0.9 are signed [distance values](#)
- We want 8-bit [density values](#) in [0,255]
- How to map distance values to density values?

Continuous Stroke Modulation

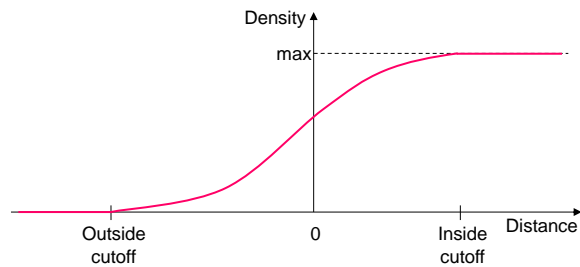
- Four CSM parameters:
 - Outside cutoff
 - Inside cutoff
 - Gamma
 - Color reduction

Continuous Stroke Modulation

- Four CSM parameters:
 - Outside cutoff
 - Inside cutoff
 - Gamma
 - Color reduction
- } *Most important*
- ← *Least important (always set to 1.0)*

Outside and Inside Cutoff

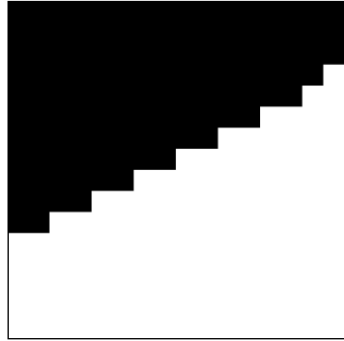
- Map distances $< OC$ to minimum density (0)
- Map distances $> IC$ to maximum density (255)
- Map distances in $[OC, IC]$ to densities in $[0, 255]$



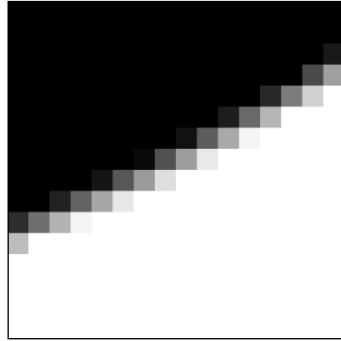
Distance-Based Antialiasing



Distance-Based Antialiasing



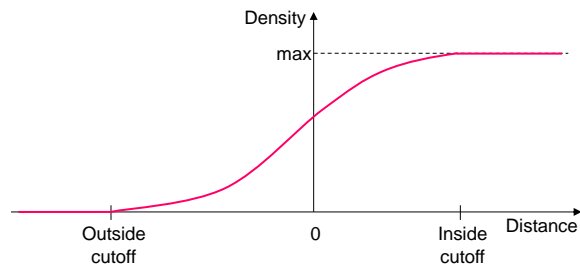
Outside cutoff equal to inside cutoff
(binary bitmap)



Outside cutoff = -0.8
Inside cutoff = +0.8
(grayscale antialiased bitmap)

Conventions

- Outside cutoff \leq inside cutoff
- Outside cutoff ≤ 0
- Inside cutoff ≥ 0



Choosing Cutoff Values

- What are the “correct” cutoff values?
- Competing goals:
 - Maximize clarity (sharpness, edge contrast)
 - Minimize aliasing (jaggies, sampling artifacts)

Appearance Is Subjective

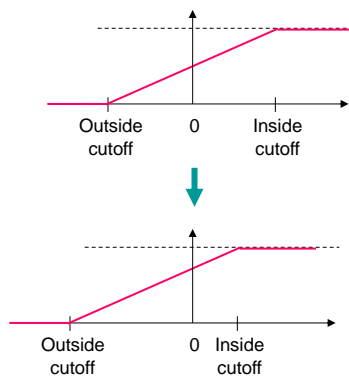
- Several factors:
 - Font design
 - Display resolution
 - Pixels per em
 - User preferences

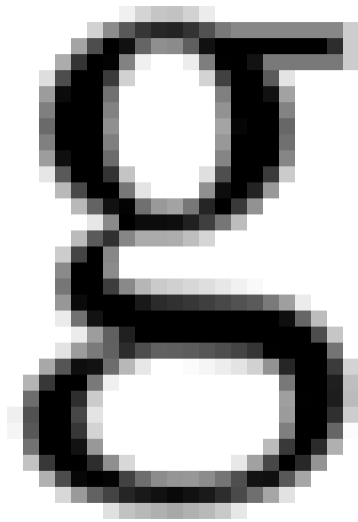
Classical Antialiasing

- Coverage-based antialiasing options:
 - Filter choice (box, tent, Gaussian, etc.)
 - Filter position
 - Filter radius
 - Isotropic vs. anisotropic (oriented) filters
- Each parameter affects appearance

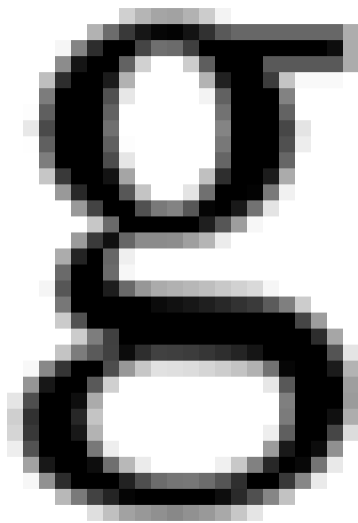
Example #1

- Decrease both outside and inside cutoff
 - Analogy: move filter position outwards
 - Makes glyph [thicker](#)





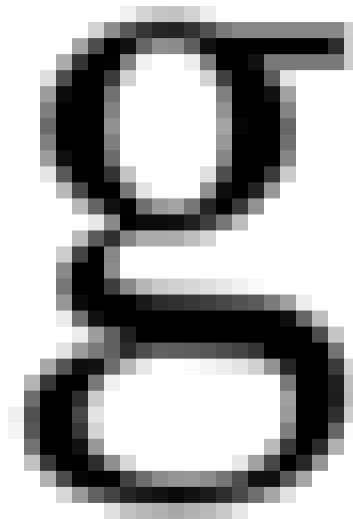
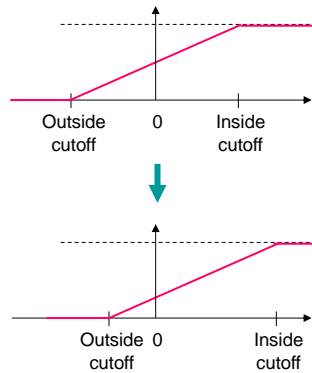
Normal



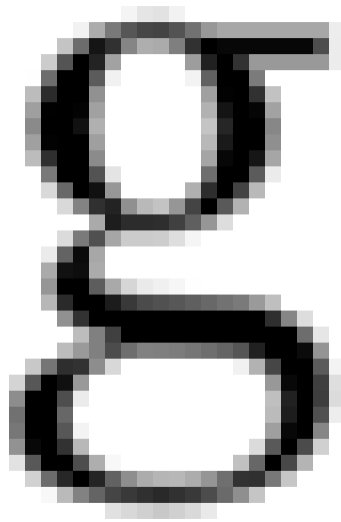
Thicker

Example #2

- Increase both outside and inside cutoff
 - Analogy: move filter position inwards
 - Makes glyph thinner



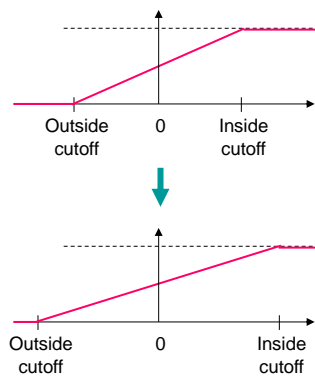
Normal

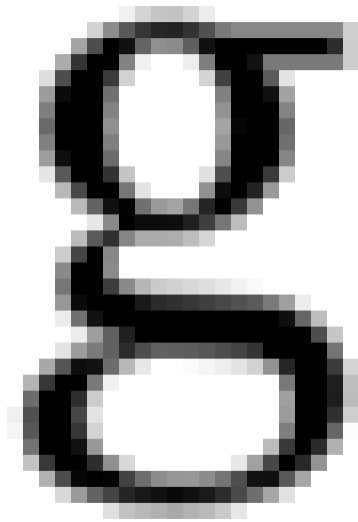


Thinner

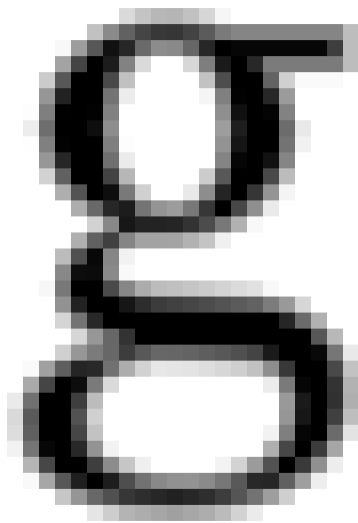
Example #3

- Decrease outside cutoff, increase inside cutoff
 - Analogy: increase filter width
 - Makes glyph edges softer





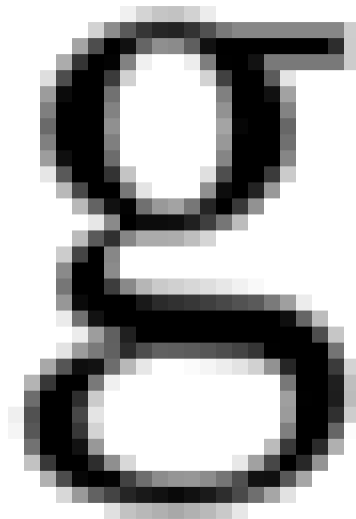
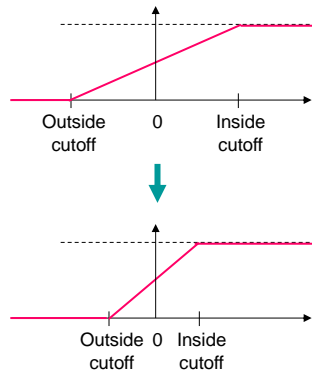
Normal



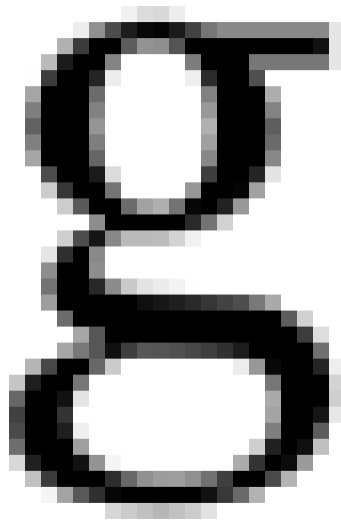
Softer

Example #4

- Increase outside cutoff, decrease inside cutoff
 - Analogy: decrease filter width
 - Makes glyph edges sharper



Normal



Sharper

Cutoff Value Summary

| Outside Cutoff | Inside Cutoff | Effect |
|----------------|---------------|---------|
| ↓ | ↓ | Thicker |
| ↑ | ↑ | Thinner |
| ↓ | ↑ | Softer |
| ↑ | ↓ | Sharper |

Continuous Stroke Modulation

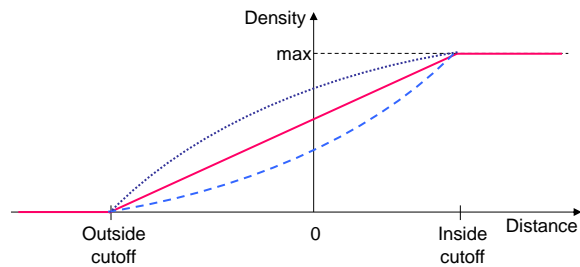
- Four CSM parameters:
 - Outside cutoff
 - Inside cutoff
 - Gamma
 - Color reduction

Gamma

- Executive summary: always set gamma to 1.0

Gamma

- Executive summary: always set gamma to 1.0
- Affects distribution curve of densities in $[0,255]$
- Gamma of 1.0 means linear distribution



Gamma

- Gamma used to help mitigate color fringing
 - Not needed for Saffron 3.0
 - $\text{Gamma} \neq 1.0$ means performance penalty

Color Reduction

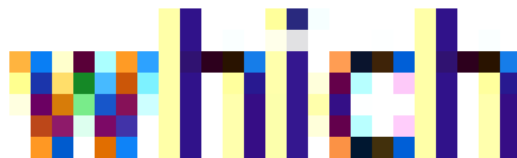
- Principle of LCD rendering
 - Use separate samples at red, green, blue component locations
 - Trade off color accuracy for spatial resolution

Color Reduction

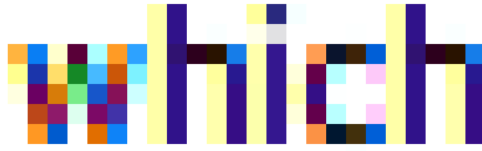
- Problem: LCD rendering produces color fringes

In written Japanese three sets of characters are used in addition to roman letters (romaji) which are used to abbreviate or highlight words. These sets of characters are Kanji, Hiragana, and Katakana. As shown on page 253, both of the syllabaries Hiragana and Katakana are developed from simplified Chinese characters. Hiragana are used for Japanese words that cannot be expressed in Kanji or have grammatical functions, while the use of Katakana is mainly restricted to foreign names and expressions adapted to Japanese so that they can be expressed with the syllables available.

There is a good reason for combined use of all three character sets, although any Japanese word can be written phonetically with one of the two syllabaries. Many words are pronounced the same (homophones), so Kanji are required in order to avoid ambiguity in the written language. Therefore, it is not likely that Kanji will become obsolete in the near future.



Color Reduction



Color Reduction Algorithm



Color Reduction Benefits

- Minimizes color fringing
- Makes tuning CSM parameters much easier

Color Reduction Control

- Two controls:
 - On/Off
 - Amount (0: minimal, 1: maximum)
- Recommendations:
 - Enable color reduction
 - Set amount to 0.5

Saffron API Settings

- ADFRenderAttrs data structure:
 - ADF_F32 insideCutoff;
 - ADF_F32 outsideCutoff;
 - ADF_F32 gamma;
 - ADF_U32 useColorReduction;
 - ADF_F32 colorReductionAmt;

Saffron CSM Tuner

CSM Real-World Scenario #1

- Limitation information available
 - Don't know which typeface will be used
 - Don't know the target display resolution
- Solution: Use a table of "general" CSM values
 - Table contains CSM values for common point sizes
 - Derive CSM values for other point sizes by interpolation
- We have developed a table that we can give you
 - Early work, but results are good

CSM Real-World Scenario #2

- Memory-constrained devices
 - Examples: PDAs, cell phones
- Zero-footprint “bold” and “thin” capability
 - Simply use “thicker” and “thinner” CSM parameter variations
 - No additional memory, no performance loss

CSM Real-World Scenario #3

- The more you know, the better
 - Closed environments (e.g., consoles, PDAs, cell phones)
 - Target device (e.g., specific cell phone model and screen)
 - Typeface (e.g., core system font on a specific cell phone)
- Can tune CSM parameters for best results