Non-Photorealistic Rendering

Pen-and-Ink Rendering and Related Techniques

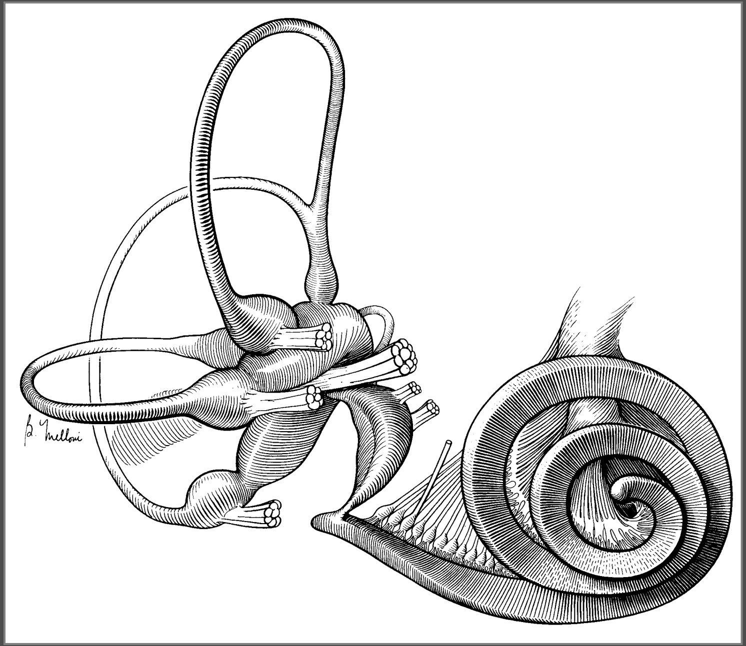
Tobias Isenberg

Overview

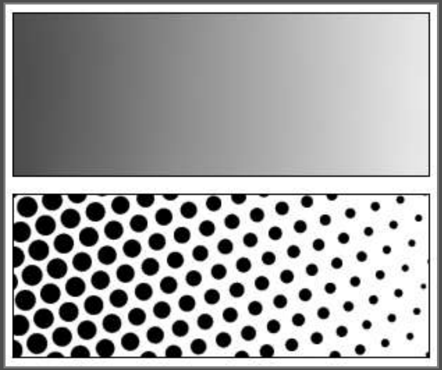
- introduction to pen-and-ink and similar dual-tone rendering techniques, halftoning, sparse lines
- binary shading
- artistic halftoning and techniques inspired by it
- rendering with points: stippling etc.
- rendering with lines: hatching etc.

Pen-and-Ink Rendering

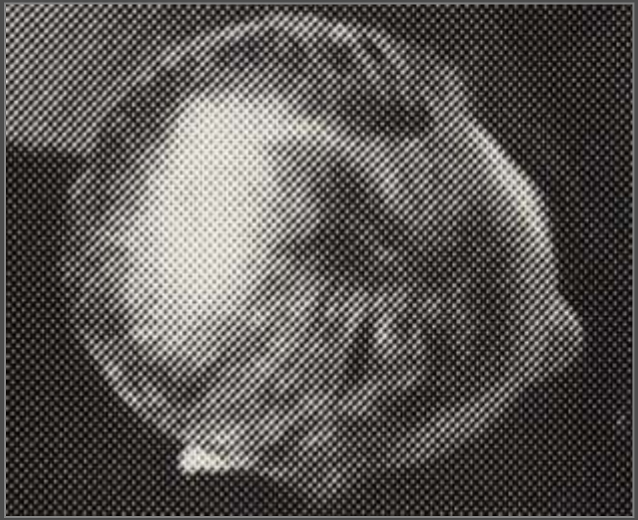
Introduction



Before we start: Halftoning

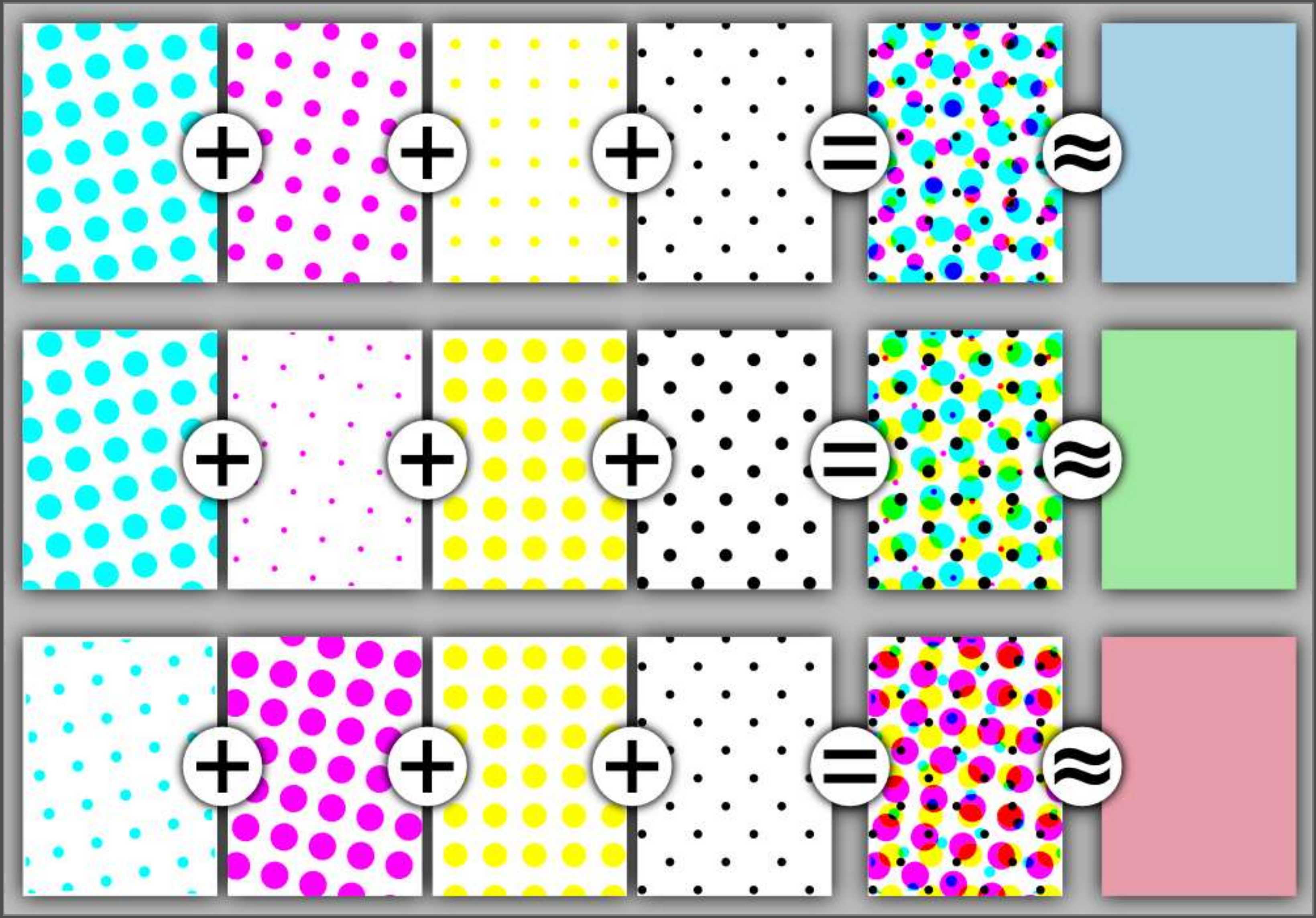


- print: can either put (spot) color at specific location or not put it there
- **task:** reproduce continuous tones of images in a discrete medium (print)
- use of automatic algorithms to translate tone levels to patterns of (spot) color; various techniques: ordered dithering, error diffusion (Floyd-Steinberg etc.), screening



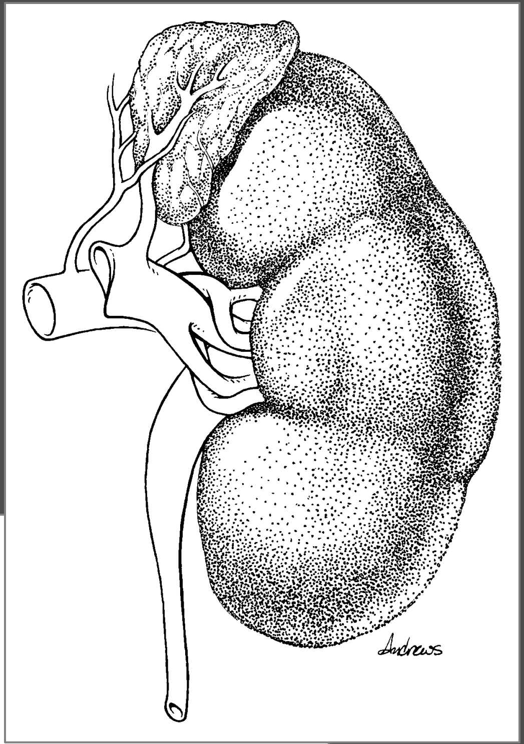
- **problem:** pattern often does not line up with the image features, thus do not capture the detail

Before we start: Halftoning – also in Color

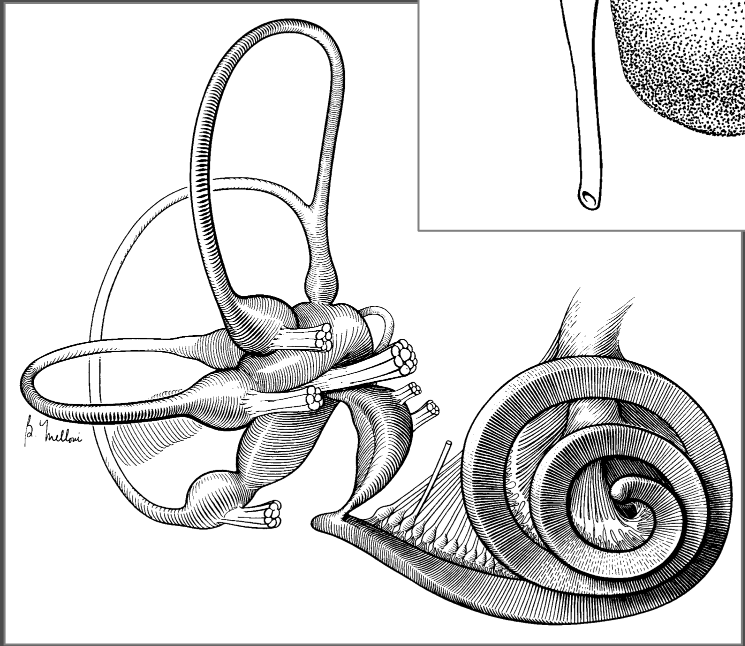


Pen-and-Ink Rendering Traditionally

- black-and-white technique that does not require halftoning
- translation of tonal changes into placement of dots and lines without irritating patterns or artifacts
- using pens to put ink marks onto paper
- two general methods:
  - stippling (dots)
  - hatching (lines)
- also combinations

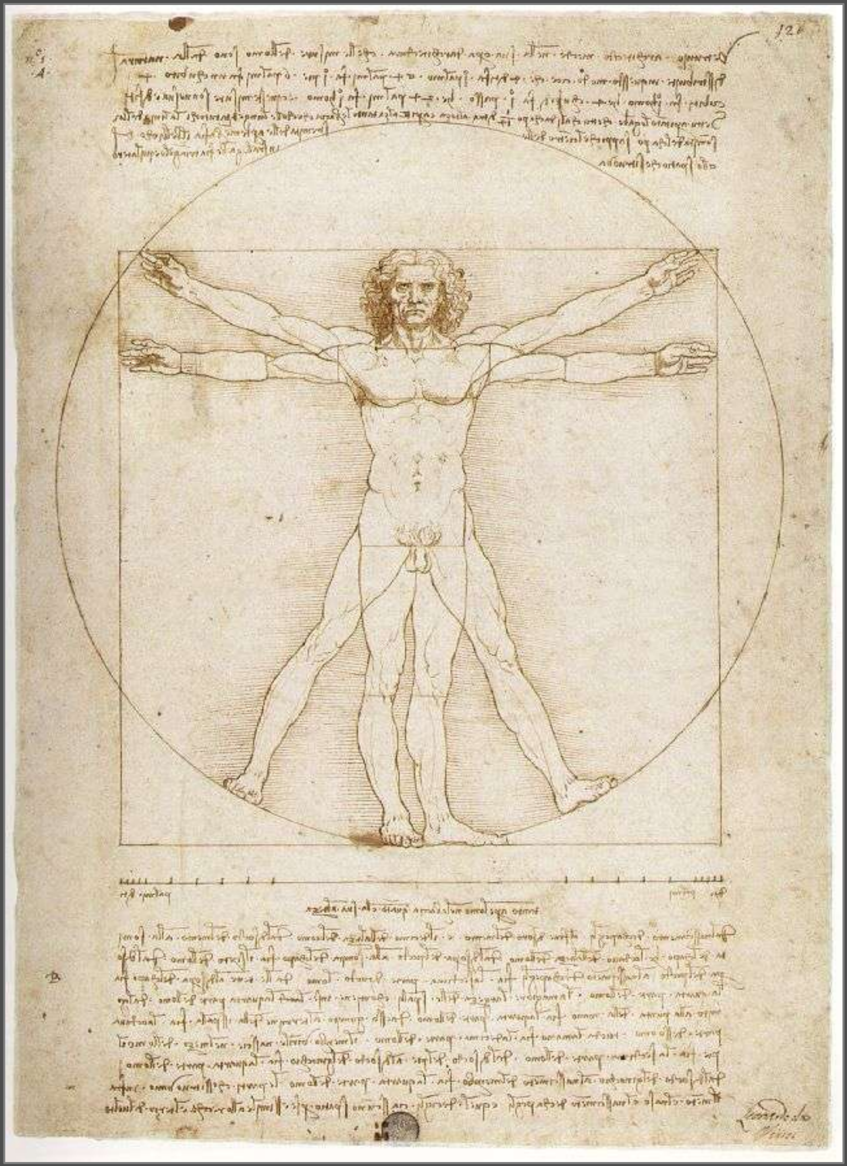


courtesy of B. Andrews



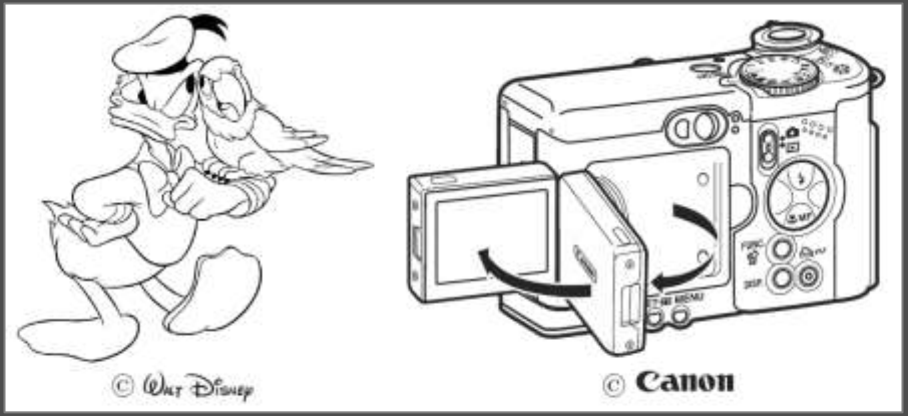
from Wood (1994)

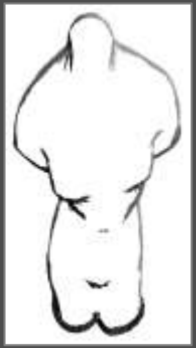
Related: Woodcuts, Copper Plates, etc.



Difference to Sparse Line Drawings

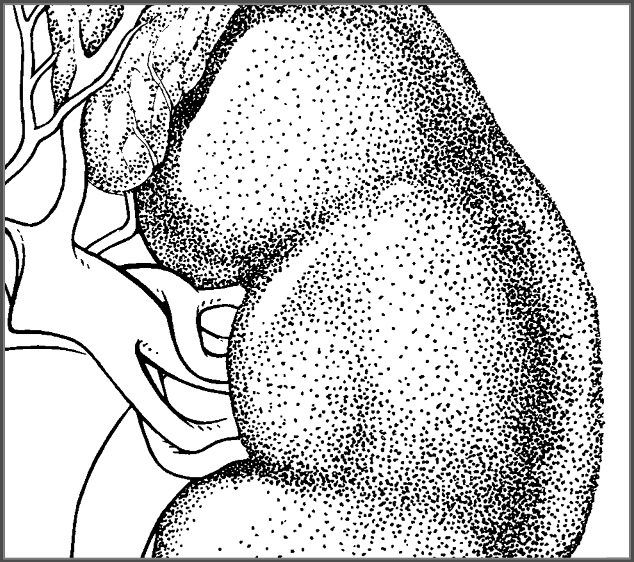
- sparse line drawings:
  - renditions restricted to a few linear features
  - silhouettes/contours/ outlines
  - feature lines: e.g., suggestive contours
- properties:
  - depict region borders
  - less information about shape
  - shading or material not depicted
  - use, e.g., in illustrations, sketches



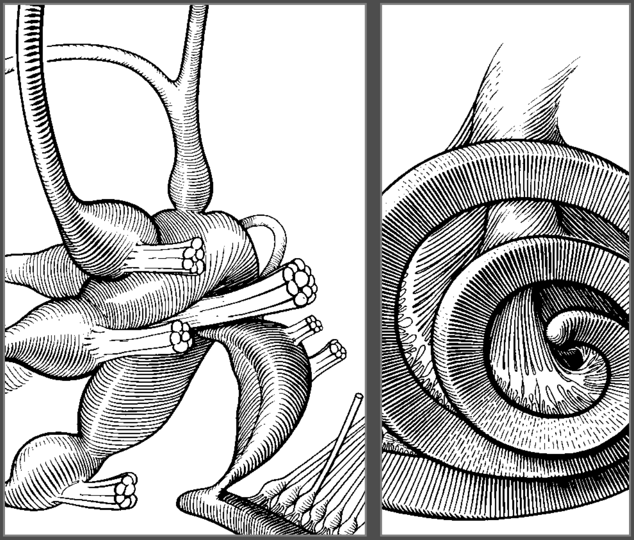


Difference to Sparse Line Drawings

- characteristics of pen-and-ink
  - showing not only region borders (maybe combined w/ sparse lines)
  - also showing shading
  - also showing surface shape
  - also illustrating material
  - abstraction and emphasis
- use of discrete elements
  - points (stippling)
  - lines (hatching, cross-hatching)
  - elements can intentionally be placed to line up with features



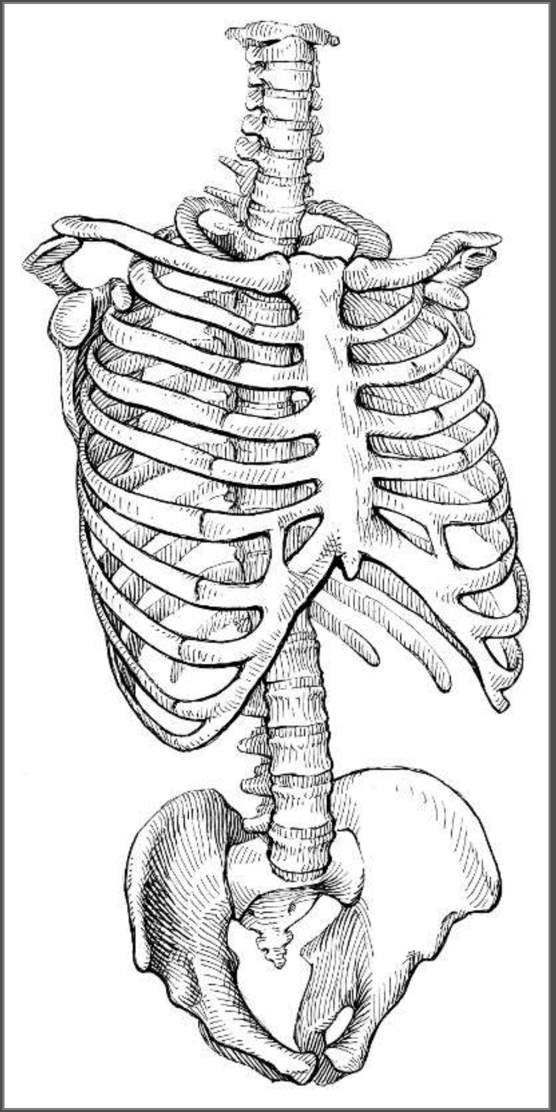
courtesy of B. Andrews



from Wood (1994)

Advantages of Pen-and-Ink for NPR

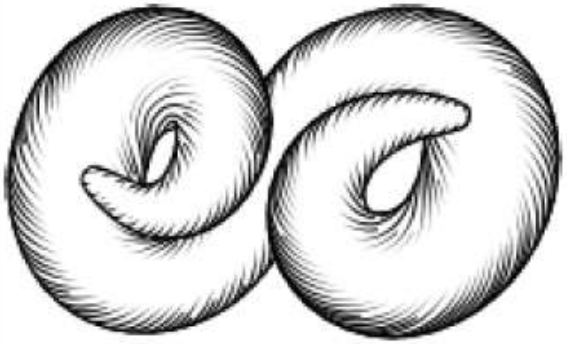
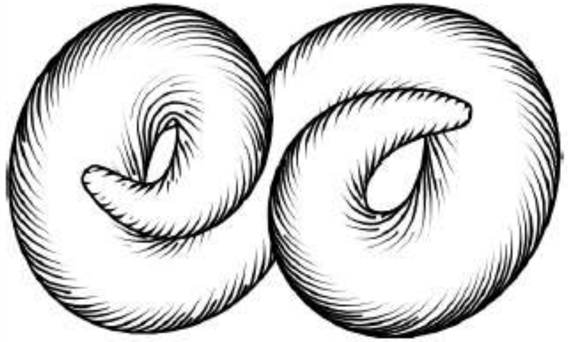
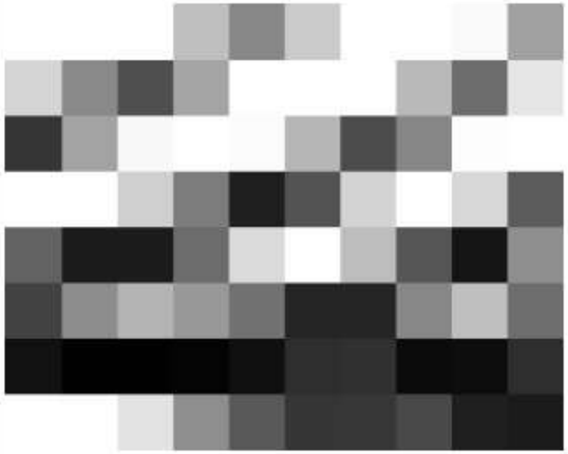
- “pen-and-ink” not in the strict technology-driven meaning
- does not require halftoning, already black-and-white
- thus images good for printing
- mark placement & stylization allow abstraction & emphasis
- precise depiction, good for illustration applications
- high-quality representation: vector or 1 bit pixel images



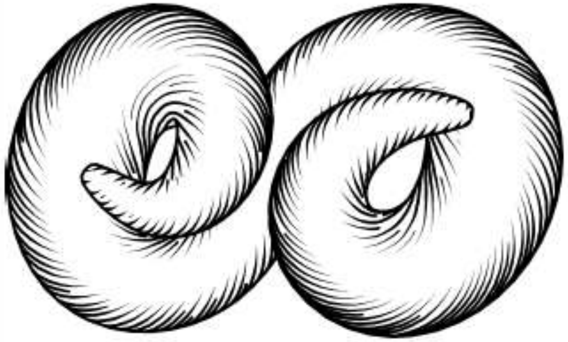
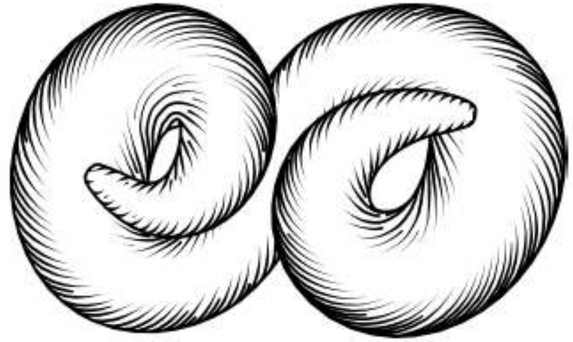
Side Note: Representing Pen-and-Ink

- two possibilities: vector and 1 bit pixel images
- vector images:
  - explicit representation of elements that make up the images (dots, areas, strokes, etc.)
  - interpreter renders these elements at display/print time (with anti-aliasing for tonal range displays)
- 1 bit pixel images:
  - discretized representation as binary images
  - use of higher resolution images (ppi ≈ 3× normal)
  - viewer sub-samples image at display time
  - printer re-scales image at print time (ppi resolution ideally ≈ dpi)

Side Note: Representing Pen-and-Ink

	197 x 119, 100 ppi, 8 bit	591 x 356, 300 ppi, 8 bit	vector graphic
uncompr.	23 kB	205 kB	417 kB
PNG/PDF	11 kB	52 kB	117 kB
image			
detail			

Side Note: Representing Pen-and-Ink

	2362 x 1423, 1200 dpi, 1 bit	4724 x 2846, 2400 dpi, 1 bit	vector graphic
uncompr.	410 kB	1,641 kB	417 kB
PNG/PDF	68 kB	166 kB	117 kB
image			
detail			

Side Note: Representing Pen-and-Ink

- vector images:
  - can represent image precisely
  - require rendering to determine elements to be displayed
  - dependent on vector “language” and, thus, interpreter
  - depending on represented detail can be large
  - cannot show detail that does not exist (scale issue)
- 1 bit pixel images:
  - can reach almost the same visual quality as vector images
  - can also represent scanned hand-drawn images
  - pixel images are quite universal
  - may be large if lot of detail (noise) is represented
  - depend even more on the given scale

Pen-and-Ink Rendering

Binary Shading

Halftoning-Inspired Techniques

Representing Shape, Illumination, & More

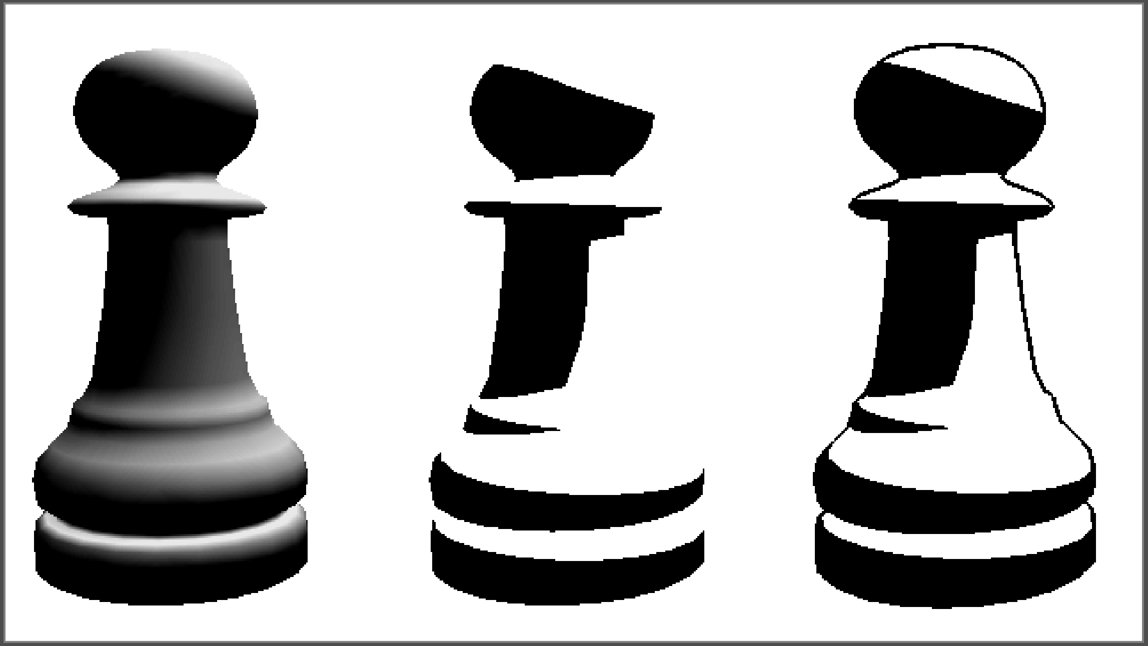
- in grayscale images: shape, illumination, material, etc. are represented through shades of color
- not possible in black-and-white, there are no shades
- different techniques
  - binary shading: large areas of black or white
  - halftoning: local pattern to simulate shades
  - stippling: the point/dot as primitive
  - hatching: the line as primitive
- thus inspired by artistic techniques (binary shading), technology (halftoning), and illustrative techniques (stippling, hatching)

Examples for Artistic B/W Images

images by Susan Throckmorton, Frank Miller, unknown, Felix Vallotton, and Jie Xu & Craig S. Kaplan

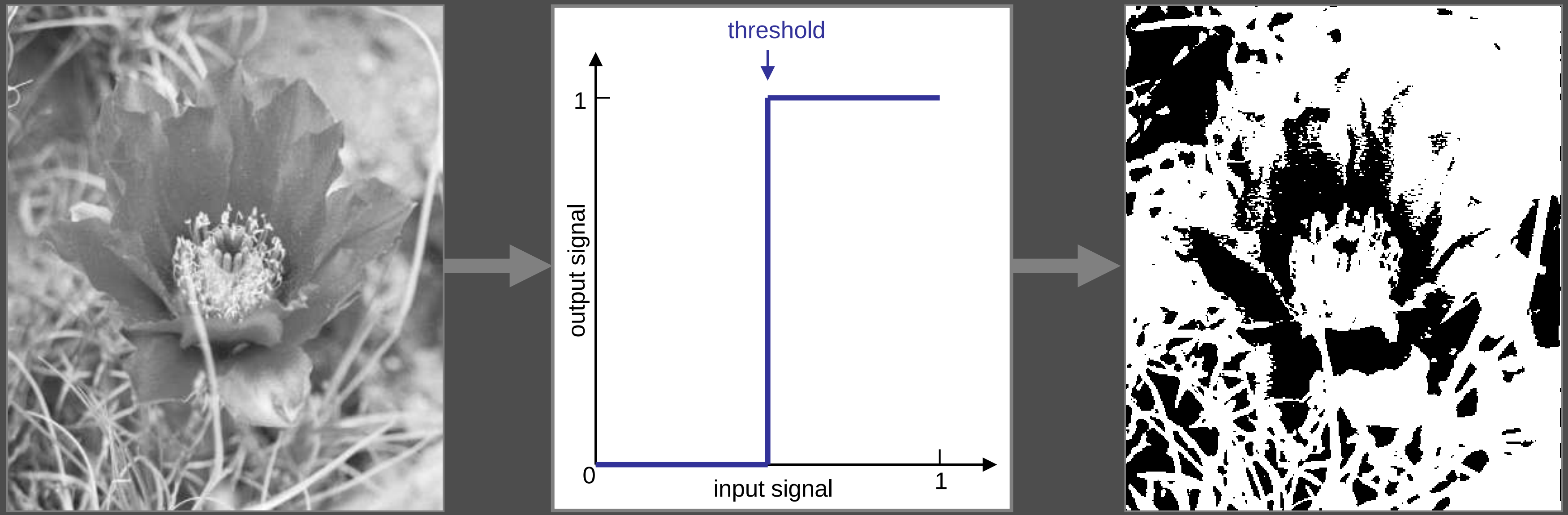
Binary Shading in Computer Graphics/NPR

- creating image with large areas of black or white
- typically based on grayscale image photograph or rendered image
- sometimes also additional elements (outlines)



Binary Shading: Thresholding

- straightforward binary shading technique: reduce the gray ramp to two colors using threshold
  - everything below the threshold is black
  - everything above the threshold is white

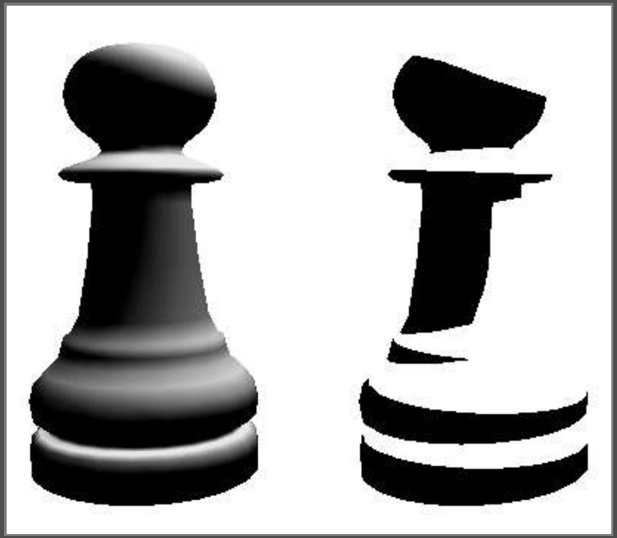


Binary Shading: Thresholding

Binary Shading: Thresholding

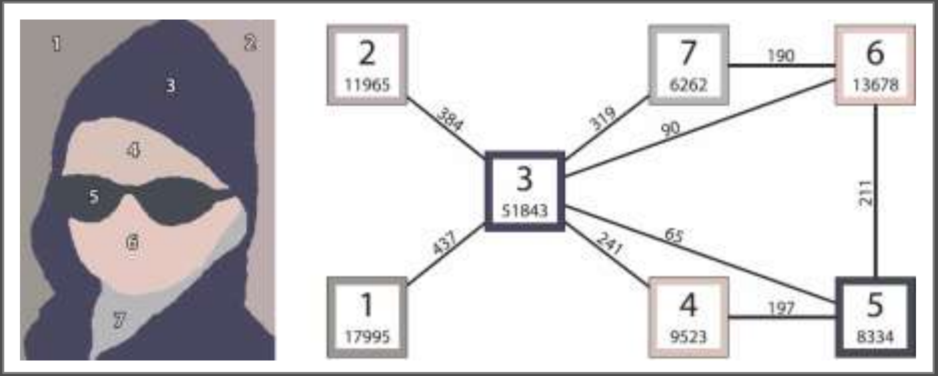
- real photographs:
  - often much noise in distribution of black/white regions
  - borders of regions often also noisy
- artificial images:
  - less noise in distribution of black/white regions
  - borders of regions often also less noisy





Binary Shading: Artistic Thresholding

- general approach:
  - image segmentation
  - color assignment & optimization
- image segmentation
  - identify N regions
  - represent as planar graph



- “color” assignment
  - 2^N possible region-color assignments
  - use of an objective function (based on color matching, area matching, boundary contrast, feature homogeneity)
  - genetic search to optimize the objective function
  - possible post-processing (adding edges between regions)

Binary Shading: Artistic Thresholding

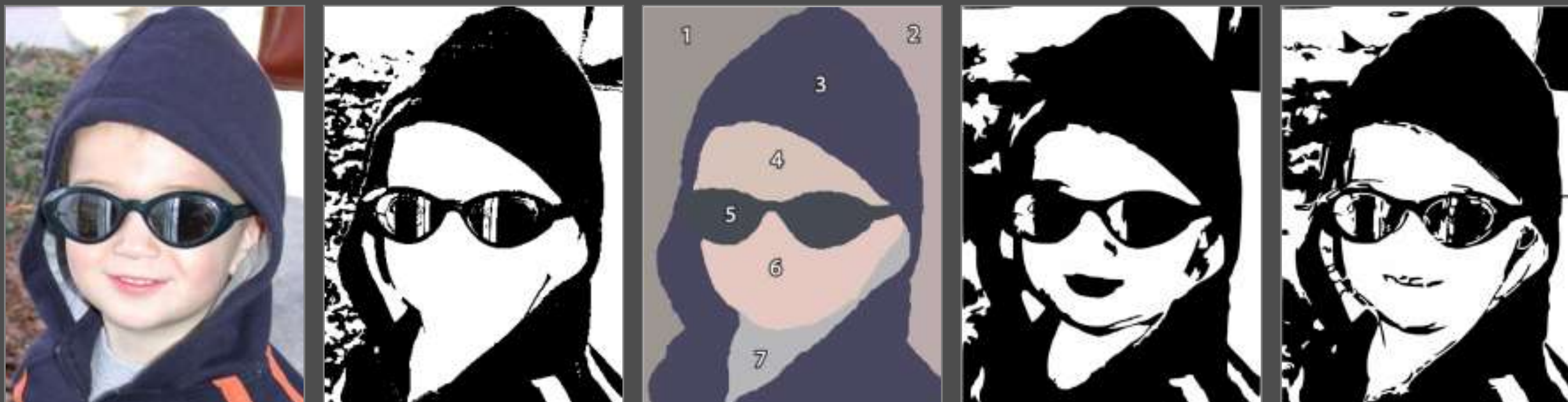
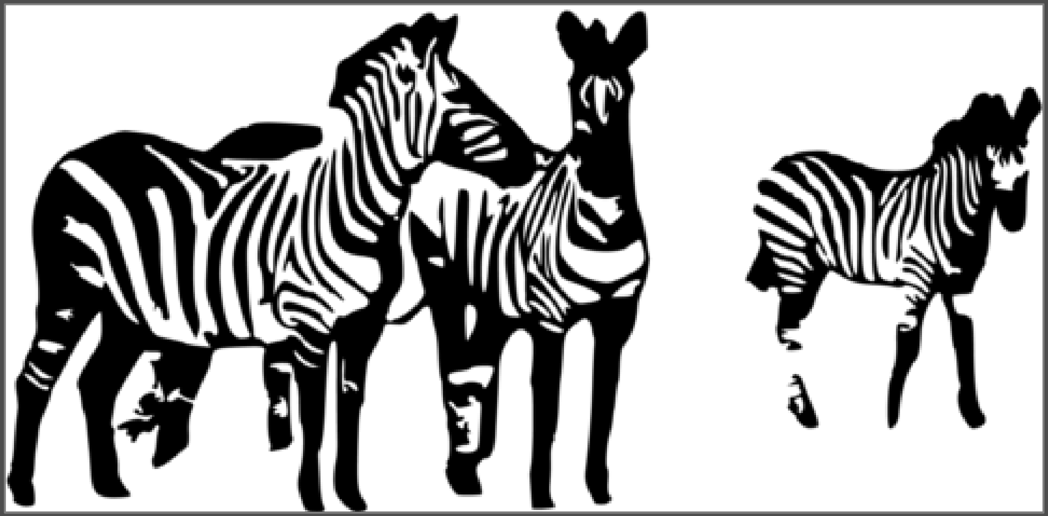


image | thresholding | segmentation | optimization | post-processing

Binary Shading: Artistic Thresholding

Binary Shading: Stylized B/W Images

- similar idea as before: cost minimization
  - combine local and global costs
  - use two layers: coarse base layer and detail layer
  - adaptive thresholding for detail layer: local classification into black, white, unknown

    → look at distance of pixel value from local average
  - combination of both layers: known from detail, rest from base
- finally: vectorize image

source image

simplified base layer

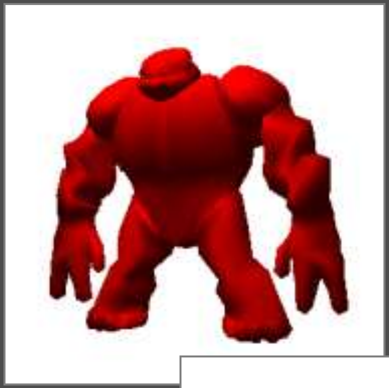
adaptive thresholding

combined result

Binary Shading: Stylized B/W Images

Binary Shading: Applications

- artistic expression
- clip art from your own images
- stencils and gobos (e.g., in theater lighting)



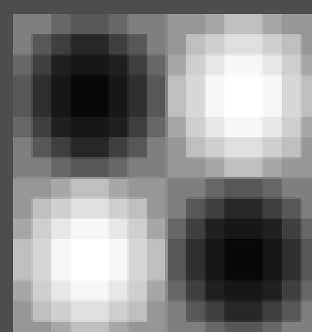
Beyond Binary Shading

- problems with binary shading
  - based on illuminated rendering or photograph
  - control of abstraction and emphasis difficult
  - local detail: result of processing, rather than intention
  - material depiction not considered
- new goals:
  - detail intentional
  - take individual marks into account

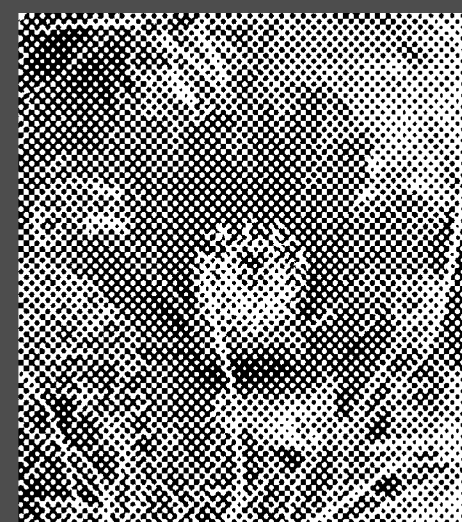
Techniques Inspired by Halftoning

- halftoning: assigning b/w based on a pattern
  - one halftoning technique: screening
  - screen: changing the threshold locally
  - screen tiled over input image (using modulo operations)
  - if input pixel darker than screen pixel: black output
  - otherwise: white output

×

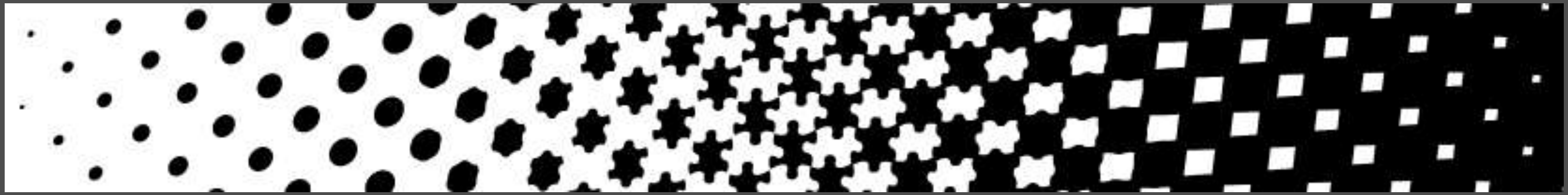


=



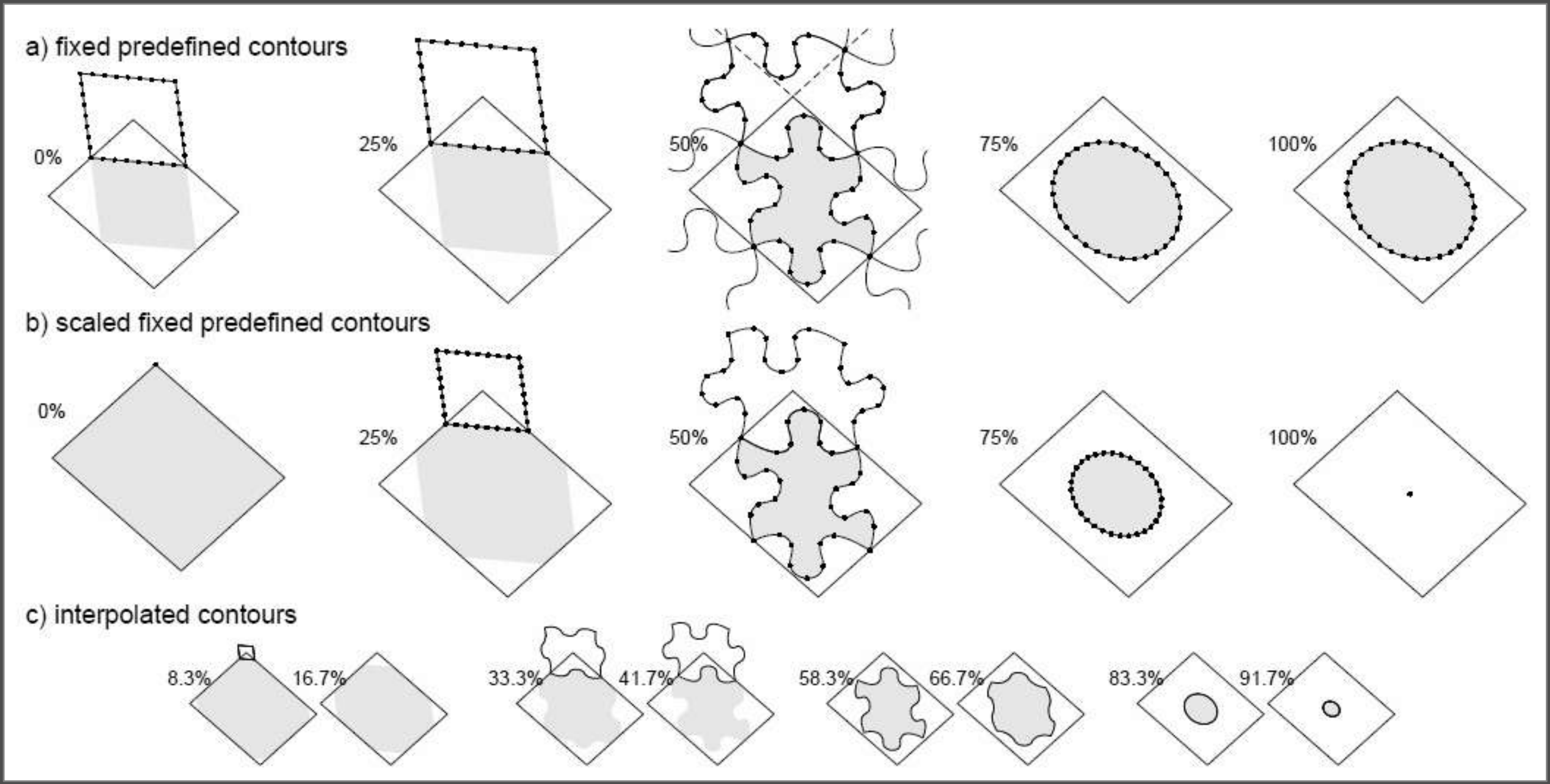
Artistic Screening

- typical constraints for dither screen
  - histogram of dither screen: constant value for all levels
  - homogeneous spatial distribution of gray levels
- artistic screening: creating screens with a meaning



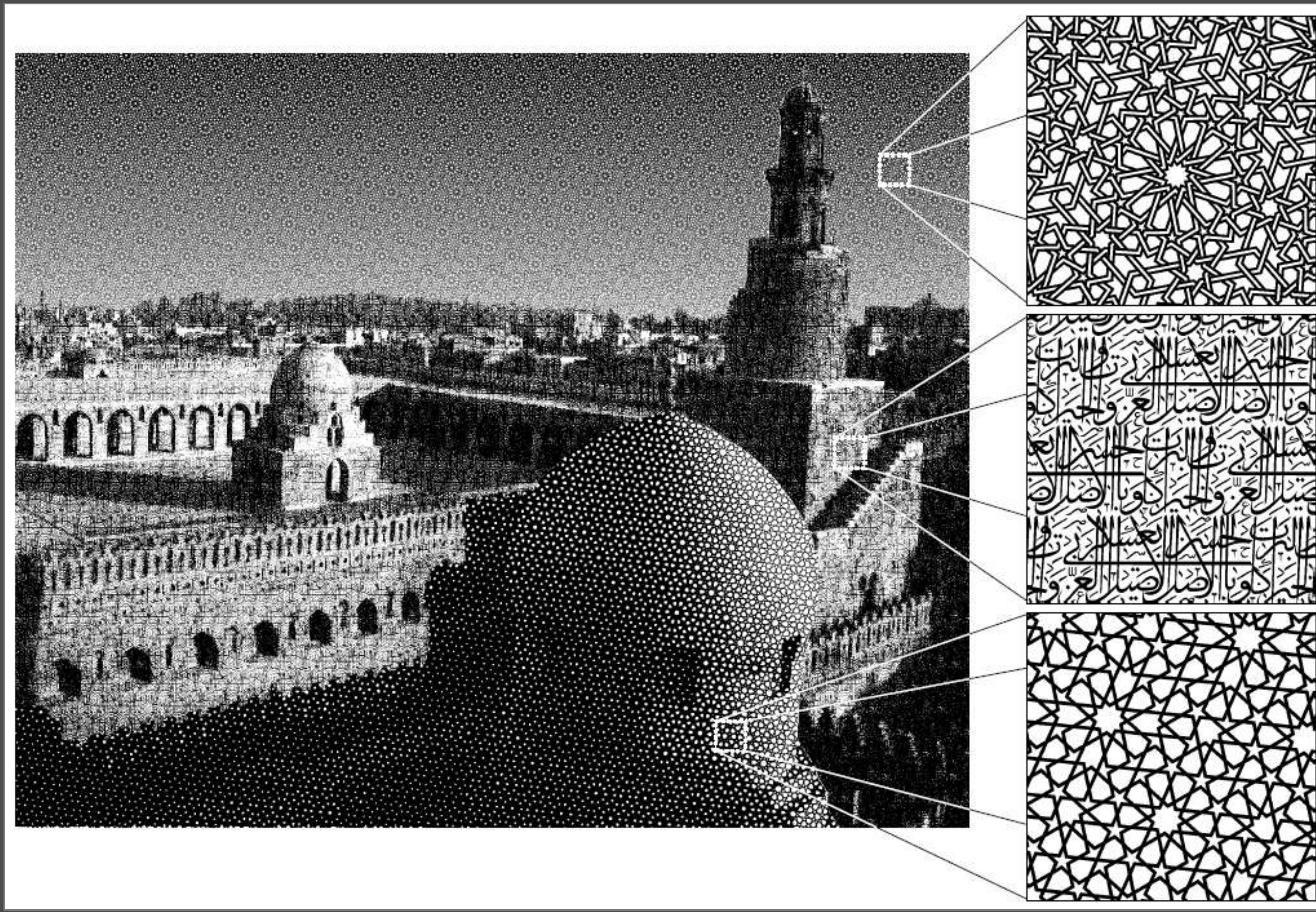
  - procedure: define outlines for key intensity levels
  - resize to that the desired gray level is achieved
  - interpolate the gray levels in-between

Artistic Screening: Procedure



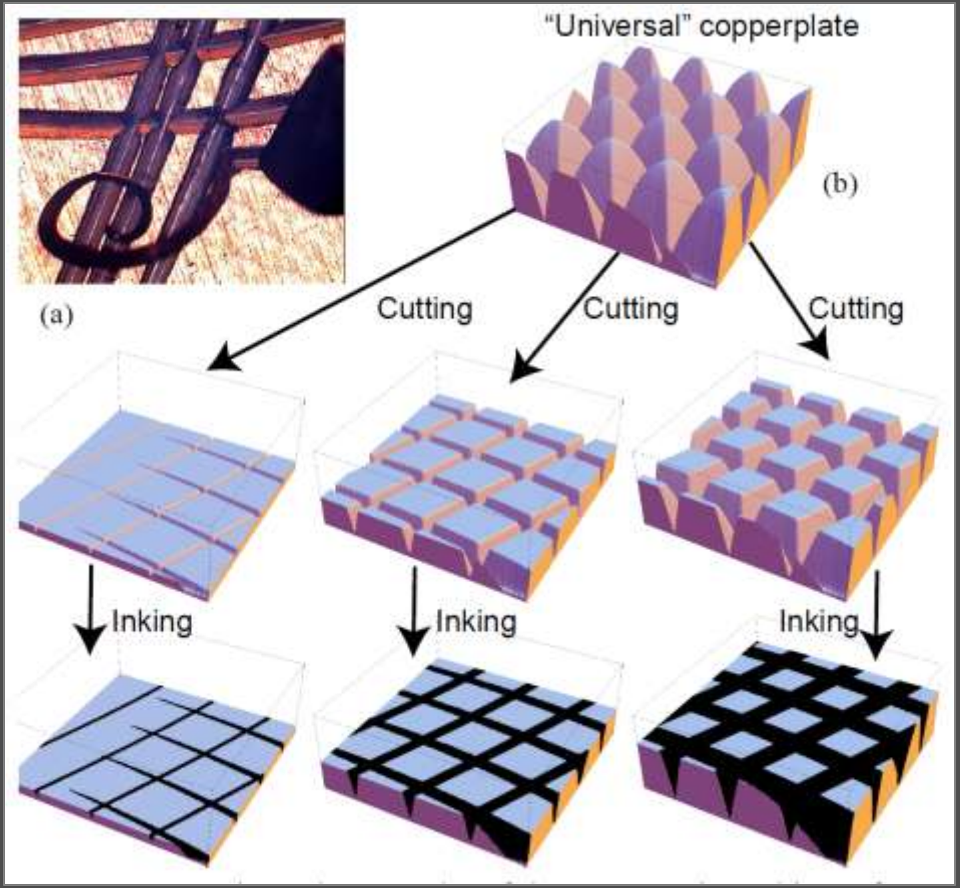
– application NOT as a dither matrix/dither screen; instead discrete elements: definition of extra grid over image with 'virtual pixels' and look up of screen by local gray level

Artistic Screening: Example

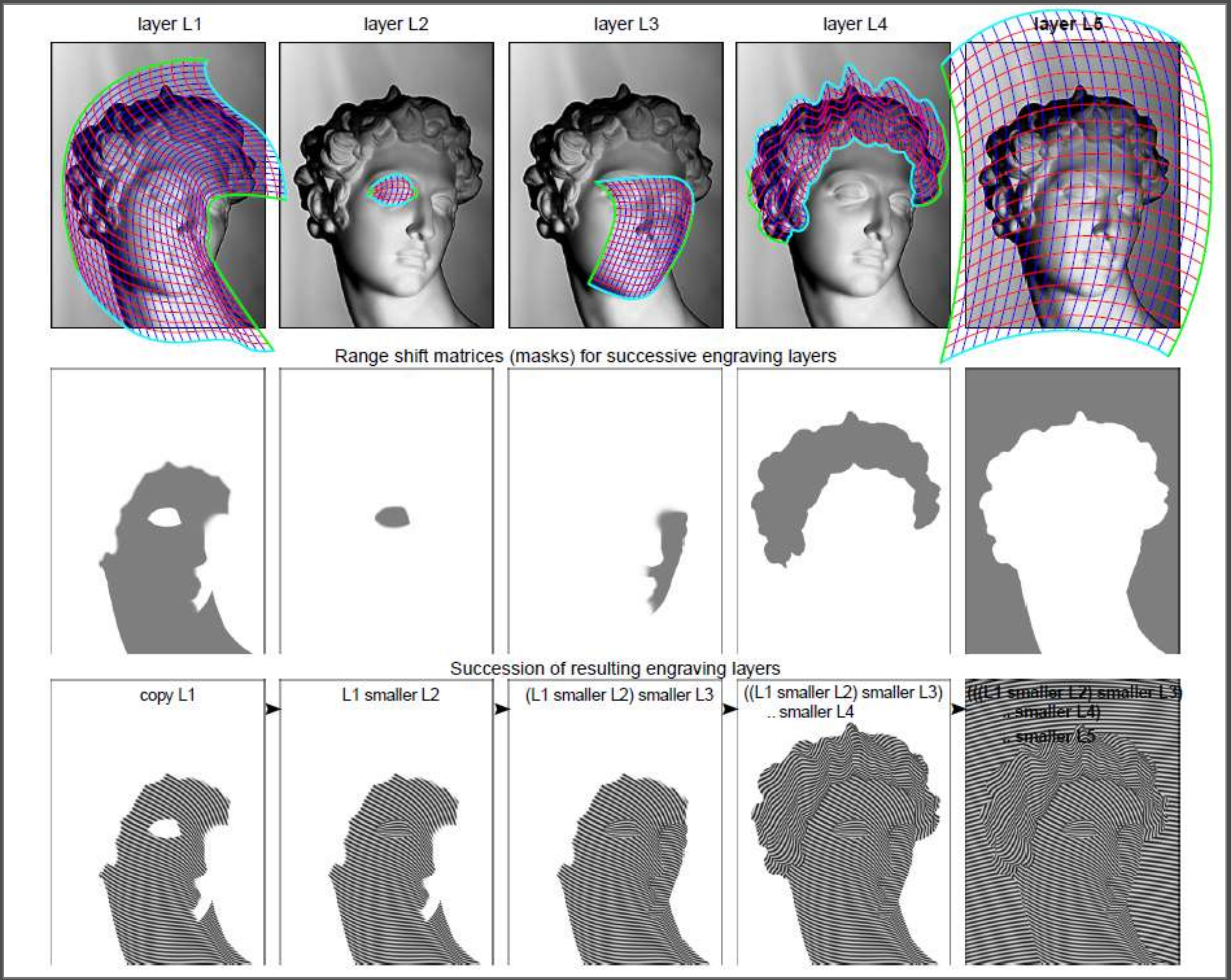


Digital Facial Engraving

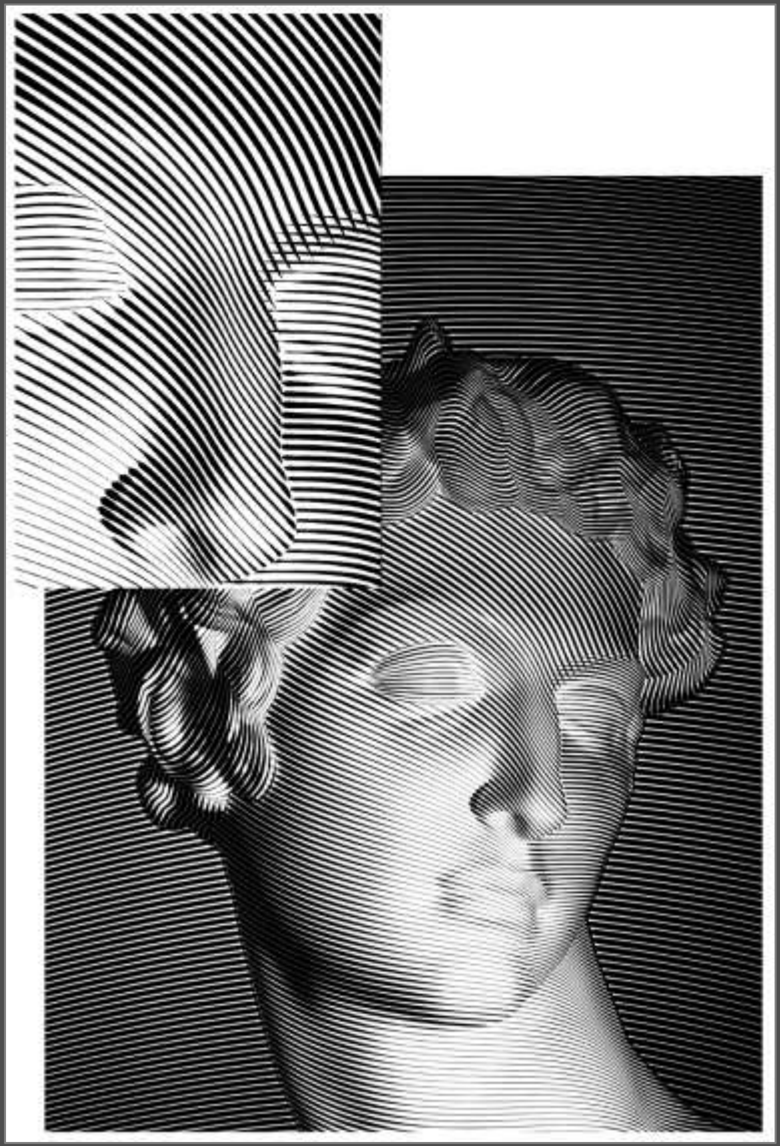
- artistic screening: halftoning elements still independent from image features (mask)
- idea: get inspired by engraving techniques
- use of controllable (copper plate) pattern, placed onto image
- depending on gray level, cut off part of the copper plate to control inking process

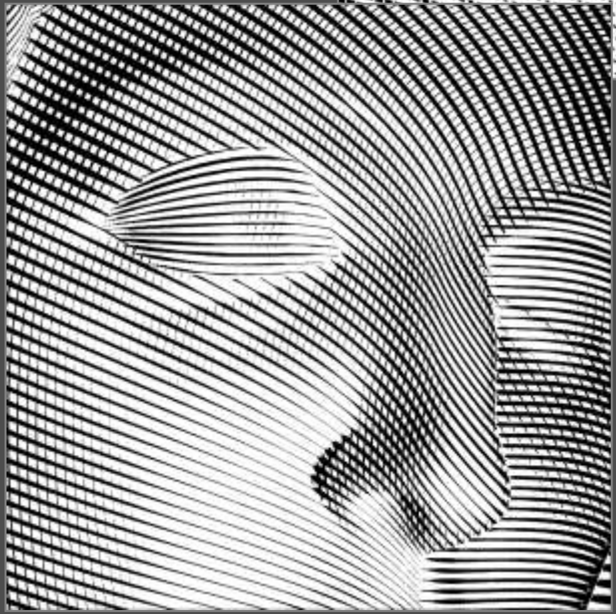


Digital Facial Engraving: Layering



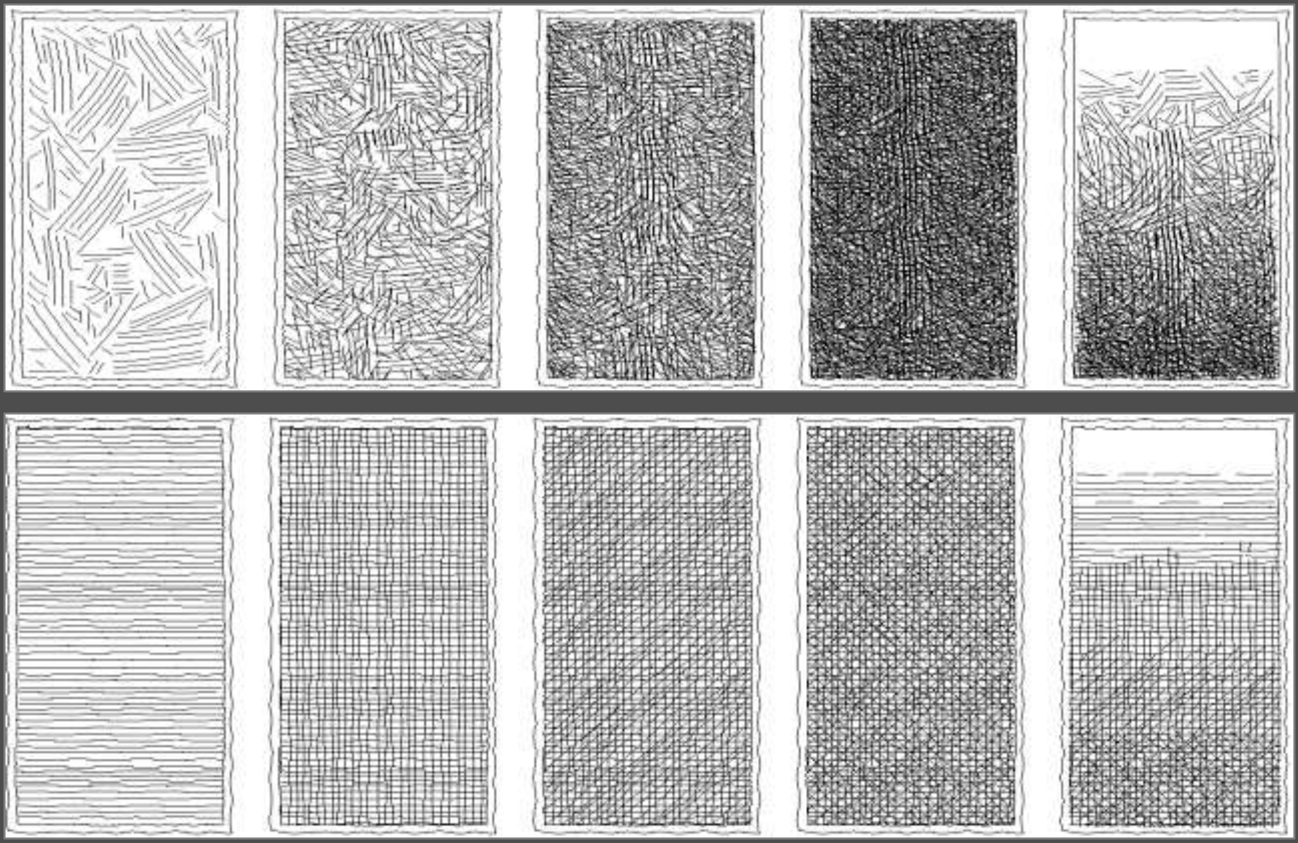
Digital Facial Engraving: Examples





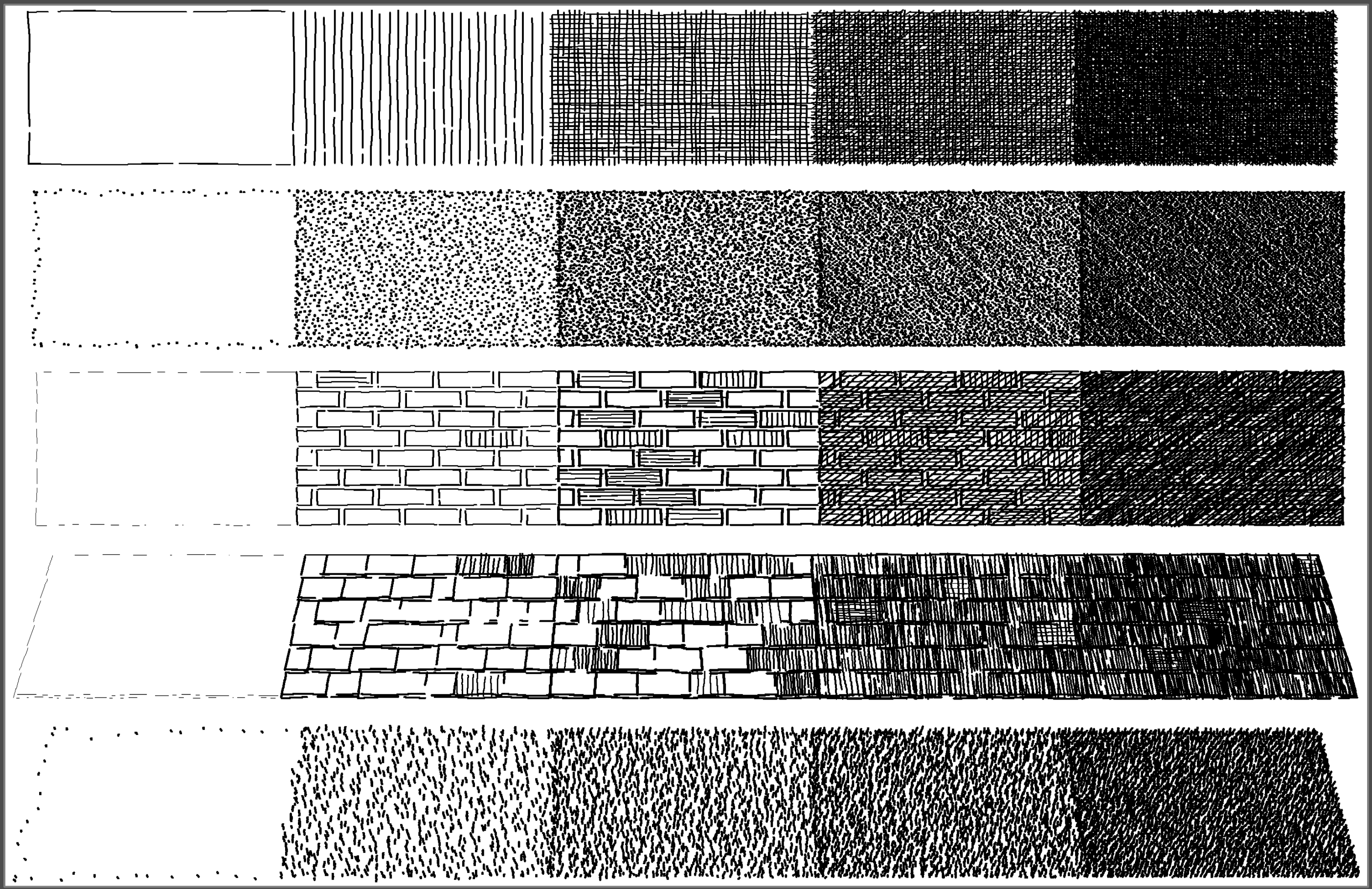
Prioritized Stroke Textures

- still a challenge: to reproduce materials
- idea: assemble textures for intended materials
  - textures as consecutive application of strokes
  - apply until intended gray level is achieved



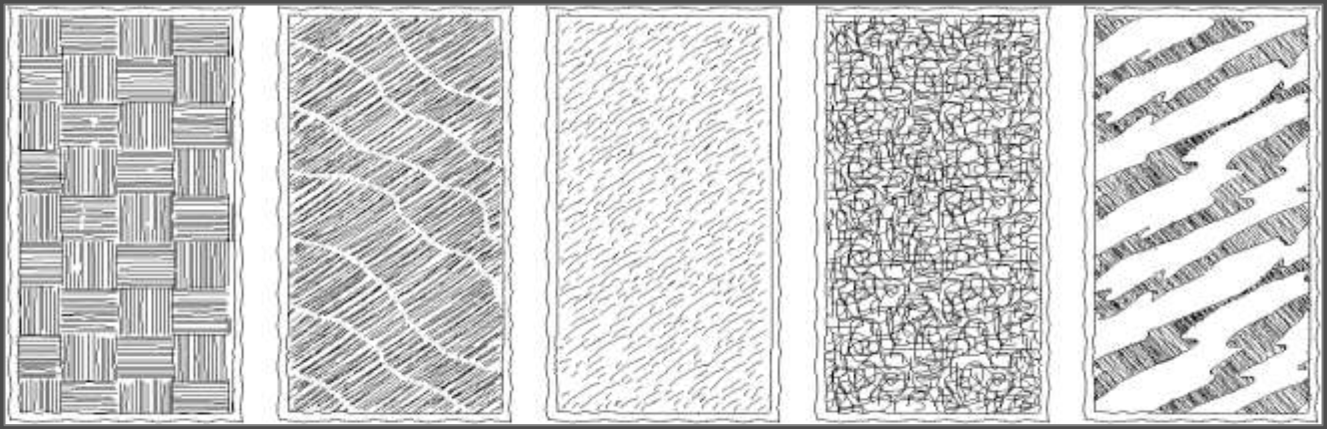
Prioritized Stroke Textures

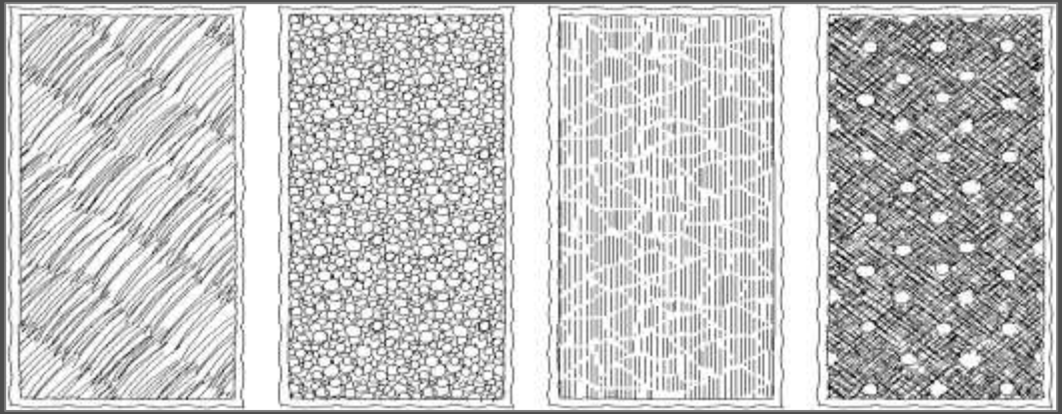
- each stroke in a “texture” is prioritized
- more important strokes rendered first



Prioritized Stroke Textures

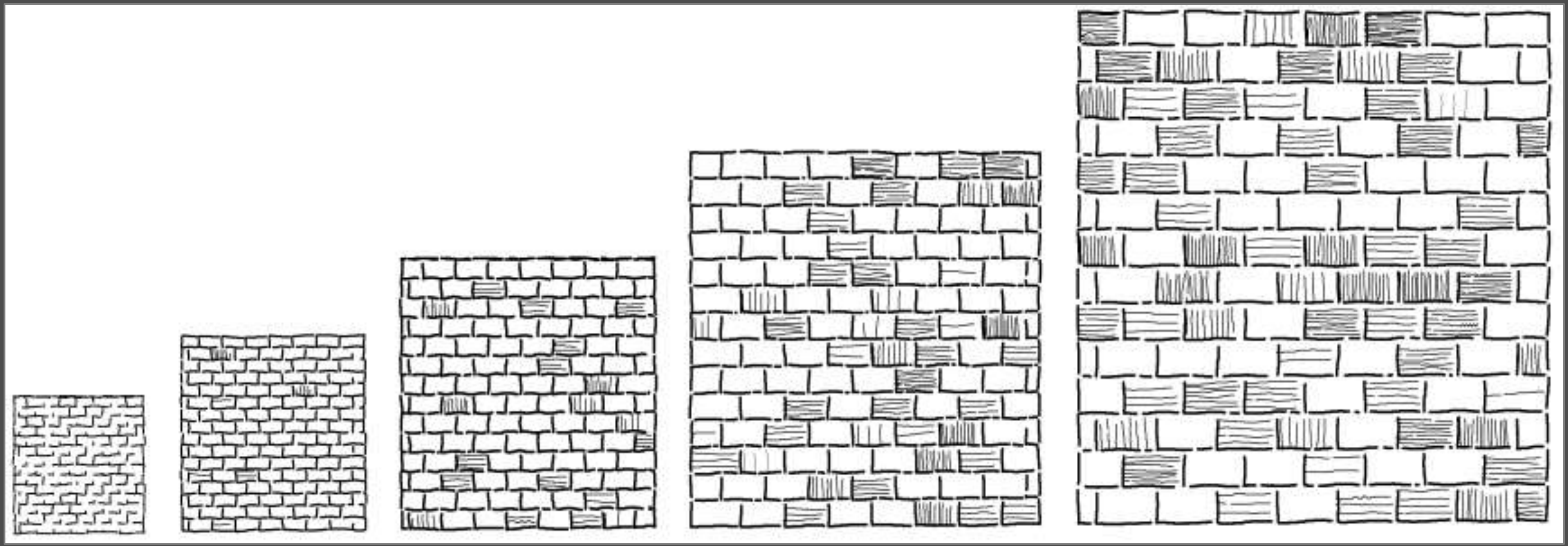
- can achieve material depiction
- captured from artists shading an image





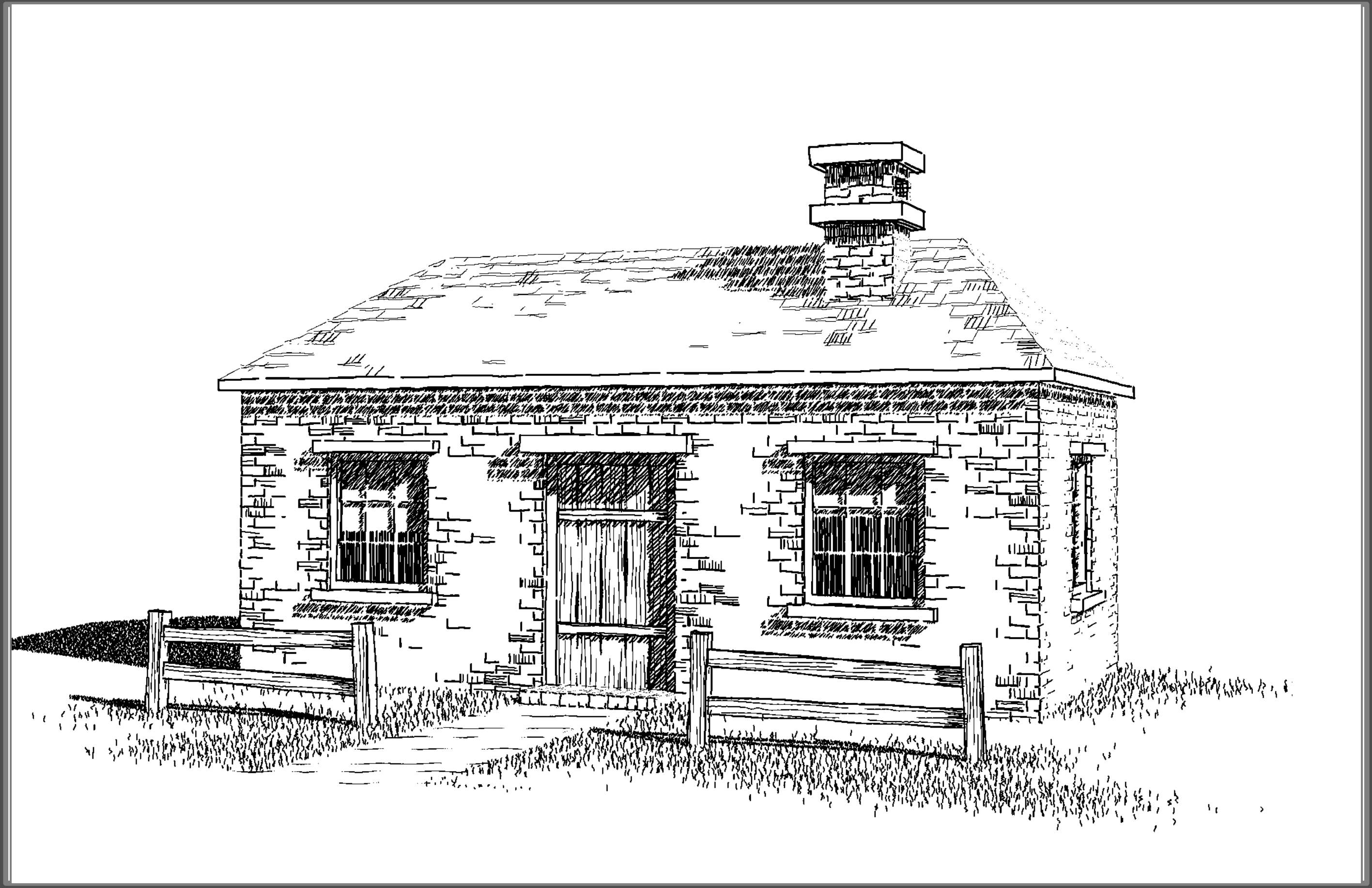
Prioritized Stroke Textures

- dealing with scale (using constant line widths)
  - at small scales, only most important strokes are drawn
  - at larger scales, more detail is added
  - the same perceived gray level is maintained



Prioritized Stroke Textures

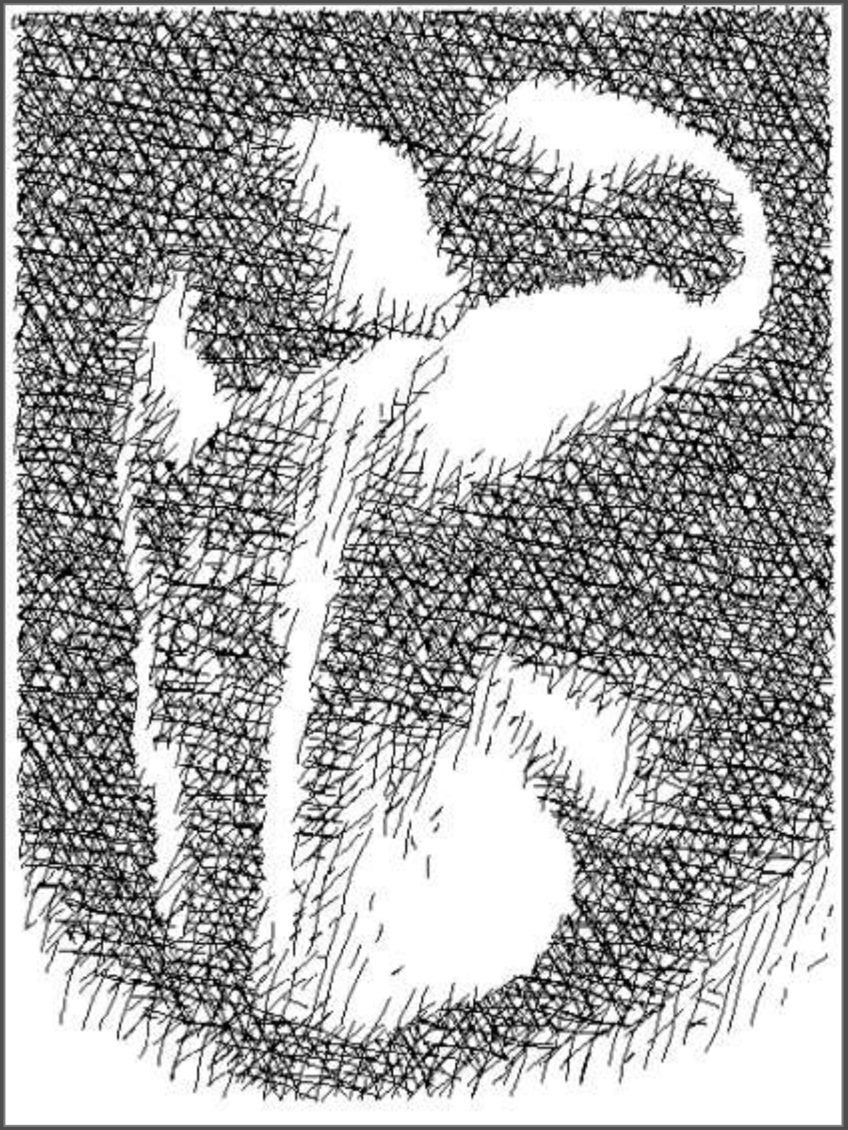
- interactive control of emphasis and abstraction



Prioritized Stroke Textures: Examples

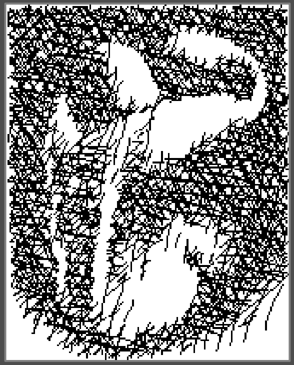
Prioritized Stroke Textures: Extensions

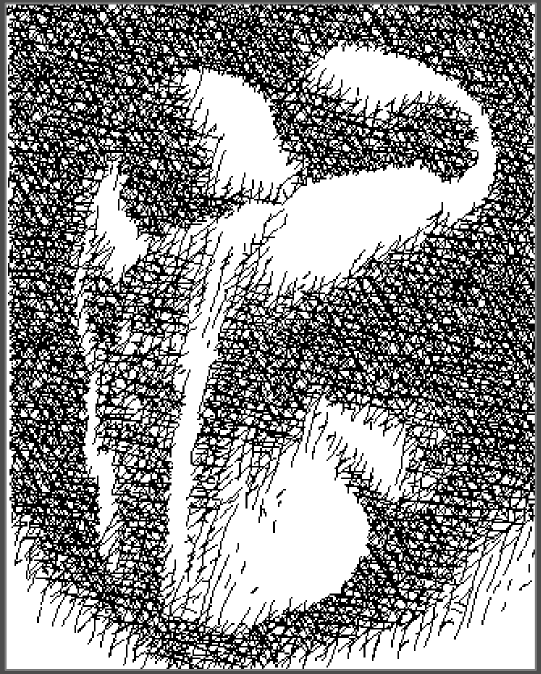
- resolution dependence
  - rescaling results in different perceived gray levels due to constant stroke widths
  - needs to be addressed



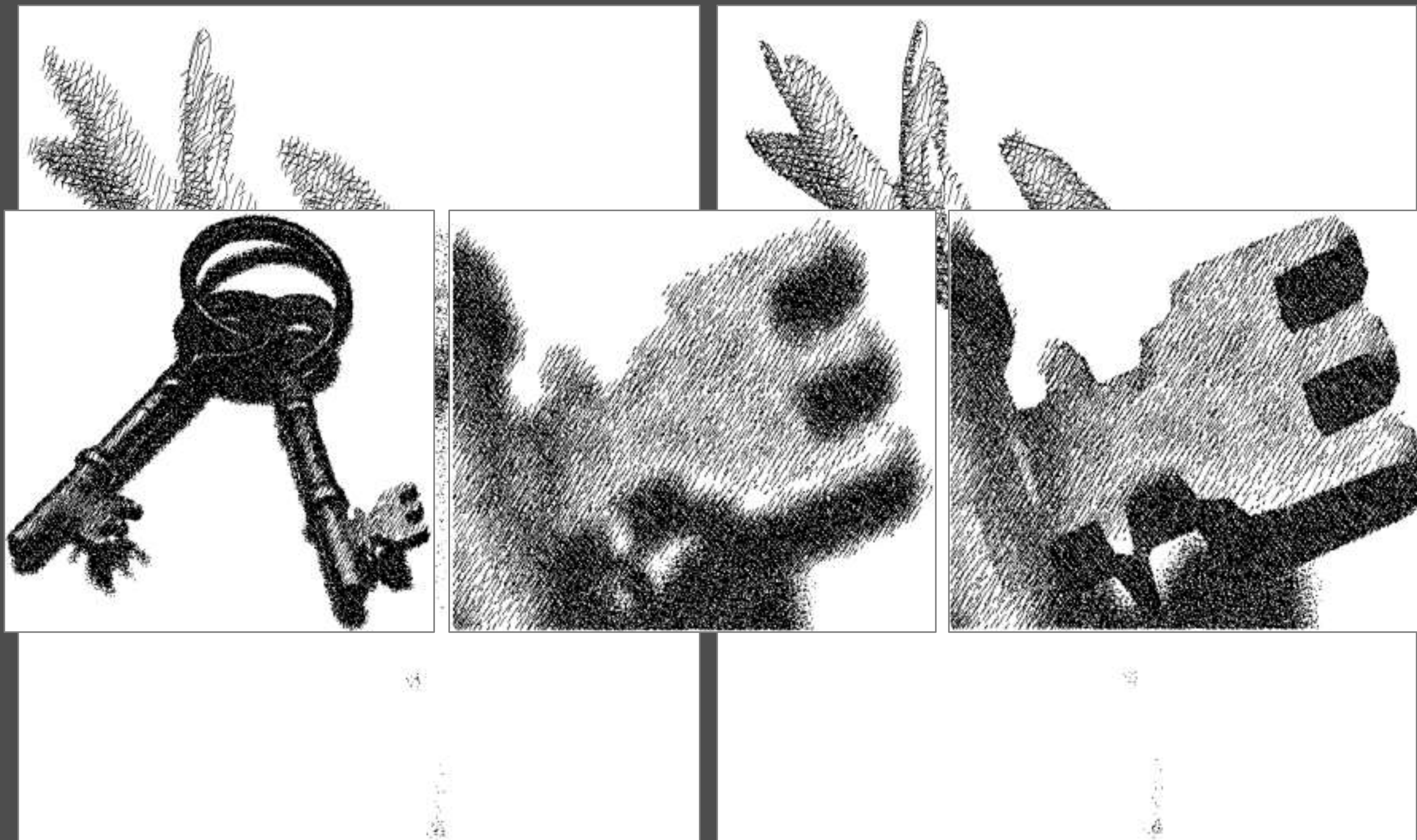
Prioritized Stroke Textures: Extensions

- solution: rescale gray level image, apply new strokes
  - issue: unsharp edges
  - solution: discontinuities need to be considered



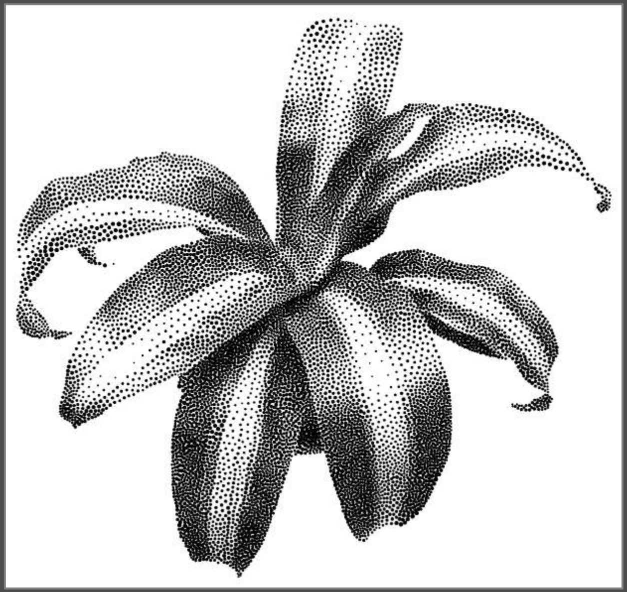


Prioritized Stroke Textures: Extensions

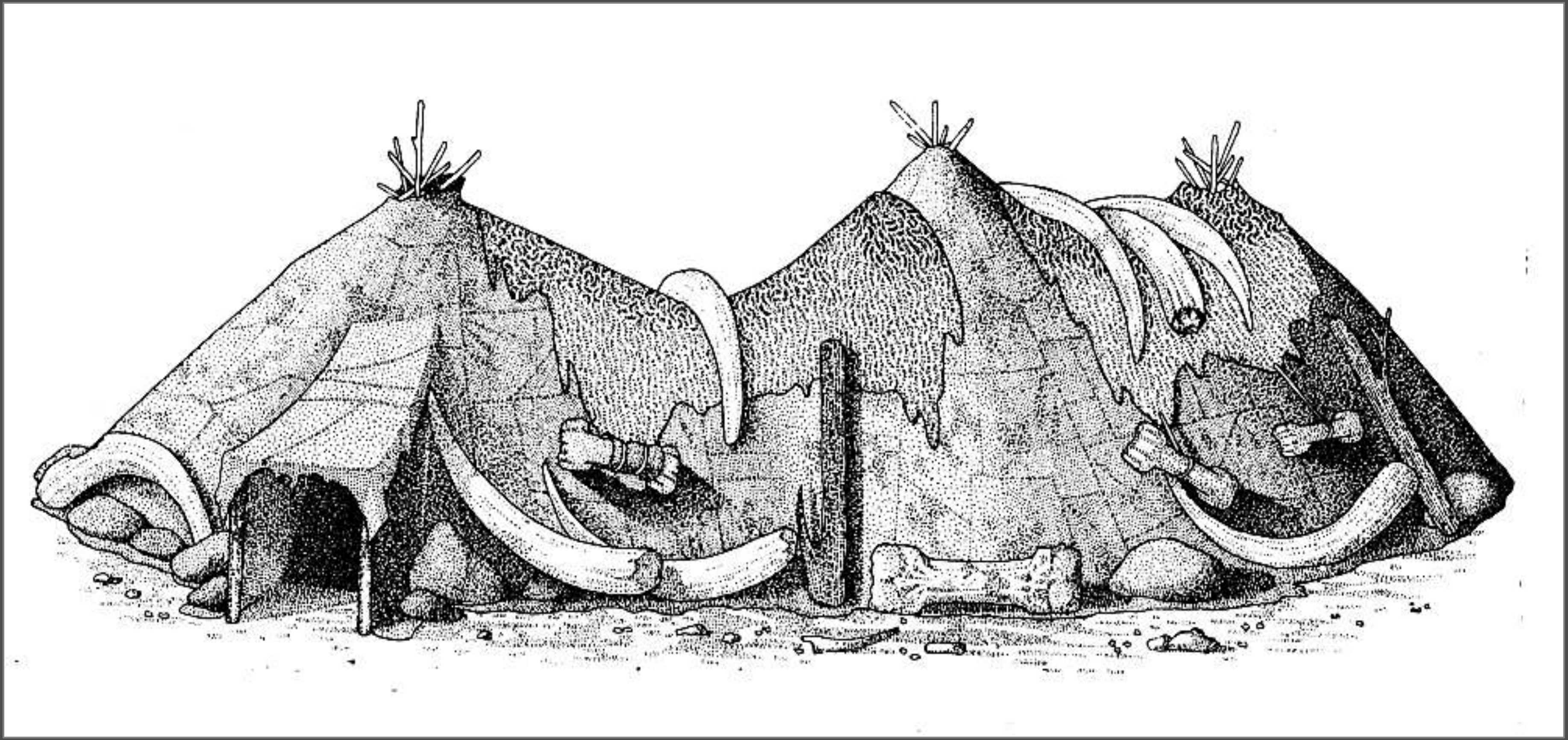


Pen-and-Ink Rendering

Rendering with Dots: Stippling

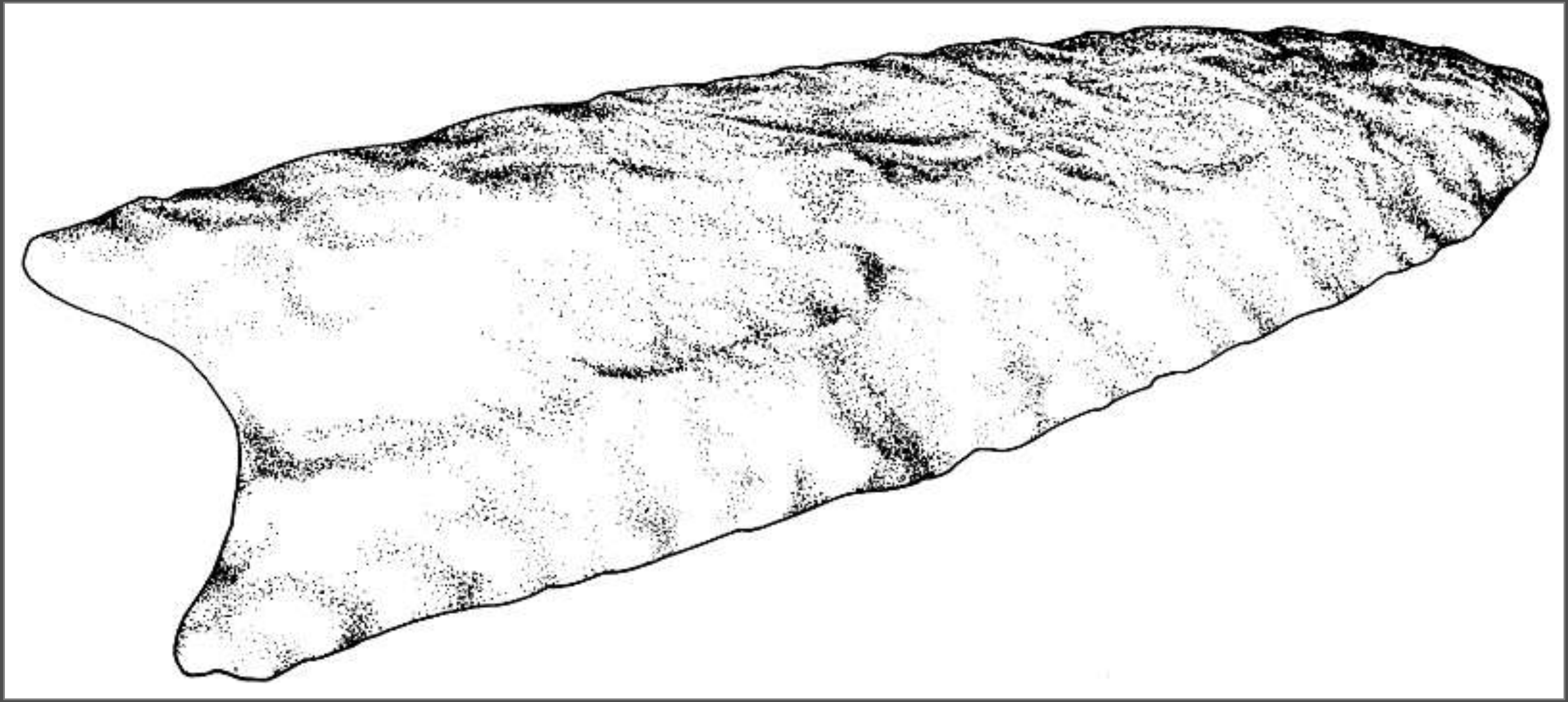


Stippling as an Illustration Technique



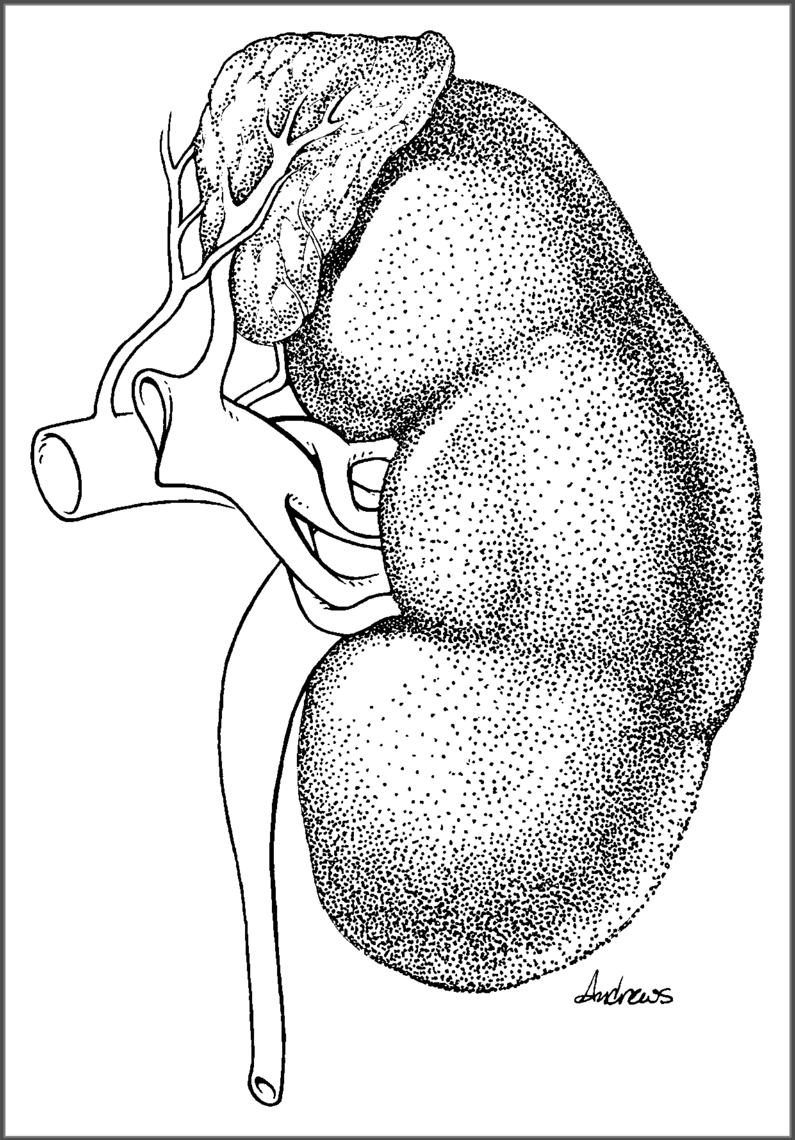
Jacquetta Hawkes: *The Atlas of Early Man.* St. Martin's Press, 1976

Stippling as an Illustration Technique



Lynda Smith Touart

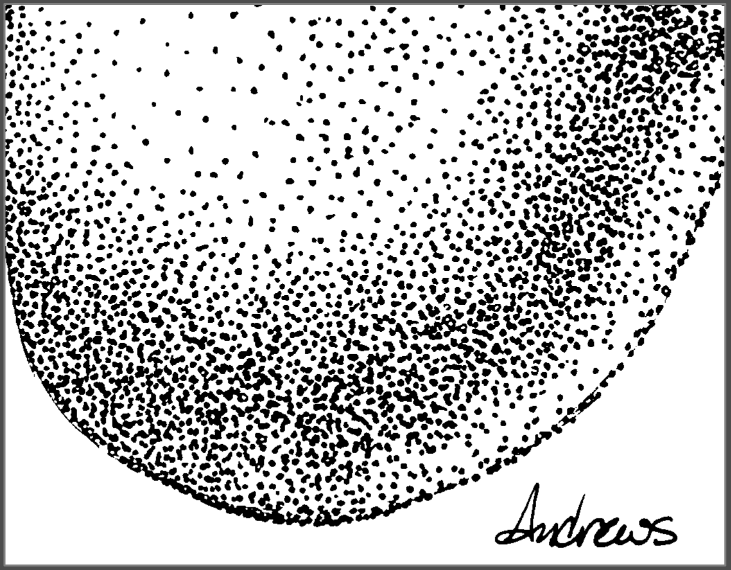
Stippling as an Illustration Technique



courtesy of
B. Andrews

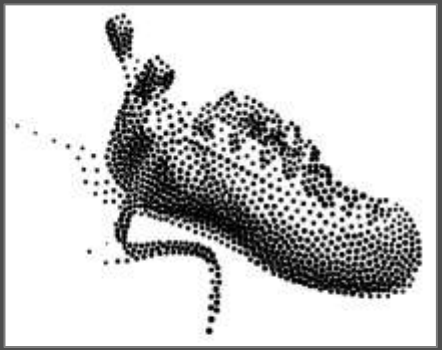
Rendering with Points: Stippling

- why points, what do the points represent?
  - suggest form in natural graduation of tone
  - do not suggest directionality of a surface structure
  - typically arranged so that they do not exhibit artifacts
  - can also suggest form if placed along (curved) lines
  - each stipple has a purpose
- dot-based rendering in NPR
  - dots **not** as intentional artifacts (e.g., some halftoning methods)
  - dots as intentional artifacts (more like traditional stippling)



Rendering with Points: General Approach

- all points are equivalent, distribution matters
- goal: approximate a (grayscale) source image

 →  → 

- general approach thus is about distribution:
  - obtain an initial point distribution
  - modify this distribution with respect to
    - image density
    - image properties (edges, gradients, etc.)
  - user-interaction for fine-tuning

Initial Point Distribution

- goal: even but random distribution of stipples
- possibility 1 – brute force
  - generate random coordinate
  - place if desired tone not yet achieved

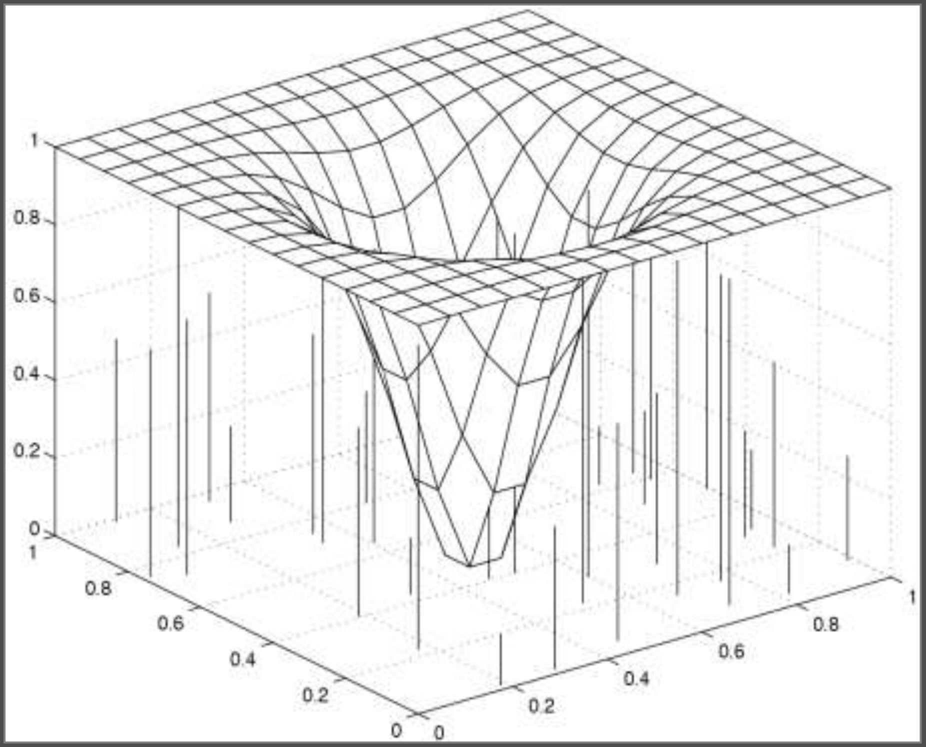
  → requires local tone estimation for each point

  → stipples may overlap and are not equally distributed
- possibility 2
  - use result of halftoning algorithm

  → usually too many points
- need to put emphasis on initial distribution to have a good starting point for further processing

Initial Point Distribution

- height map approach (based on brute-force)
  - generate random locations and for each a value in [0, 1]
  - view image intensity as a surface and the locations as lines with a height determined by the random number
  - select points to draw if they intersect the surface
  - white regions (with high intensities) will have few points that pass
  - dark regions (with low intensities) will have more points that pass



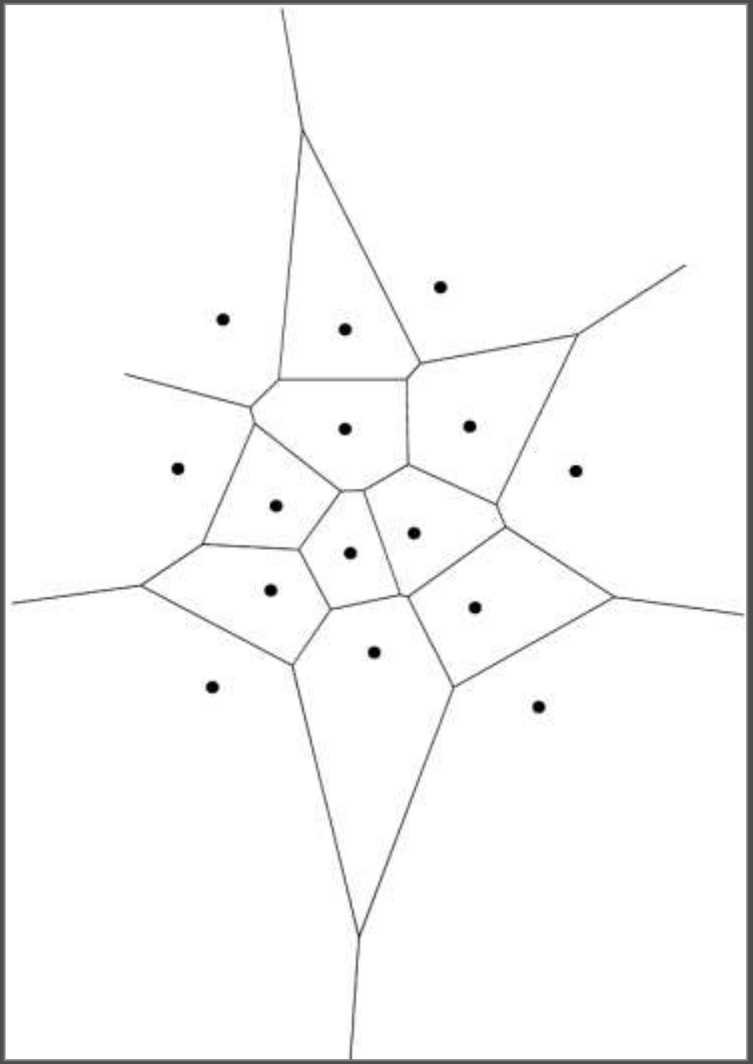
Goals for NPR Stippling

- no overlapping stipples
- even distributions:
  i.e., stipples should have about the same distance to their neighbors
- linear intensity response:
  the perceived gray values generated for a linear gray ramp should also behave linearly
- no visual patterns if possible

- **idea:** use point relaxation to achieve distribution

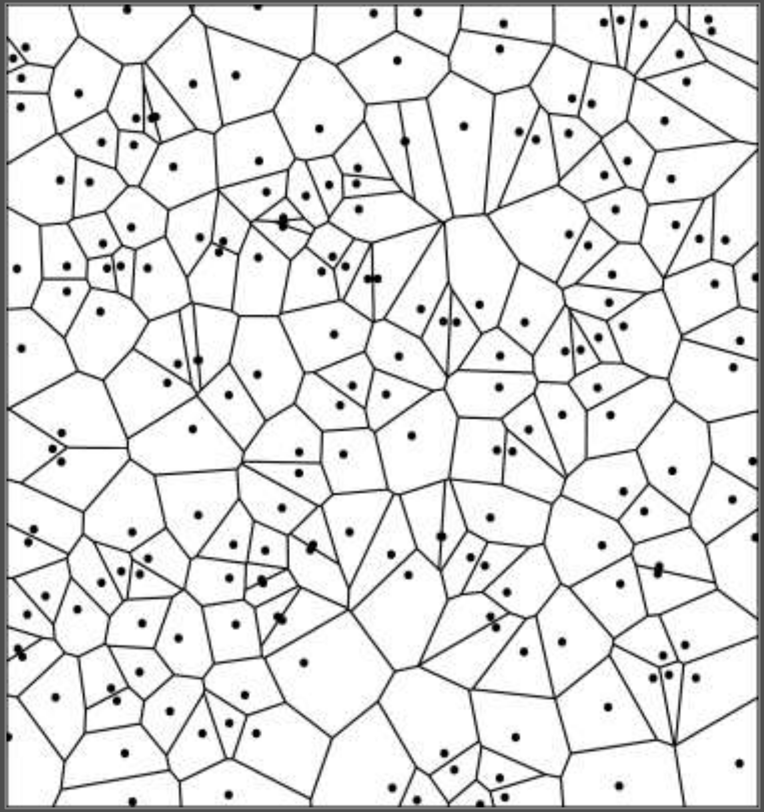
Relaxation to Obtain Even Distributions

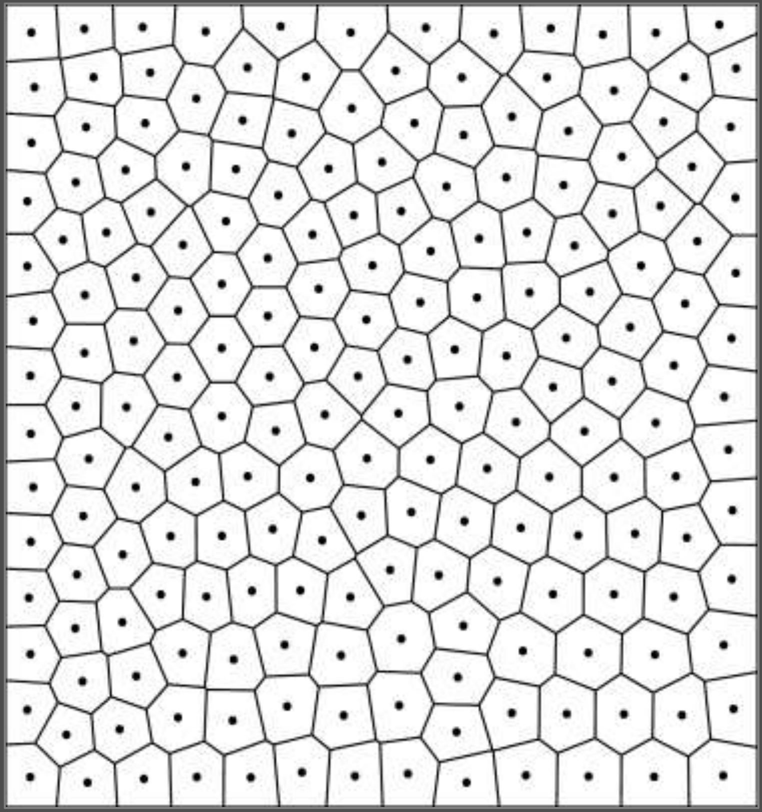
- based on Voronoi diagrams
- goal: starting from an initial distribution and its Voronoi diagram, compute the Centroidal Voronoi Diagram
  - each point lies on the centroid of its Voronoi region, i.e., it minimizes the mean square distance from the points to the centroids
  - thus, points are well-spaced
  - called "Lloyd's method"



Relaxation to Obtain Even Distributions

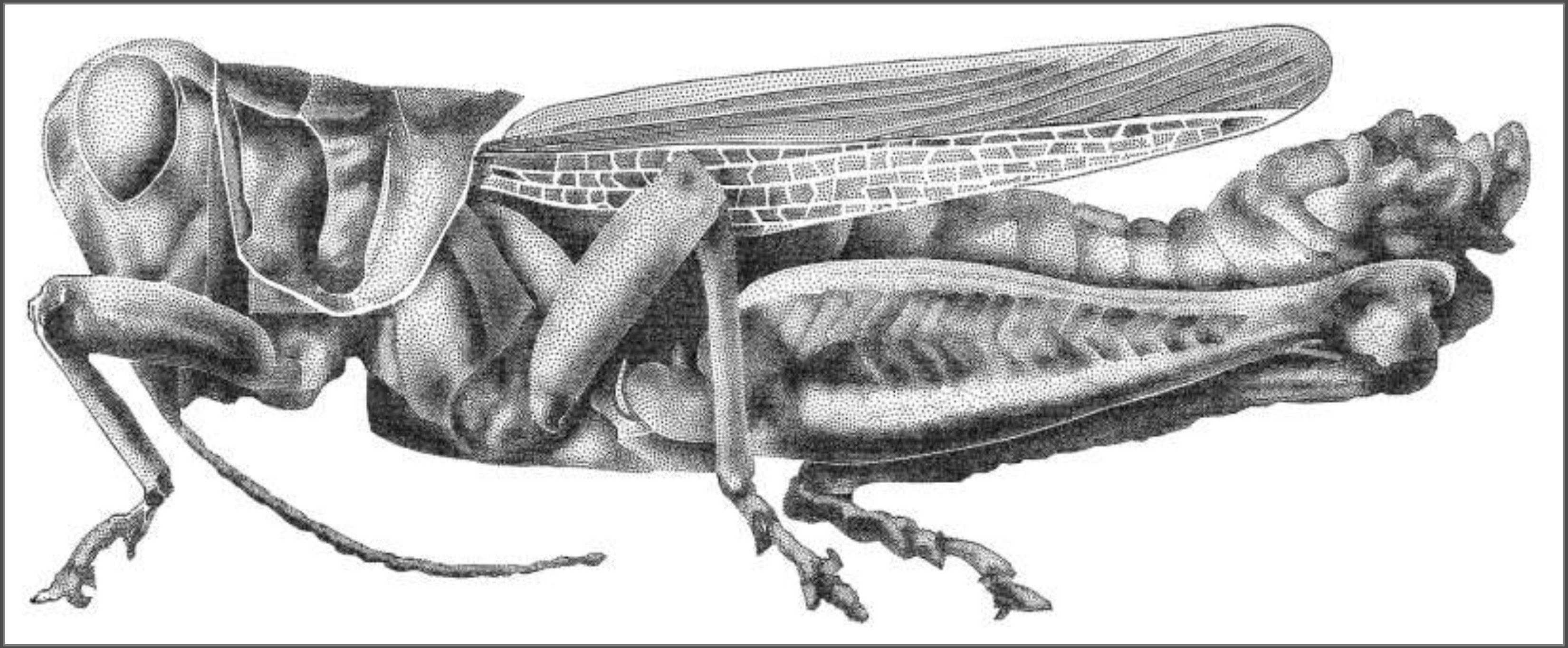
- iterative process:
  - compute Voronoi diagram
  - move the points to the centroid of its region, repeat





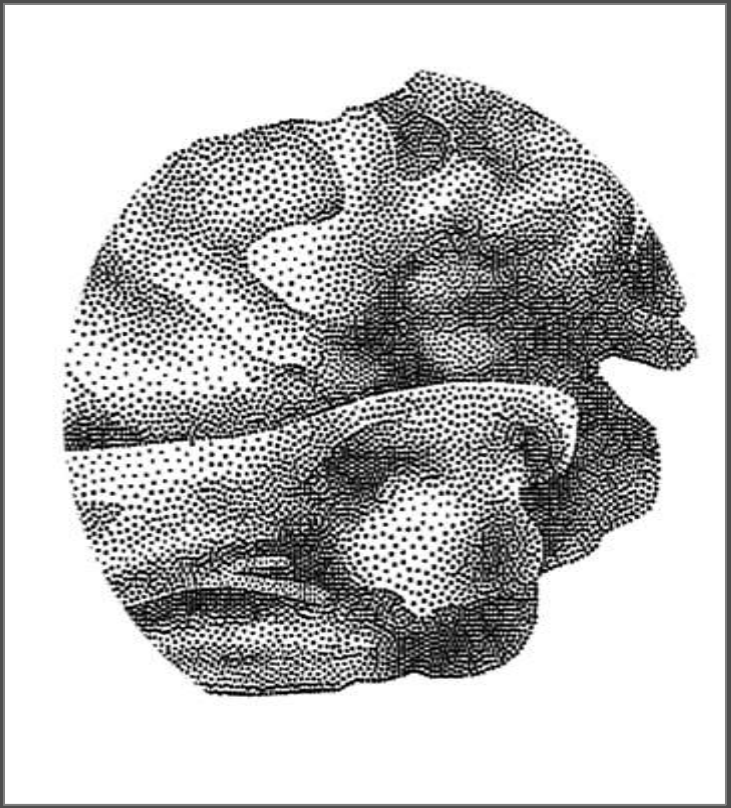
Interactive Relaxation-Based Stippling

- 1st approach: interactive refinement
  - subdivide image into separate regions to be able to separately apply different stippling styles
  - use brushes to apply relaxation (and other interactions) to save computational effort $\rightarrow$ O(n log n)



Interactive Relaxation-Based Stippling

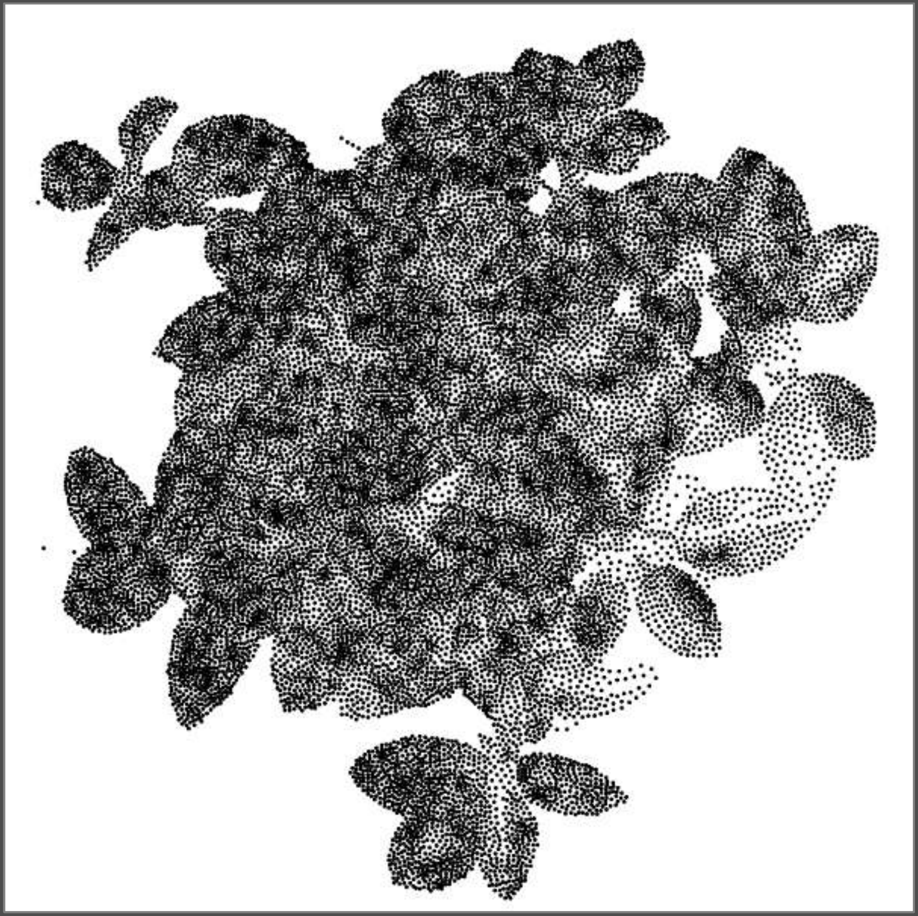
- small regions to treat at a time (complexity)
- brushes similar to those in paint programs
- can have several influences
  - selection brush
    $\rightarrow$ identify larger regions
  - relaxation brush
    $\rightarrow$ local relaxation
  - jitter brush
    $\rightarrow$ add some randomness
  - shape brush
    $\rightarrow$ modify shape & size of dots
  - new points brush



Automatic Relaxation-Based Stippling

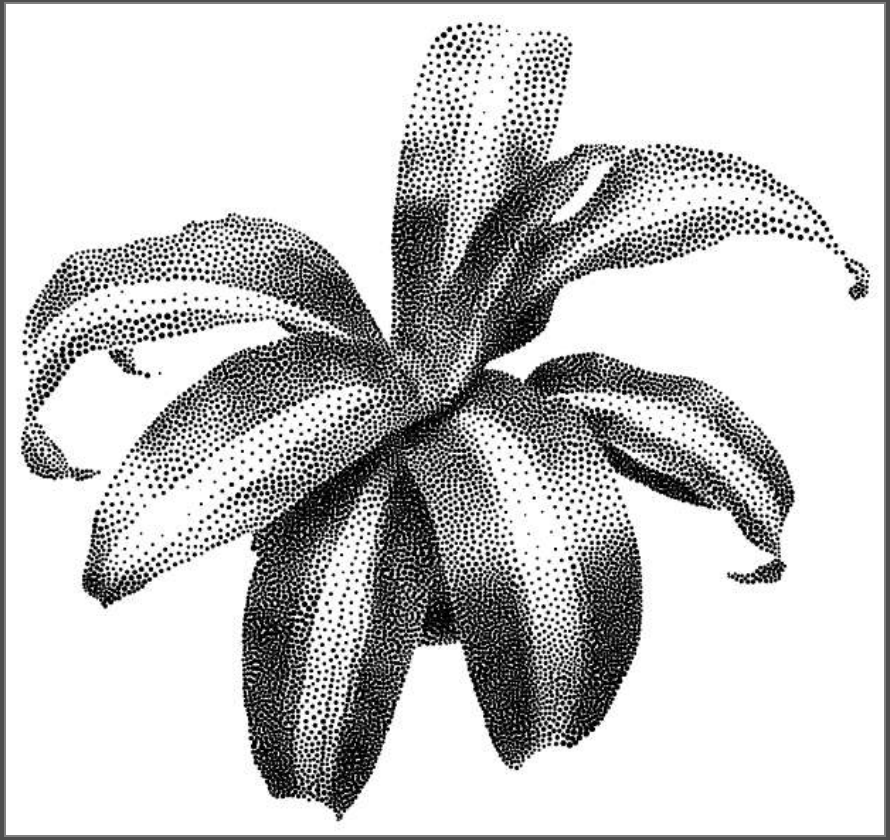
- while local influence is good, it requires much effort
- thus, automatic technique desired; problems:
  - repeated relaxation evens out also the initial distribution
  - also burrs boundaries because image is not considered
- idea: use weighted centroidal Voronoi diagrams
  - density function that packs points closer together in darker regions and wider apart in whiter regions
  - density function included in Voronoi region computation
- extensions
  - affect point size based on density
  - pre-process input image (contrast)

Automatic Relaxation-Based Stippling



constant point size

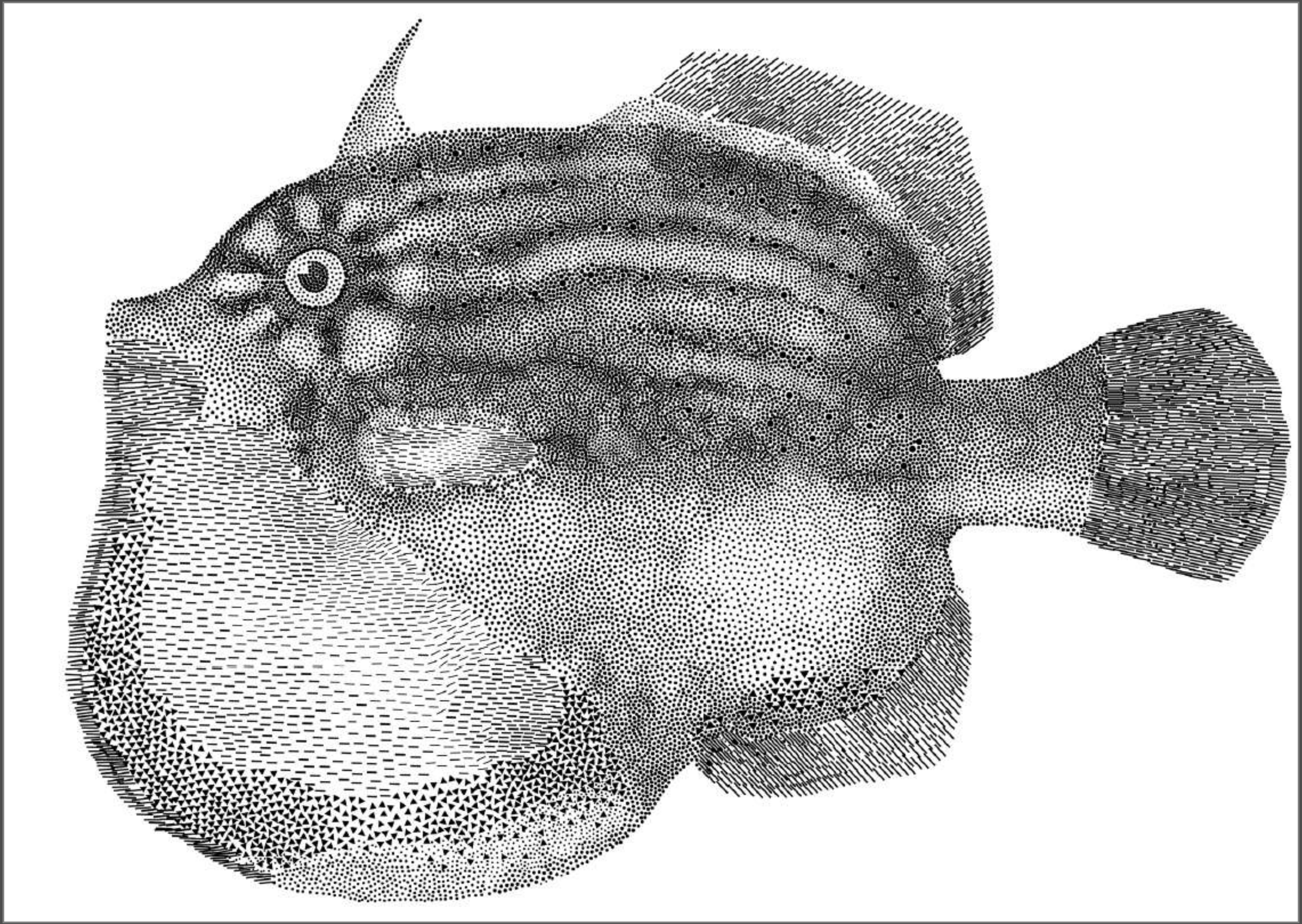
Automatic Relaxation-Based Stippling



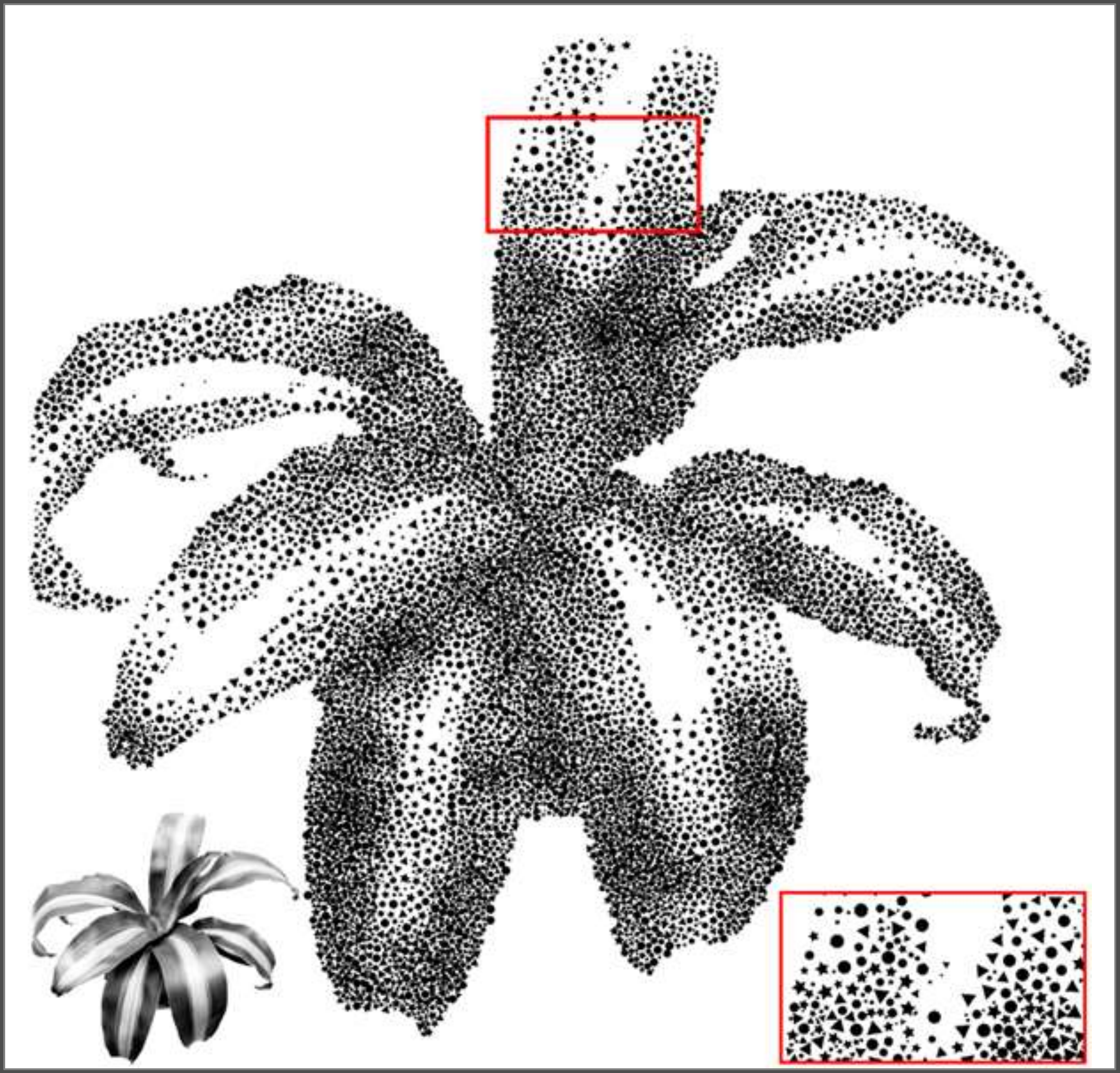
Stippling with other Primitives

- in reality the dots are not perfectly round
  - have a noisy shape
  - have a general shape that is not round
- can also be used for stippling: considering primitive shape as well in the Voronoi diagram computation
  - by looking at the area covered by the primitive
  - by looking at the outline of the primitive
- considering size, outline, and area yields better packing of stipples that have different shapes and sizes

Stippling with other Primitives

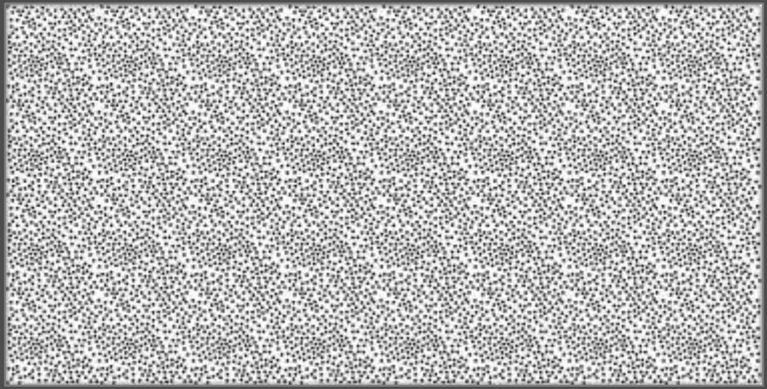


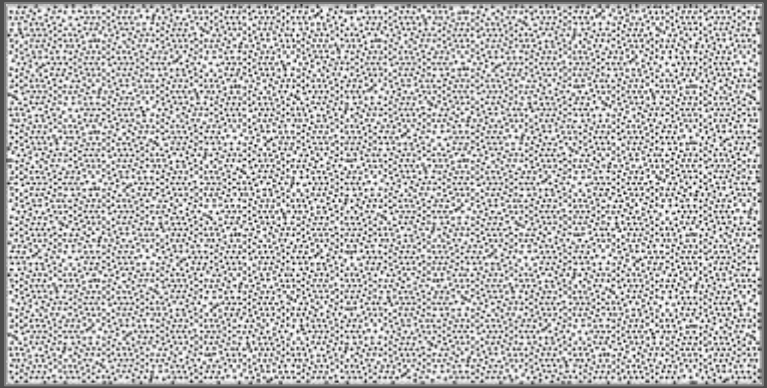
Stippling with other Primitives



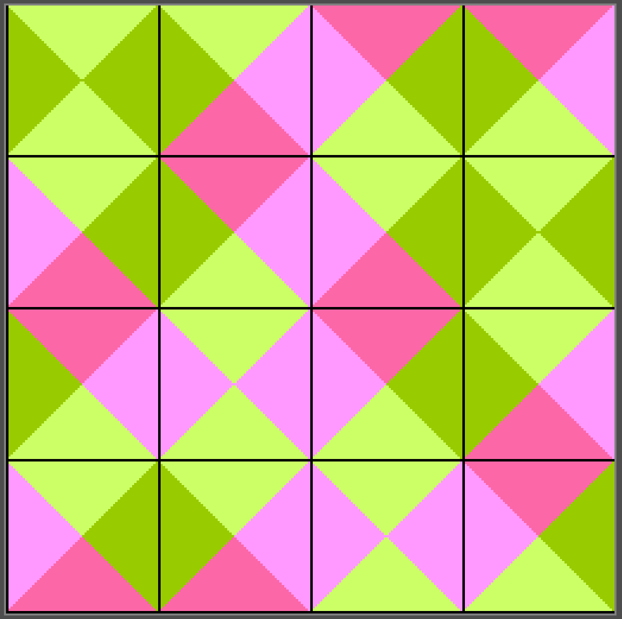
Stippling with Recursive Wang Tiles

- problems that still remain:
  - high computational cost of computing stipple distributions
  - remaining artifacts resulting from relaxation approach
- possible solution: use tiles of pre-computed distributions of stipple points
- problems:
  - stippling based on tiling may look repetitive
  - stipple distributions do not match at the tile borders





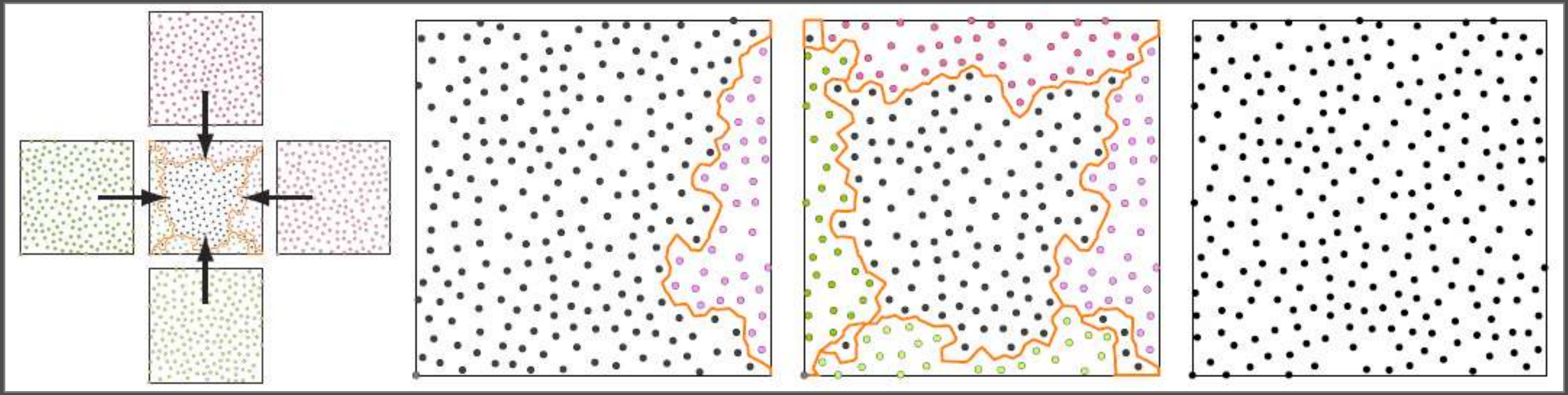
Stippling with Recursive Wang Tiles



- idea: use Wang tiling
  - tiling of the plane using a set of tiles with color-coded edges
  - each edge can only be matched with another tile of the same color
  - depending on # of colors used yields large non-periodic tiling
- associate each tile with matching point distribution
  - need to make sure that each individual tile has "good" distribution properties
  - need to make sure that tiles match according to colors

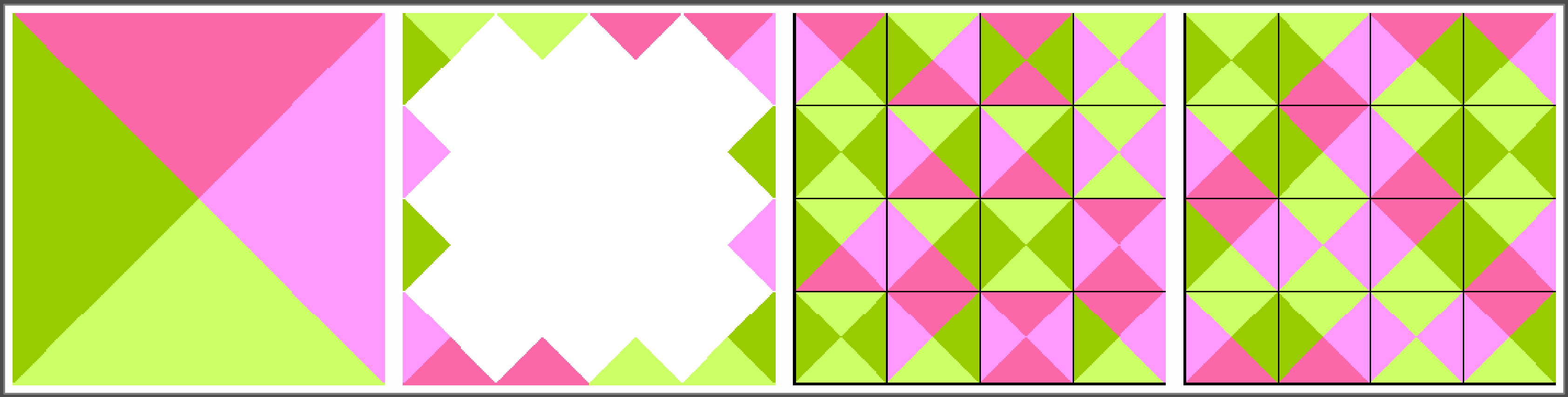
Stippling with Recursive Wang Tiles

- associate each tile with matching point distribution
  - start with a distribution for each color
  - for each unique tile of color combinations create another distribution and try to make it fit with the color ones
  - find seams that have the lowest costs with respect to the distribution constraints



Stippling with Recursive Wang Tiles

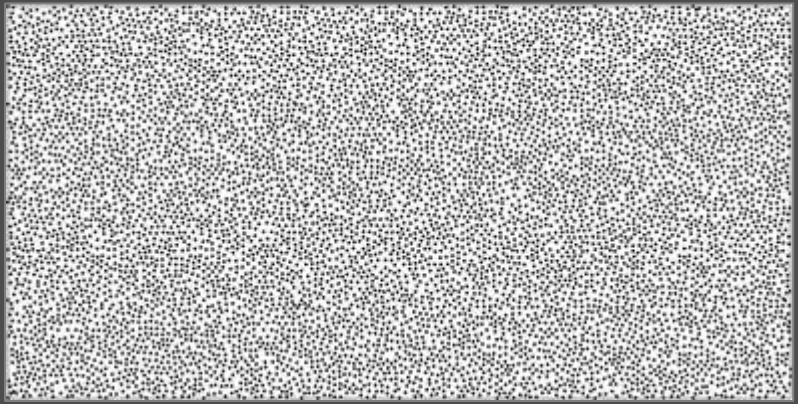
- recursion to be able to “zoom in” to a distribution
  - need order in the stipple distribution of each tile
  - need a subdivision scheme for the tiling itself
- approach
  - each color is assigned a unique sequence of new colors
  - stochastic search of possible tiling with border constraints



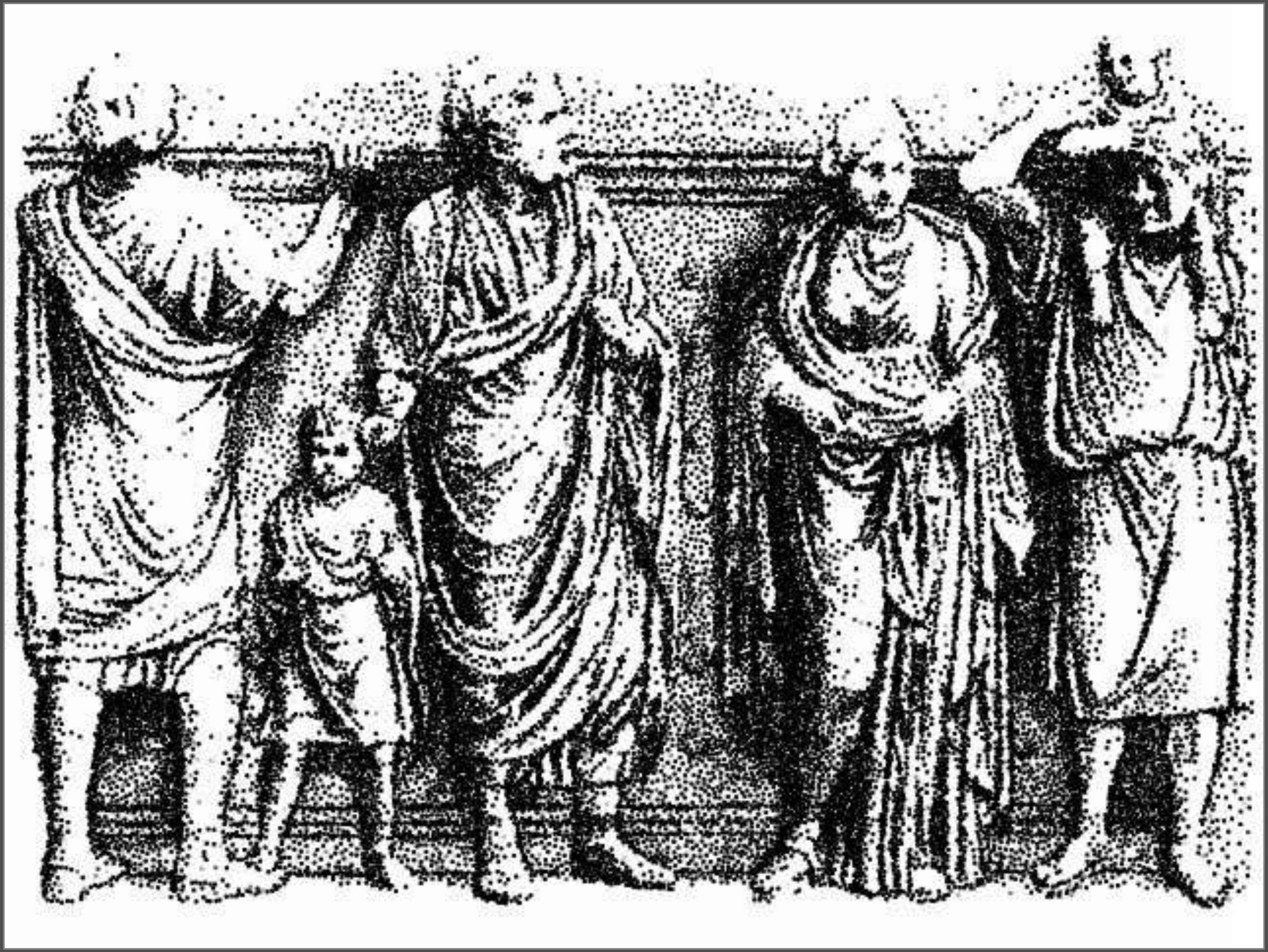
  - relaxation to match base and detail point distribution

Stippling with Recursive Wang Tiles

- result: recursive, non-periodic tiling that can be applied to interactive stippling
- ranking of each stipple can be used to determine which stipple is rendered first to approximate a grayscale image

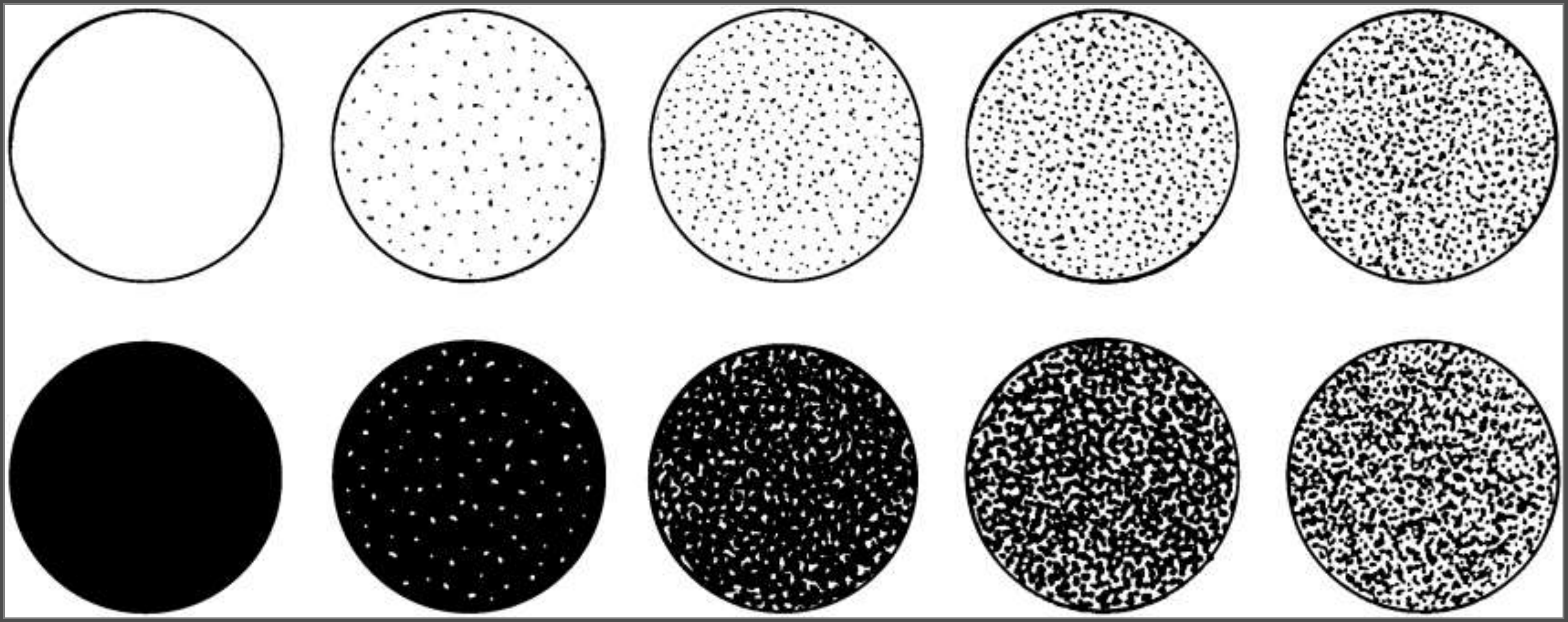


Stippling with Recursive Wang Tiles

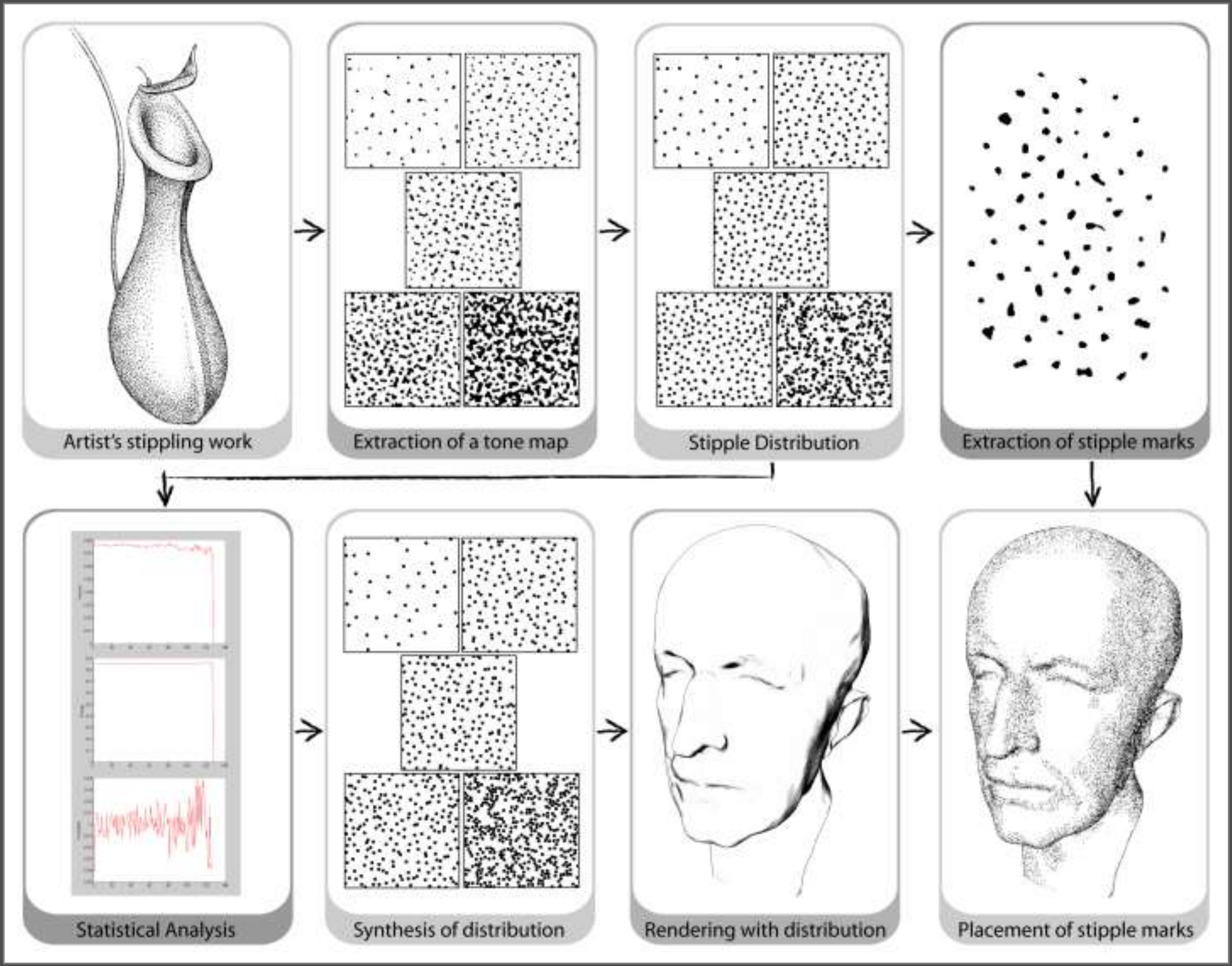


Example-Based Stippling

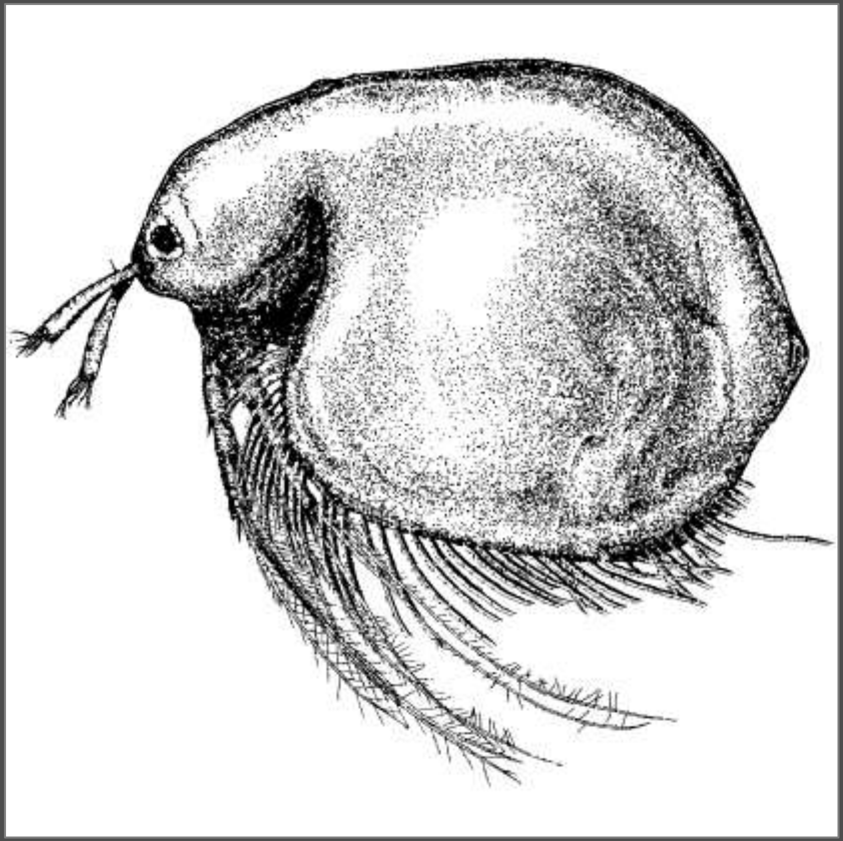
- problems so far: results too sterile
- idea: base stippling on human-drawn exemplars
- 2 issues to address: placement & stipple dots

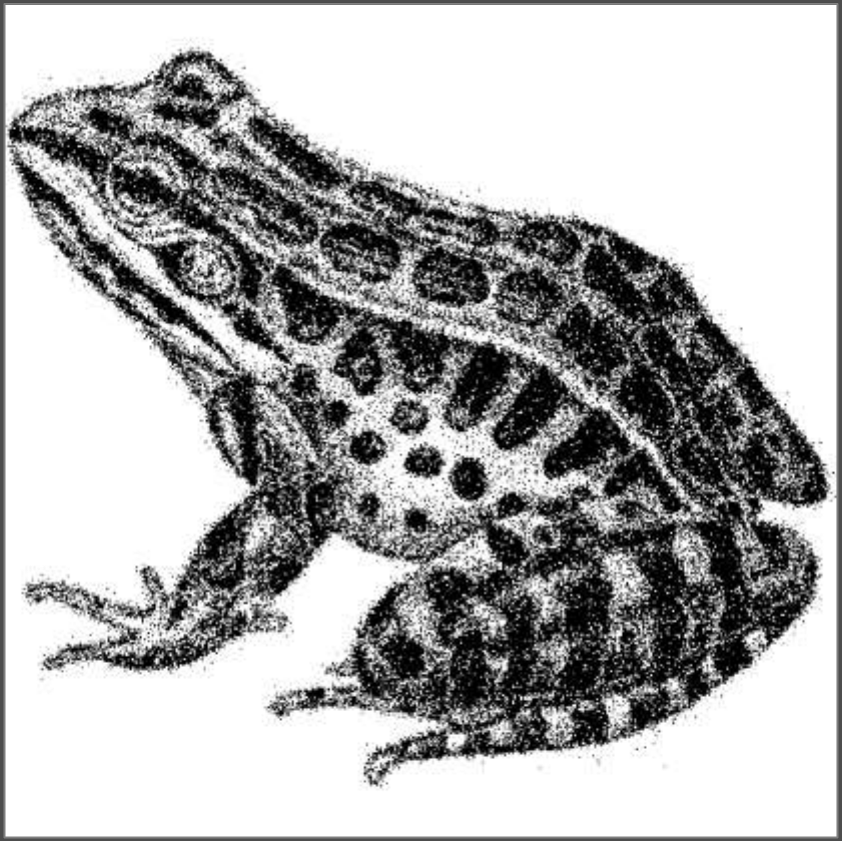


Example-Based Stipple Placement

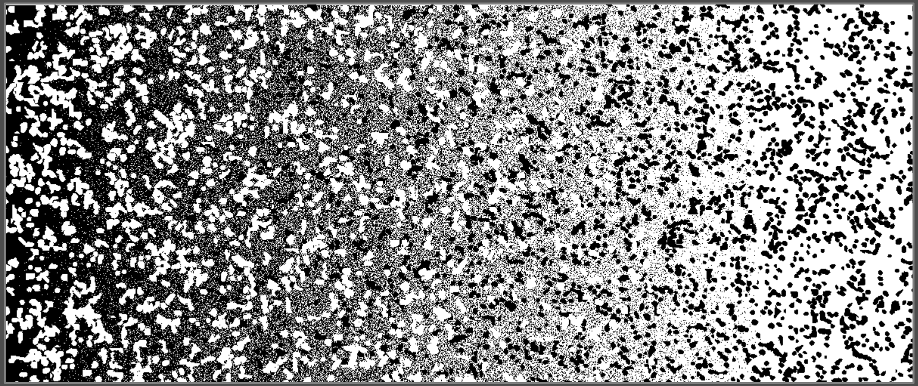


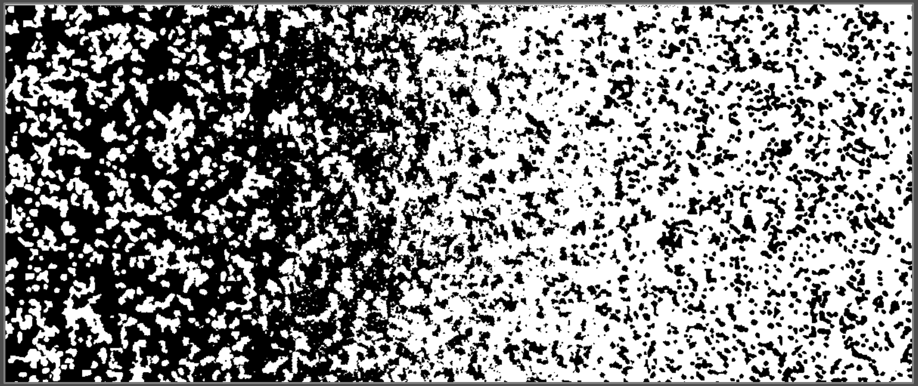
Example-Based Stipple Placement

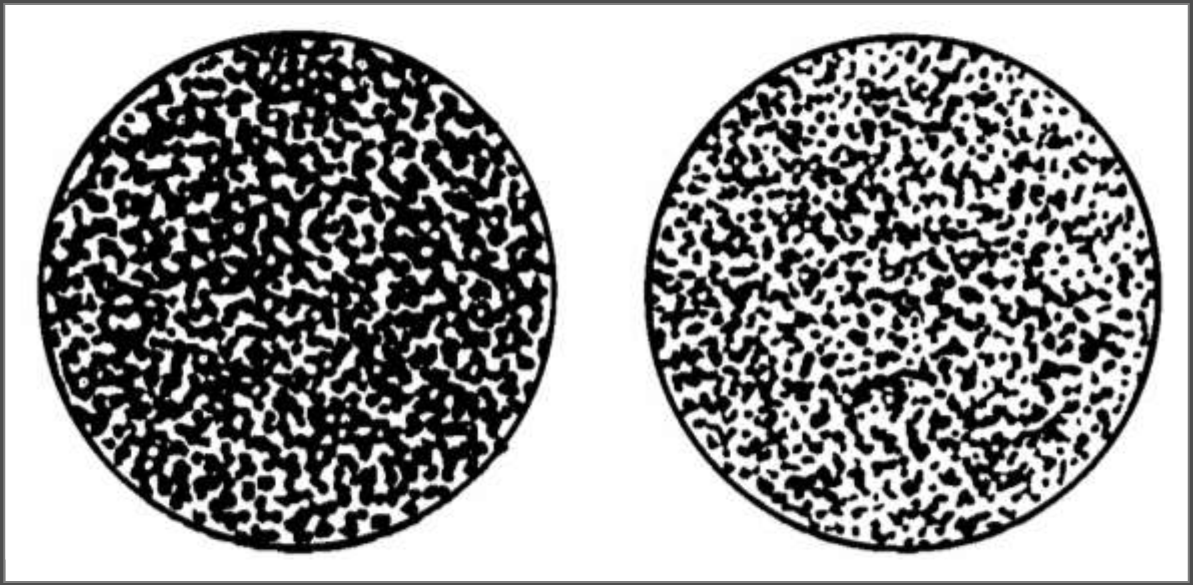




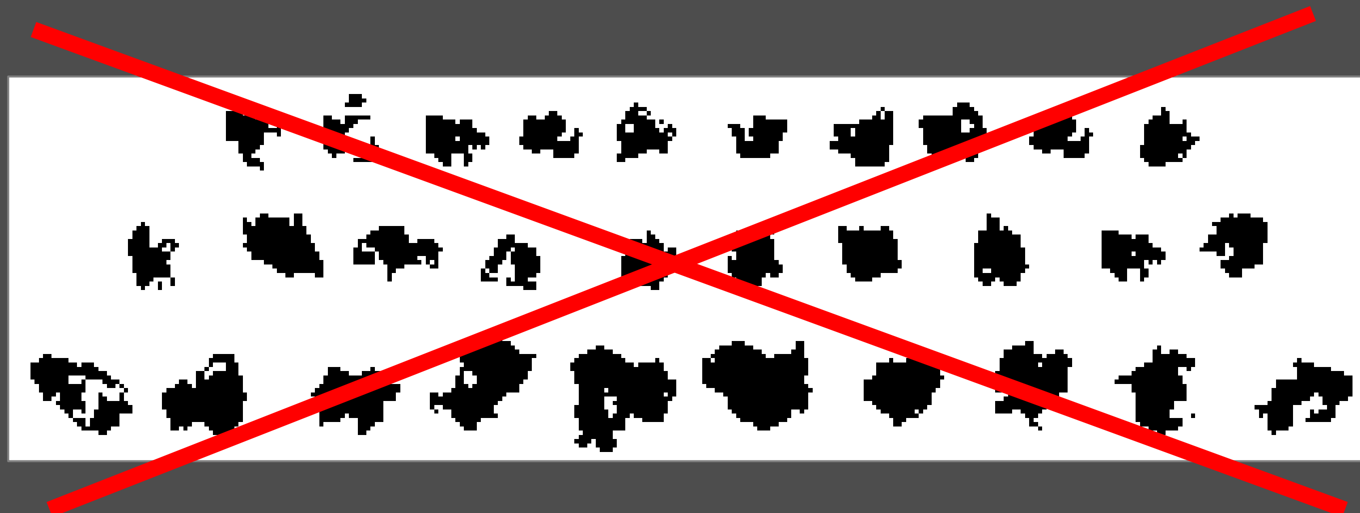
Example-Based Stippling: Problem

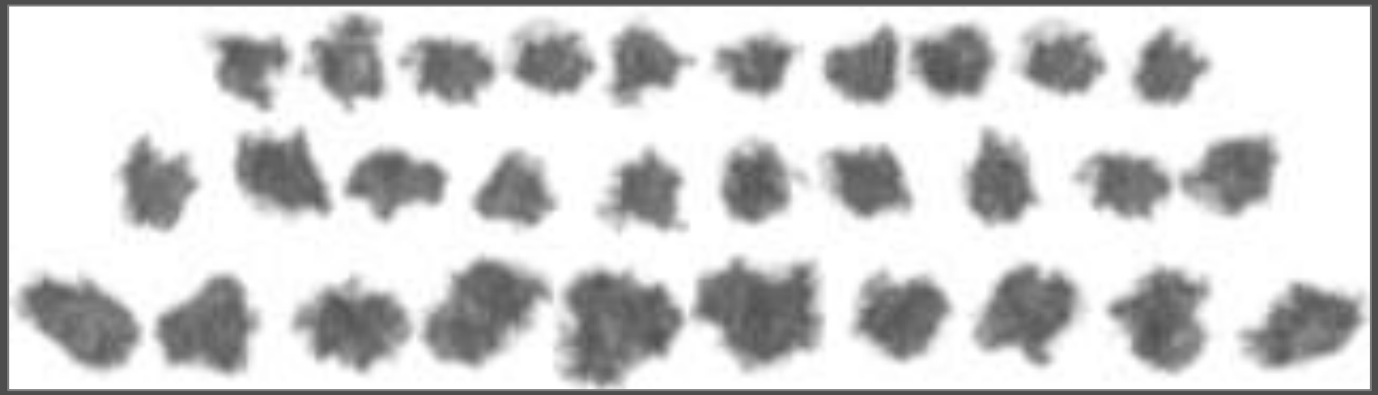






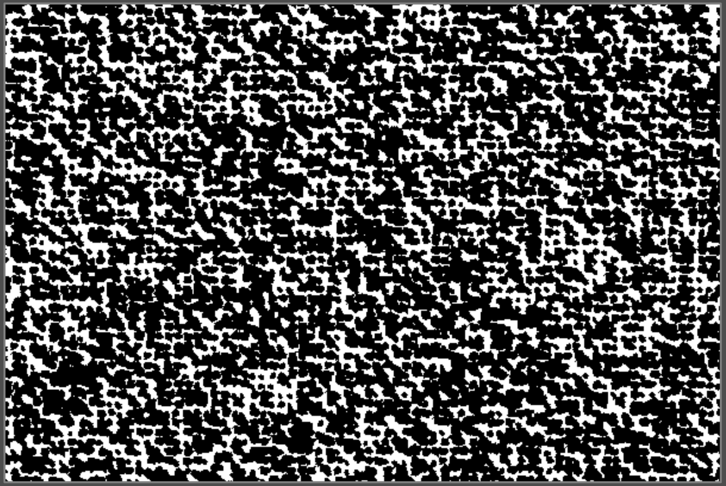
Solving the Problem: Grayscale Stippling



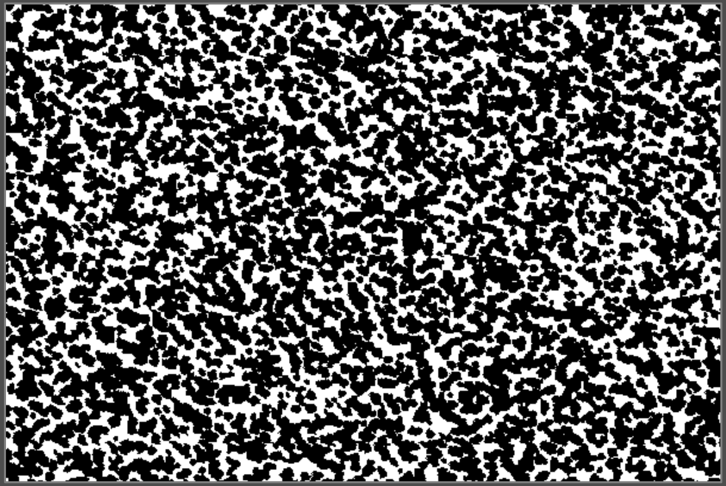


Grayscale Stippling, Halftoned Distribution

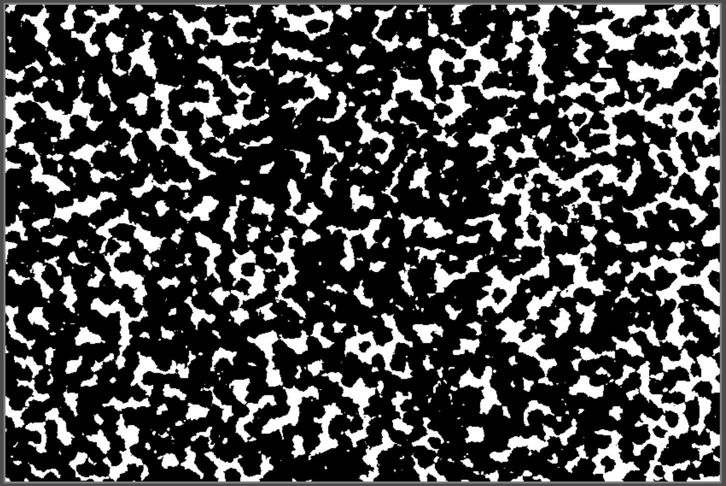
just distributed w/ halftoning



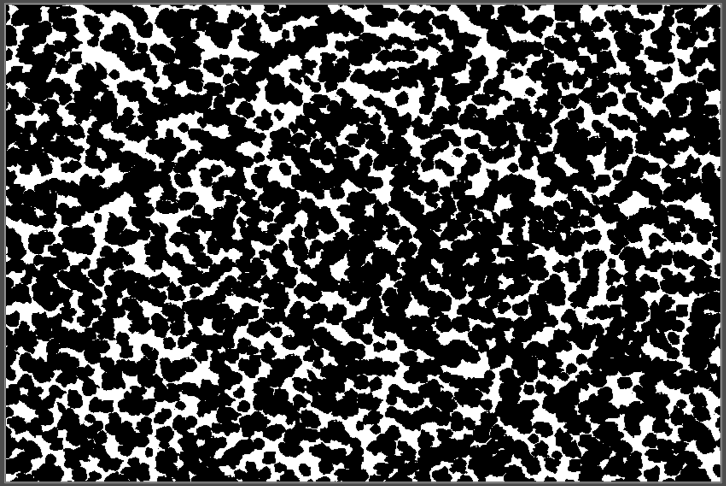
random jitter added



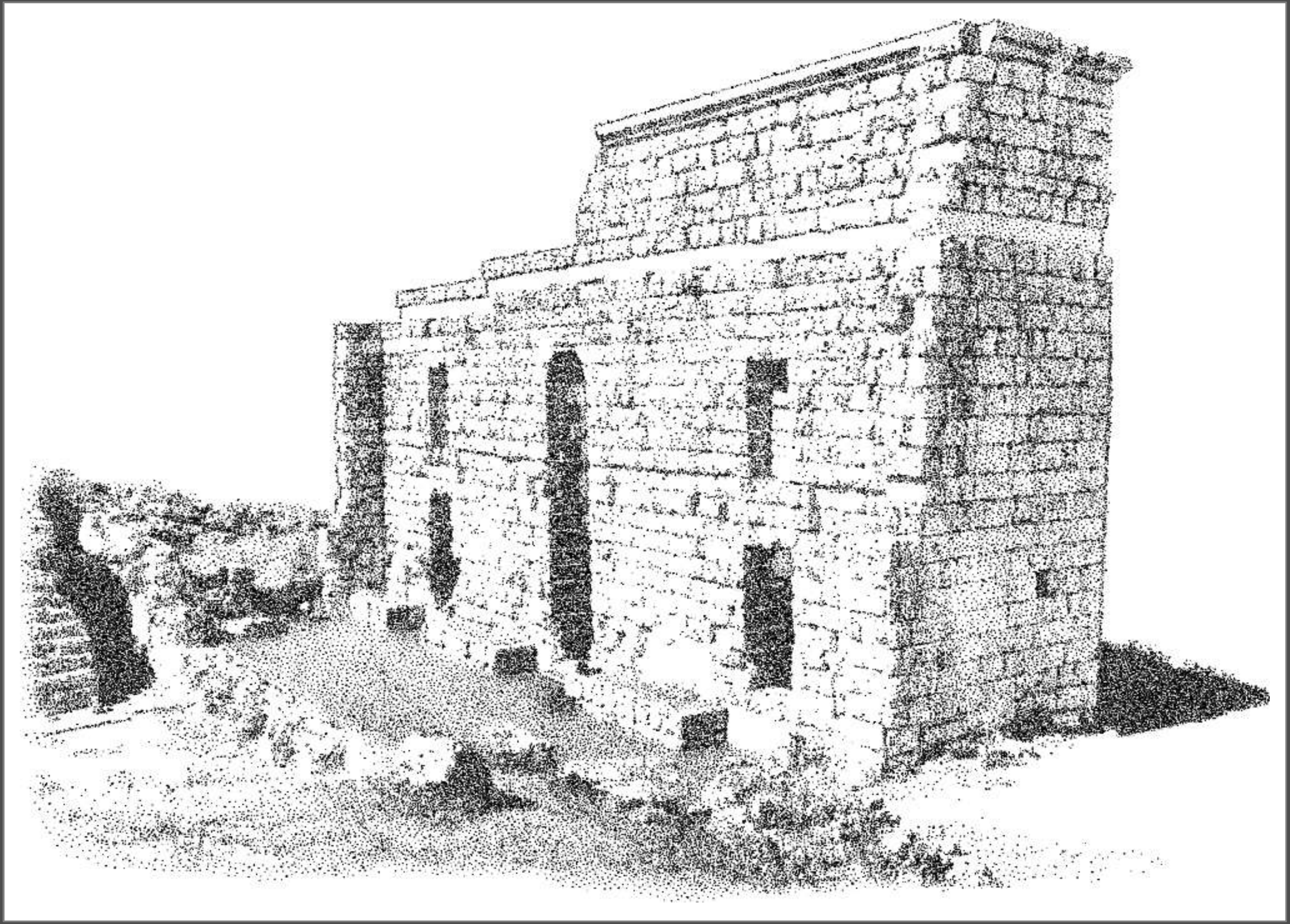
artist's stippling



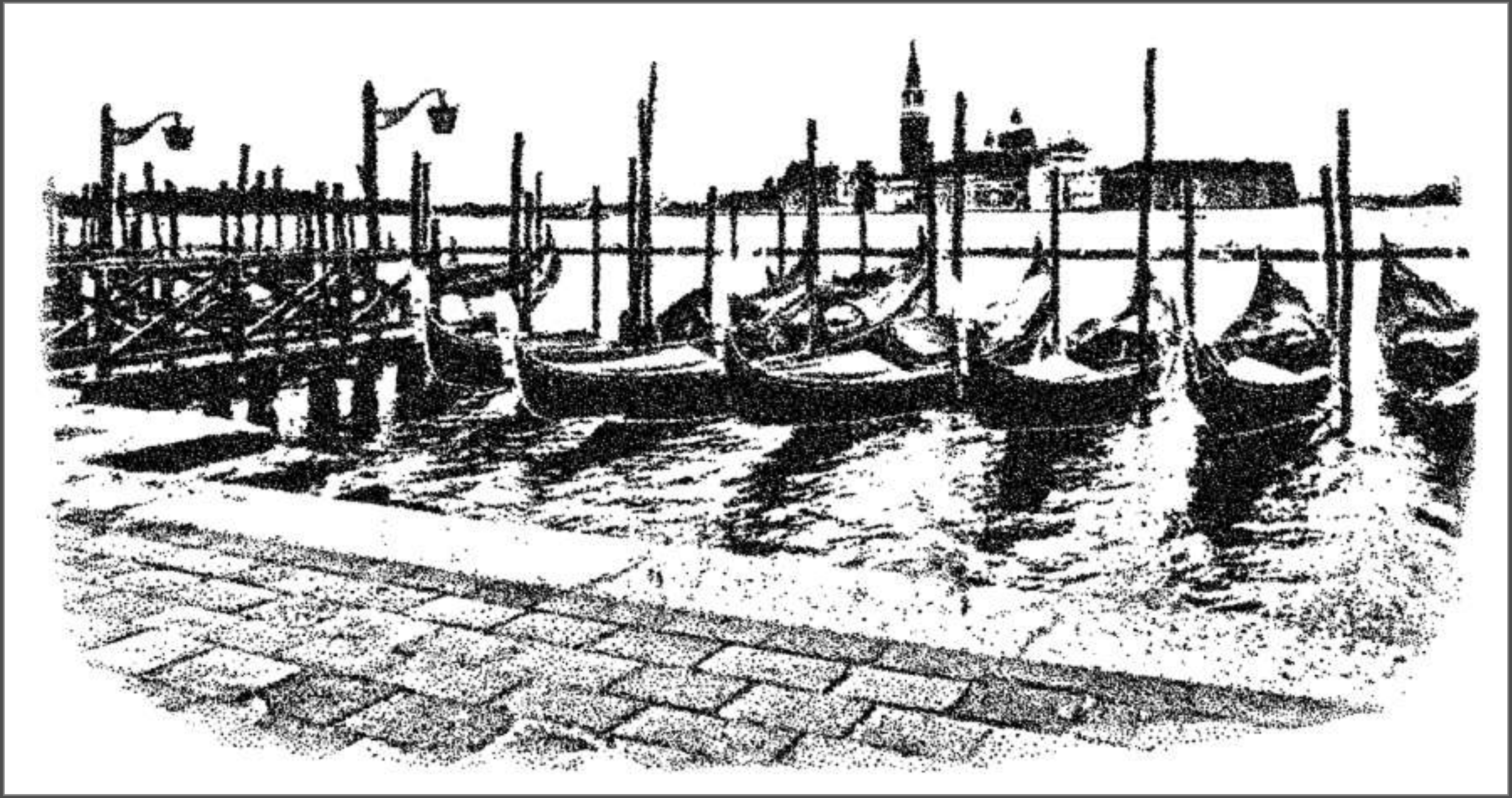
computer-generated stippling



Grayscale Stippling: Result

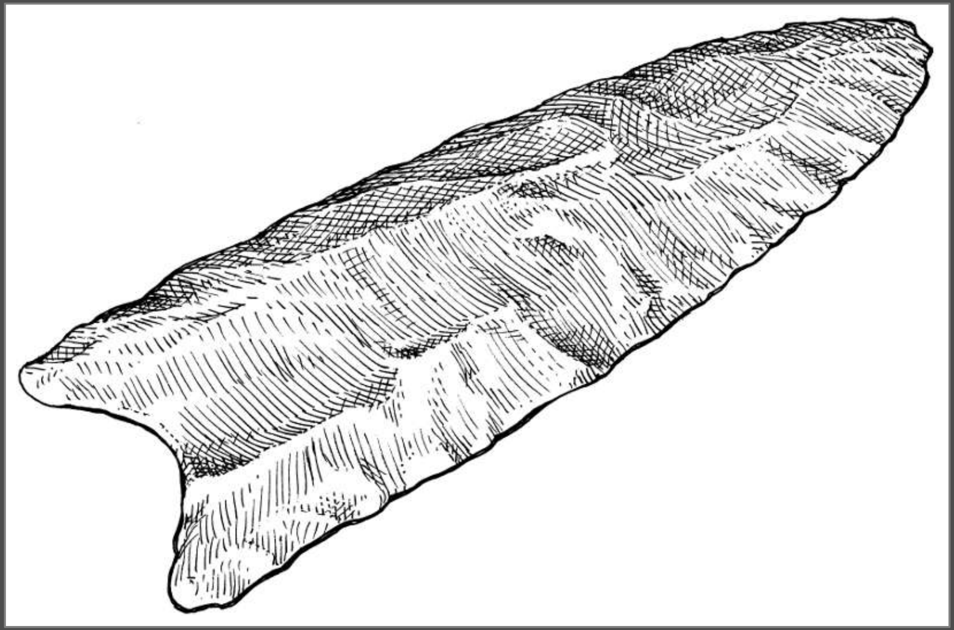


Example-Based Stippling



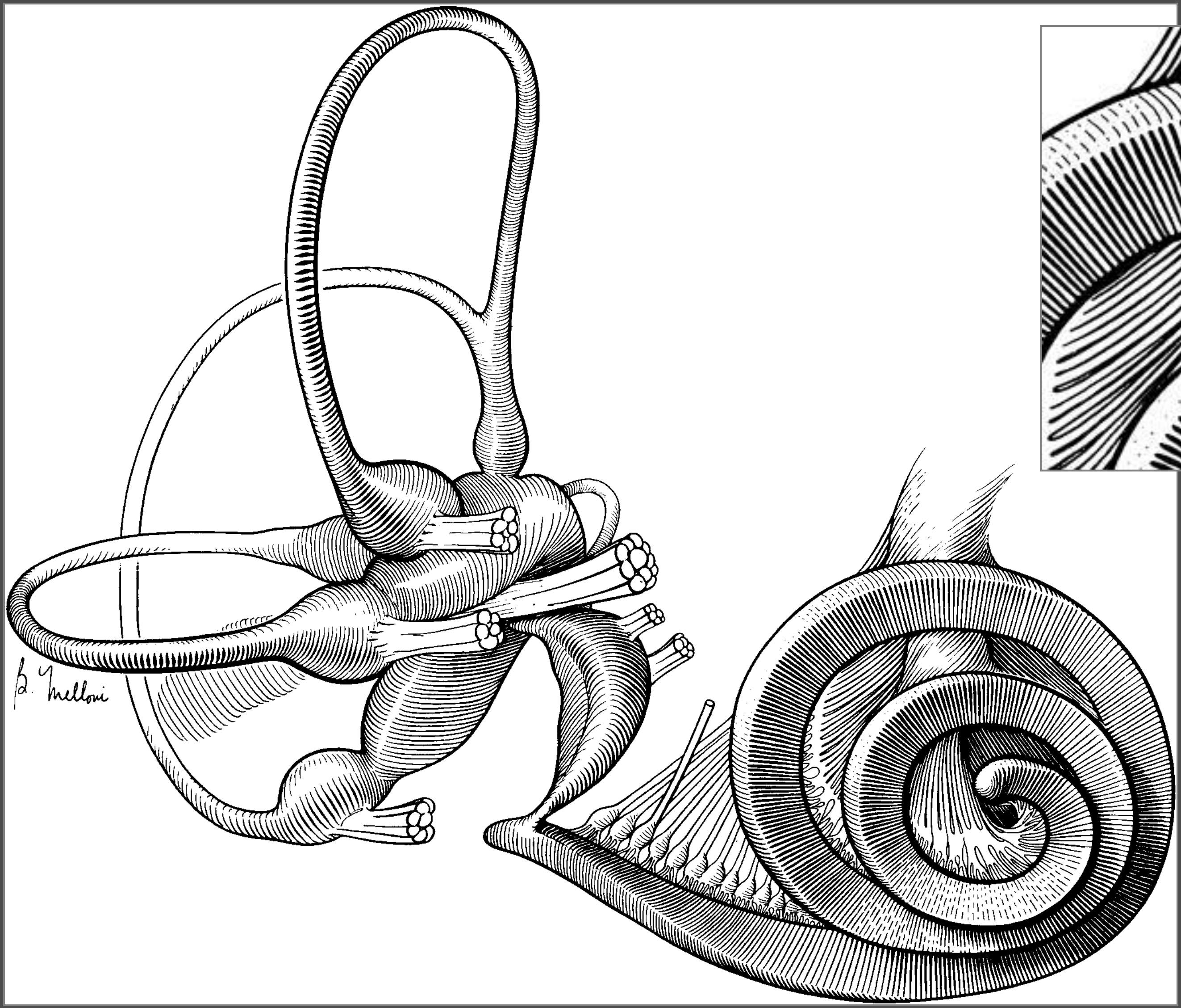
Pen-and-Ink Rendering

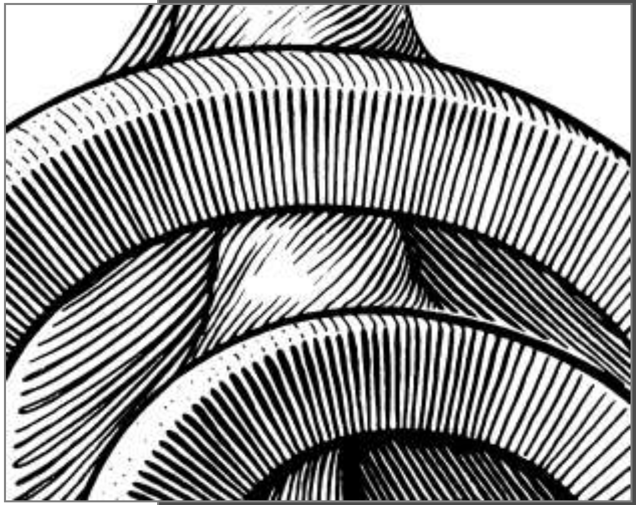
Rendering with Lines: Hatching



Traditional “Hatching” Examples

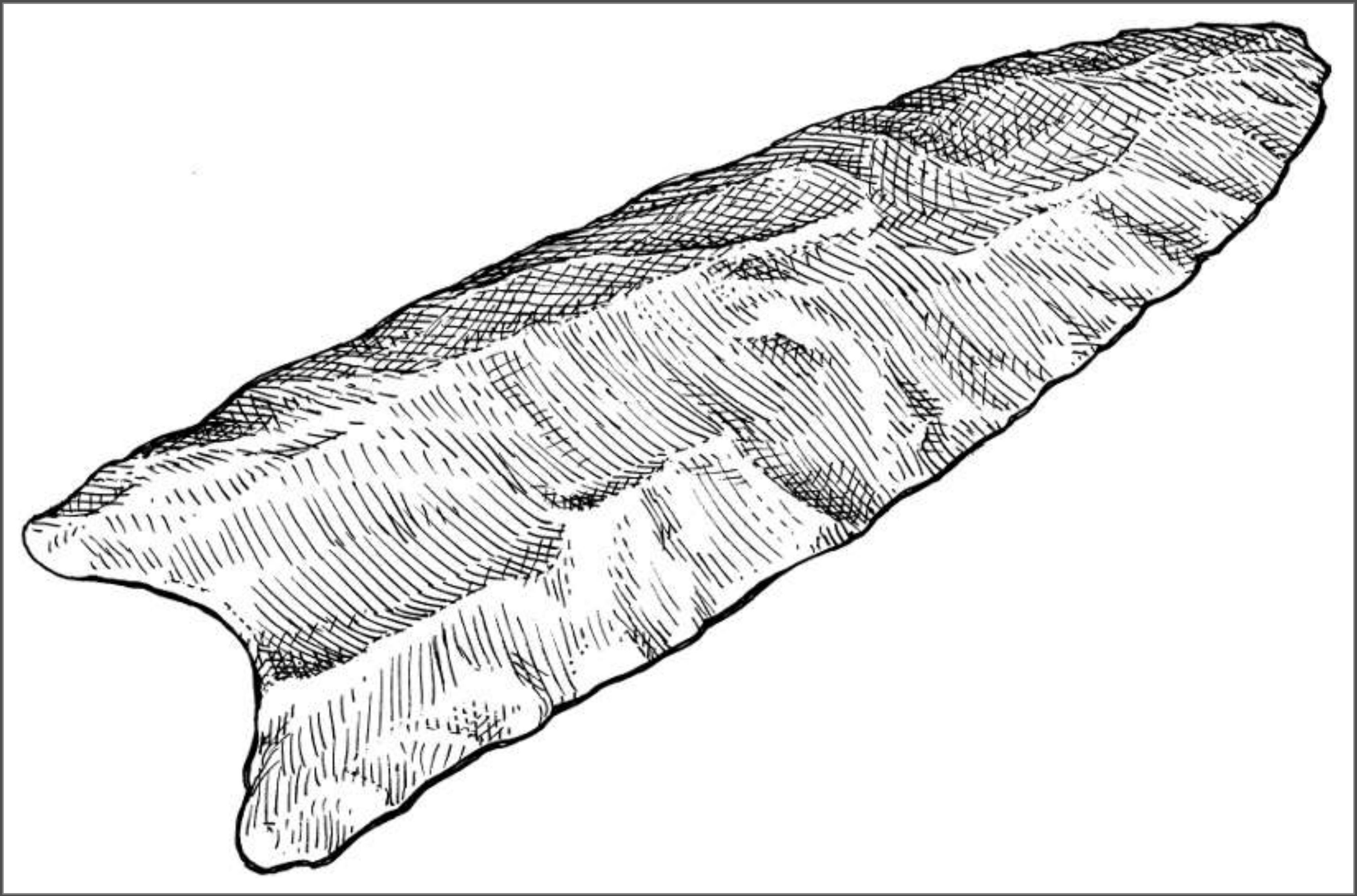
Traditional "Hatching" Examples



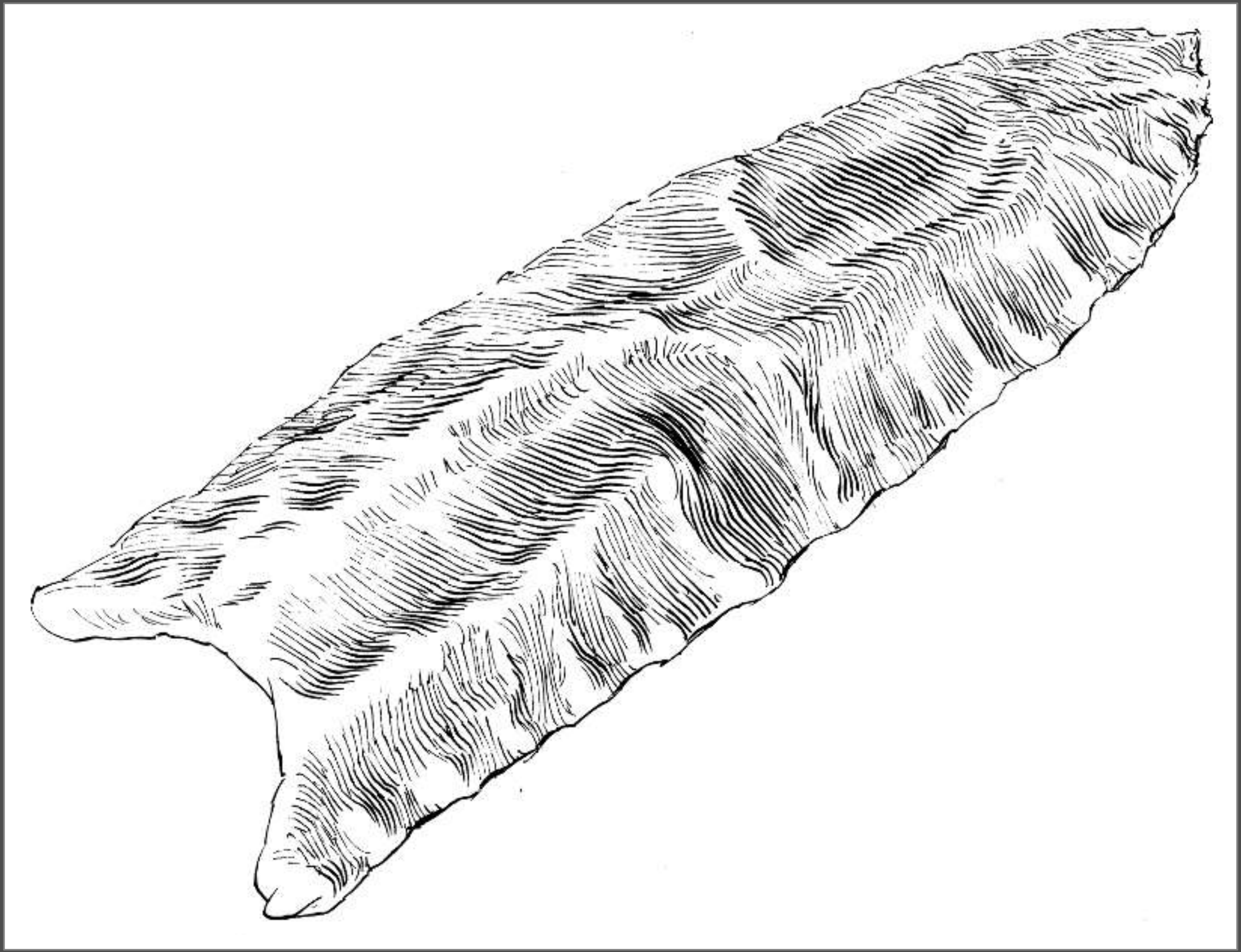


from Wood (1994)

Traditional "Hatching" Examples

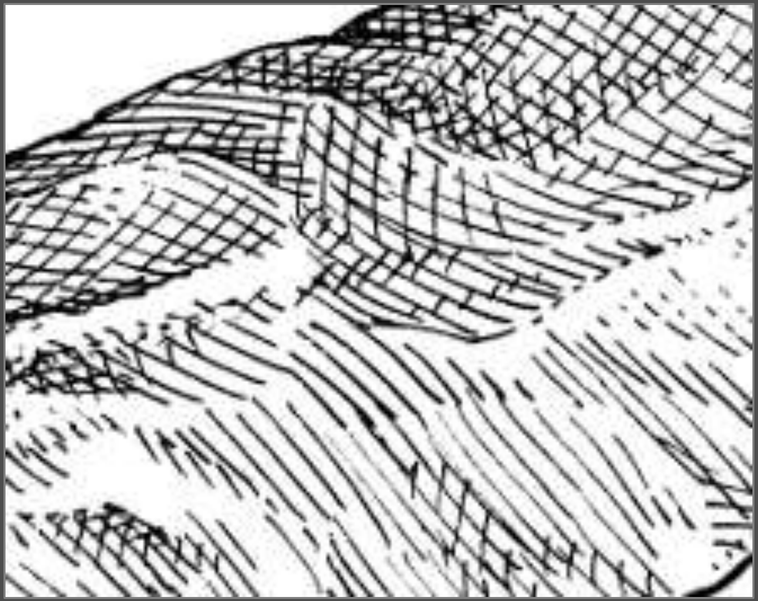


Traditional “Hatching” Examples



Rendering with Lines: (Cross-)Hatching

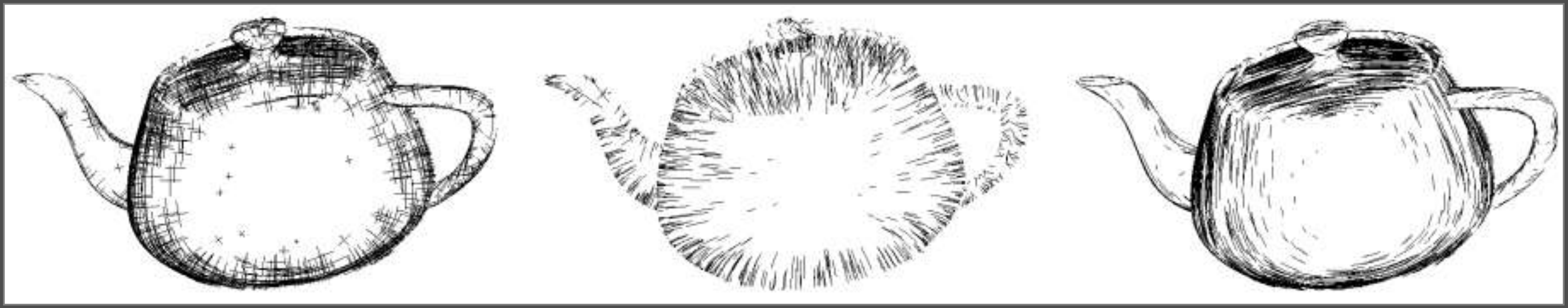
- why lines, what do the lines represent?
  - lines can convey form, shading, and material properties
  - line styles for shading
  - line paths for form
  - both for local material properties
  - double function of outlines and surface-filling lines

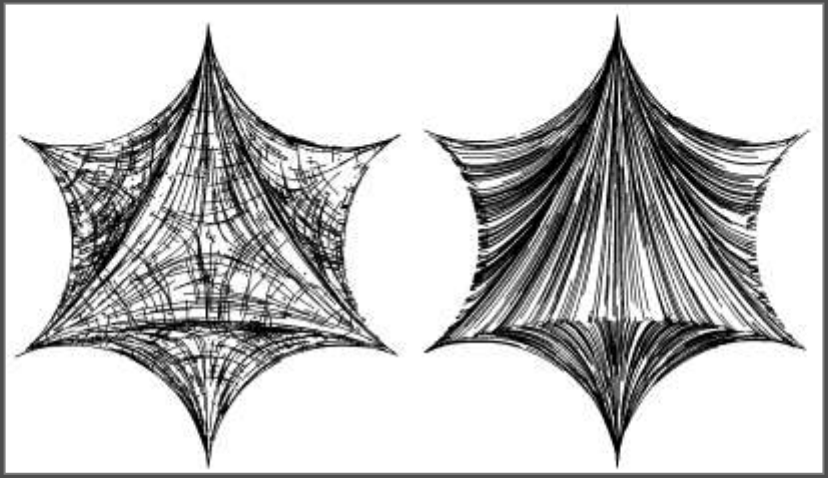


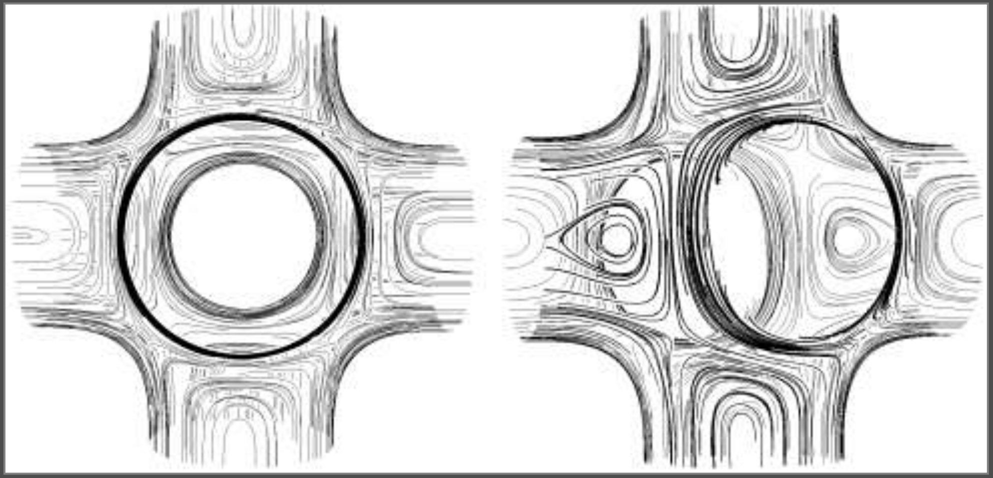
- line-based rendering in NPR
  - simple hatching (shading)
  - advanced hatching (form)
  - some 2D techniques already mentioned previously

Representing Shape and Tonal Values

- hatching of smooth surfaces
  - extract local properties of shapes: curvature directions
  - problems: HLR, distribution, meaningful placement

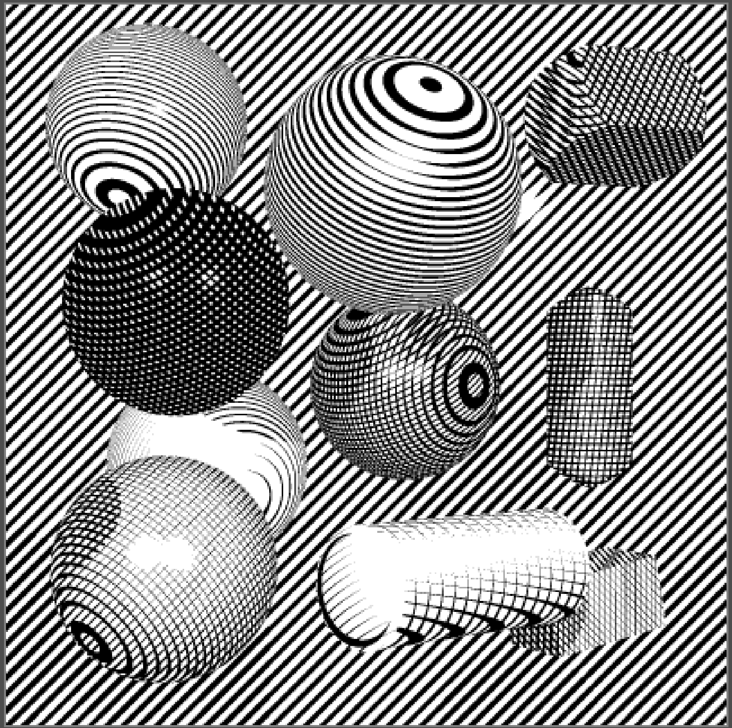




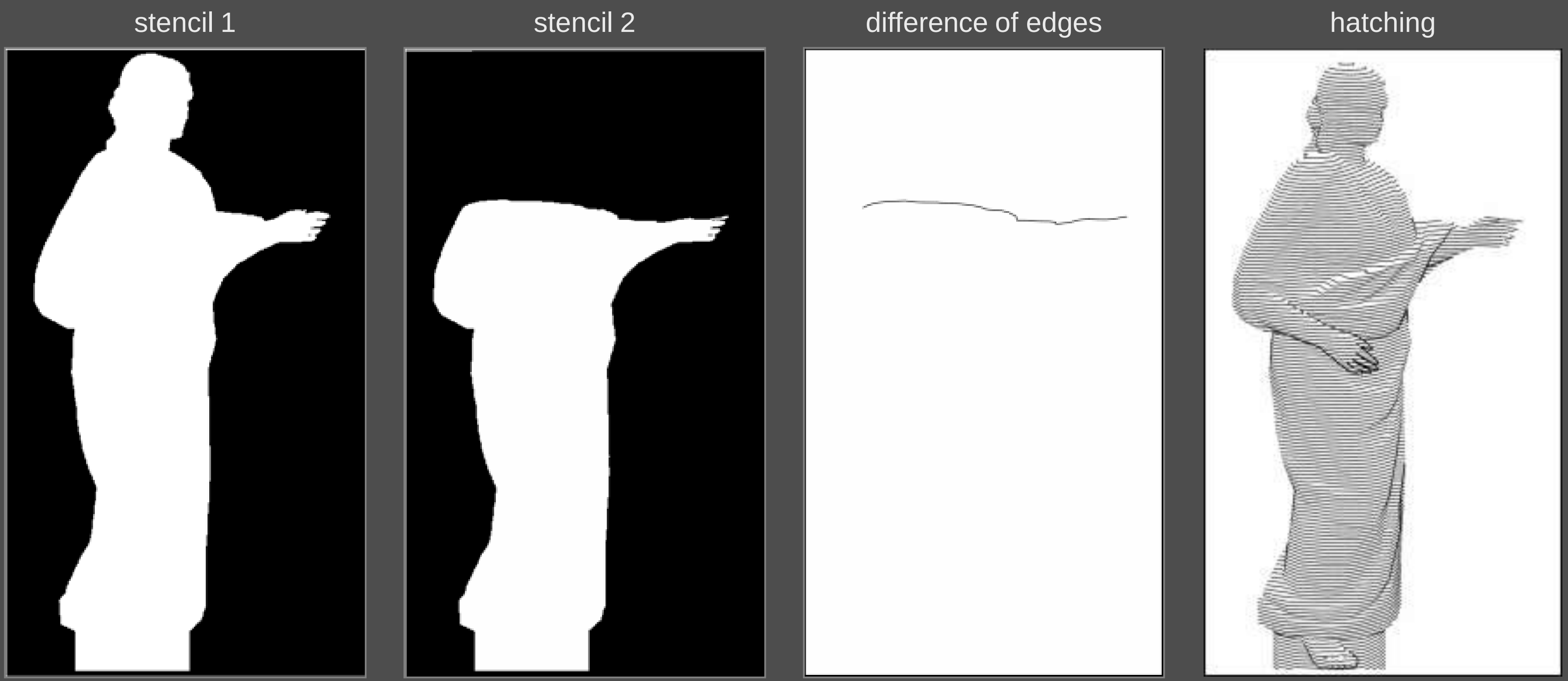


Simple Hatching Techniques

- lines as dedicated elements
  - placement in layers, shading by controlling line width
  - initially for raytracing
  - extended to mesh data



Slice-Based Hatching for Meshes

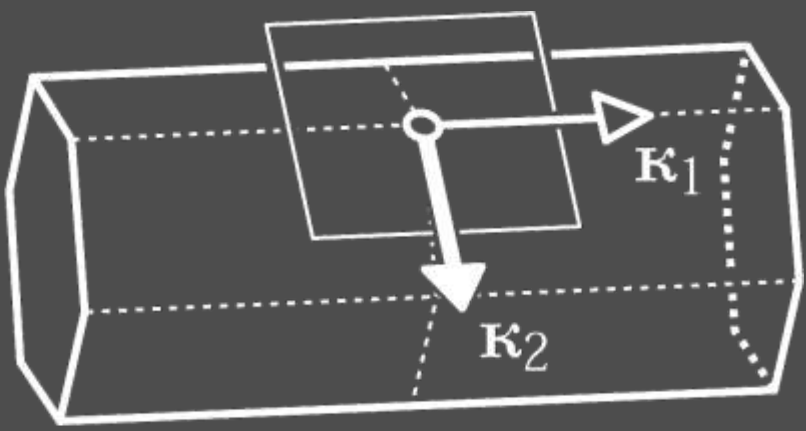


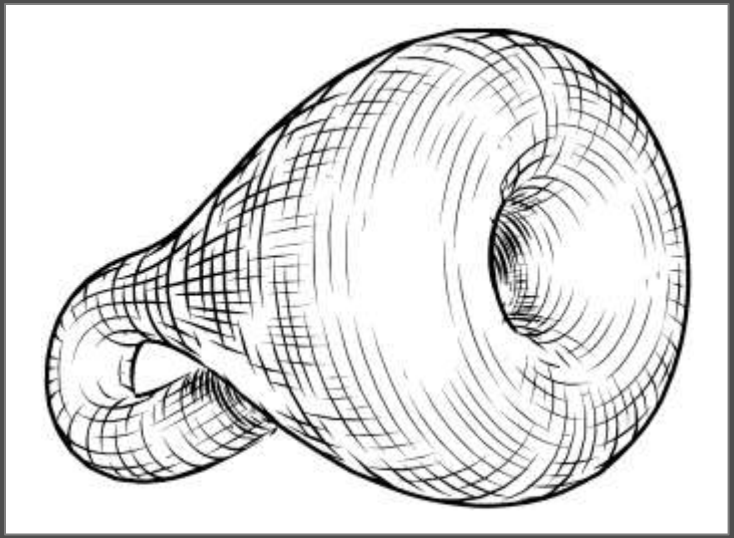
Deussen et al., 1999

- pro: avoids line thickening
- con: expensive rendering (once per cut)

Hatching to Represent Smooth Surfaces

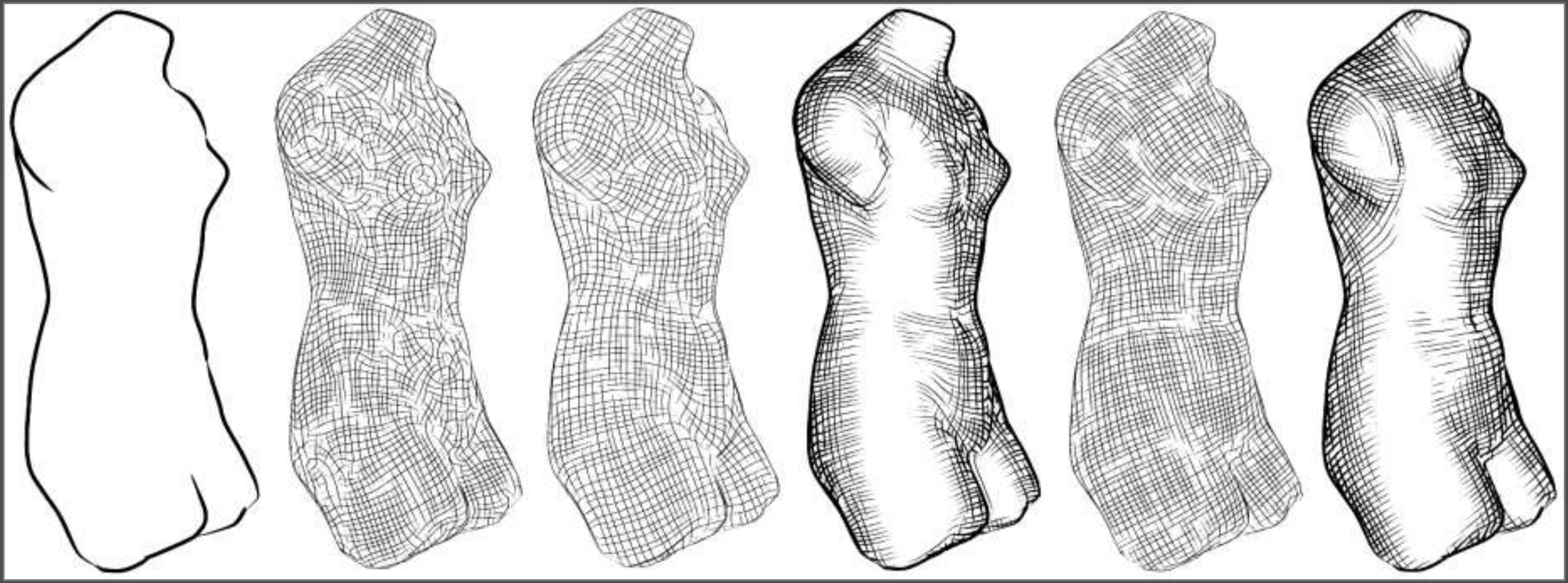
- general concept: meshes represent smooth surfaces
  - extract properties to the "ideal" smooth surfaces
  - principle directions (directions of principle curvatures)
  - direction field, use to guide line placement
  - direction field smoothing to reduce artifacts
  - projection to 2D, trace lines in 2D





Hatching to Represent Smooth Surfaces

- the process illustrated:

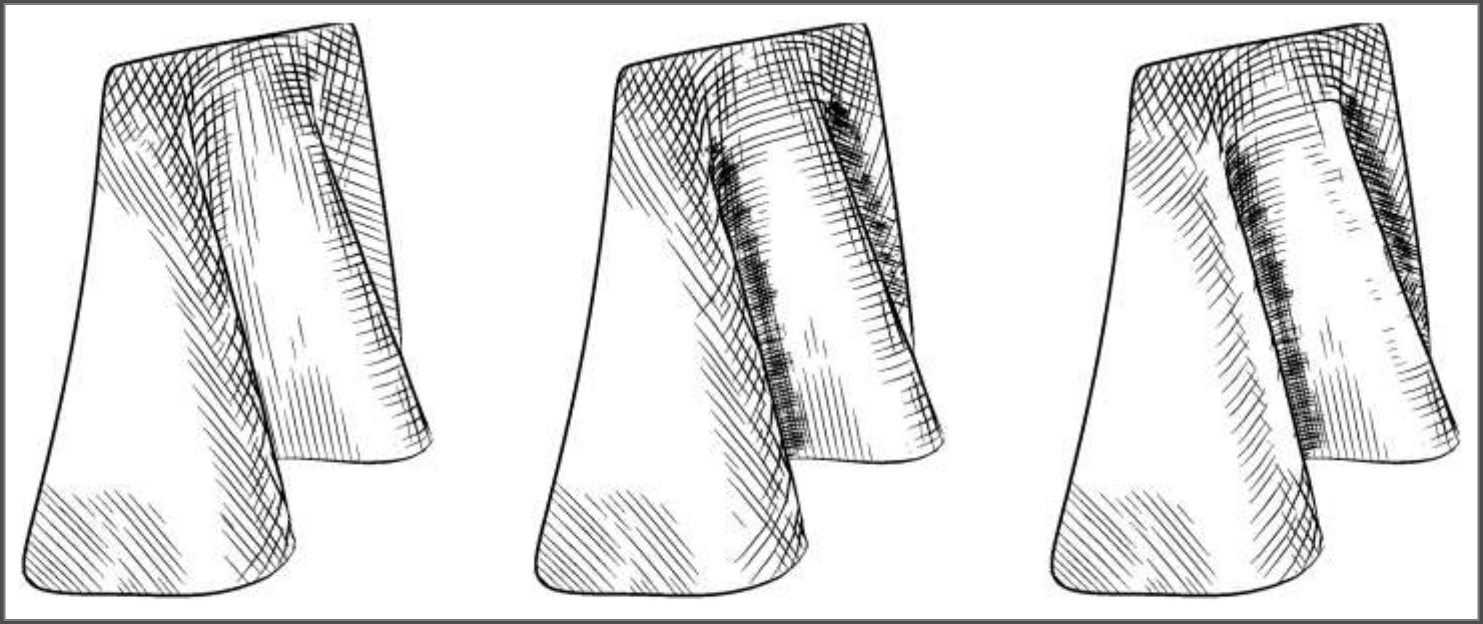


silhouette only | direction field | smoothed direction field | resulting hatching | direction field smoothed further | resulting hatching

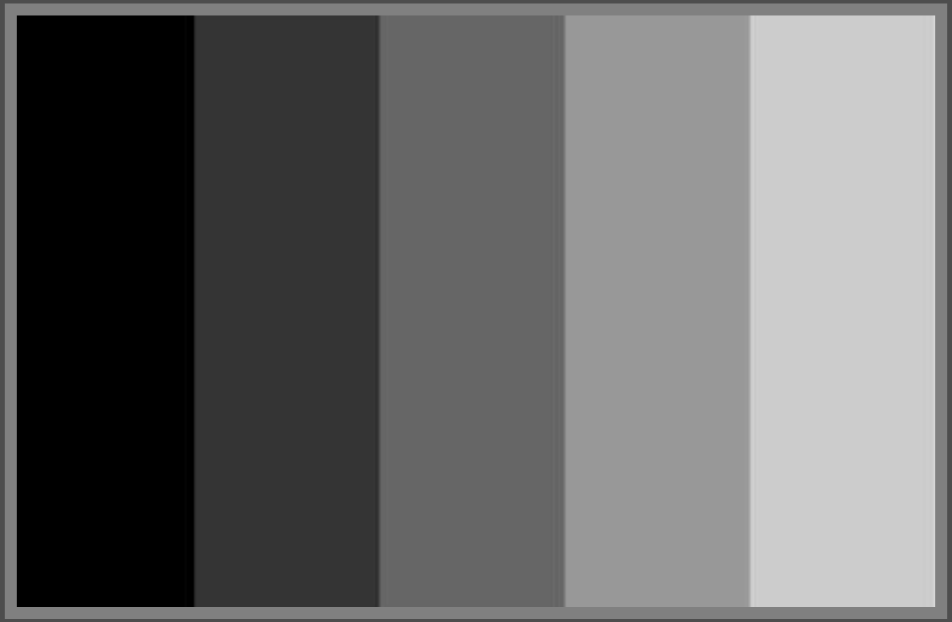
- smoothing based on stability of curvature directions and the variance of directions in local neighborhood

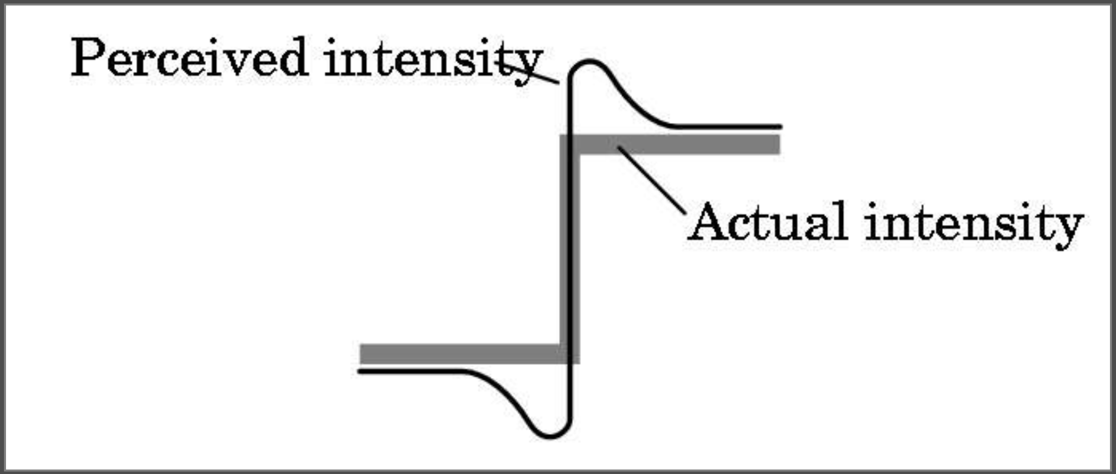
Cross-Hatching and Lighting Visualization

- cross-hatching: using more than one direction
- show illumination by no hatching, single hatching, and cross-hatching
- emphasize illumination discontinuities: undercuts and Mach bands

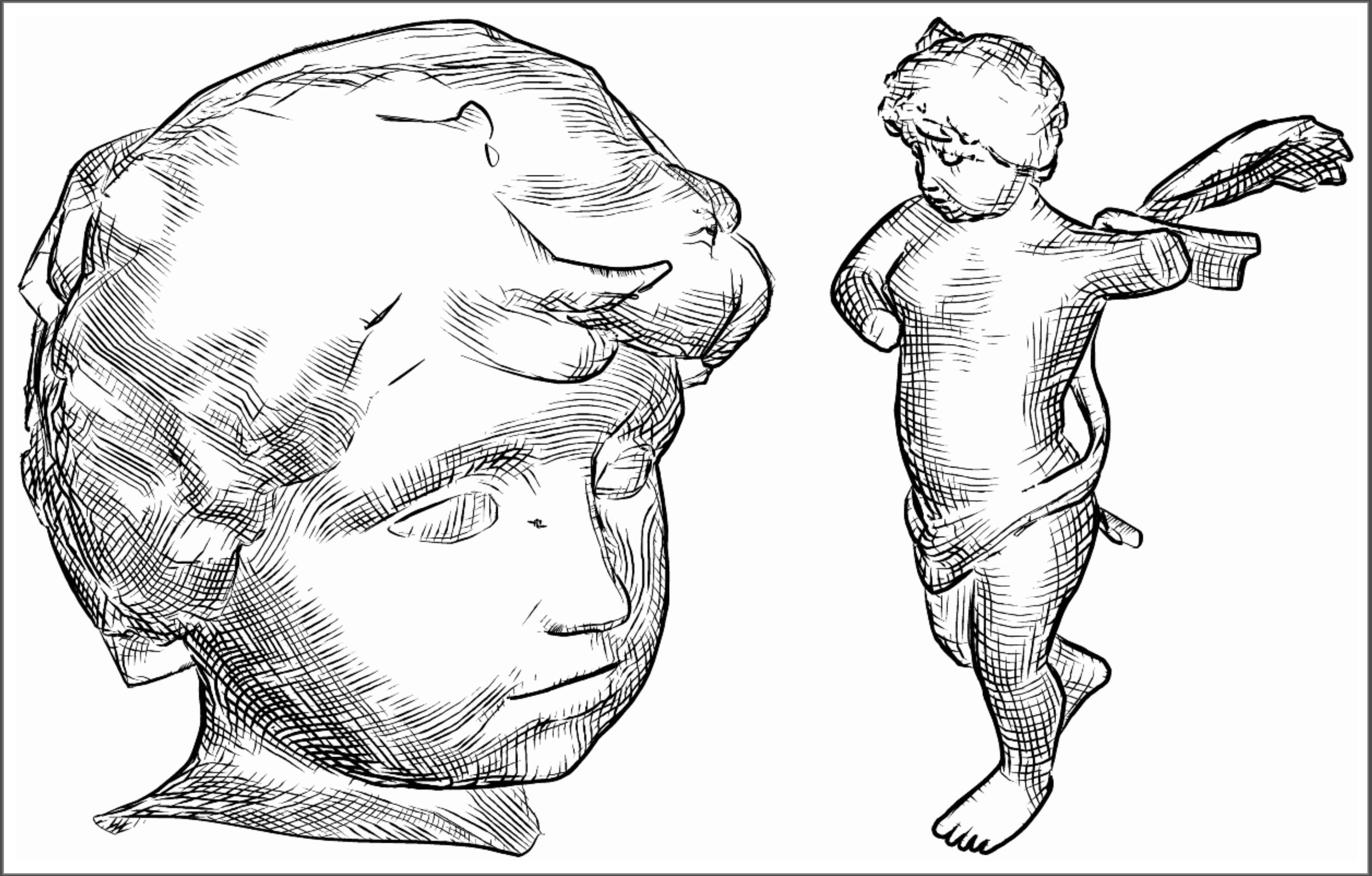


Side Note: Mach Band Effect

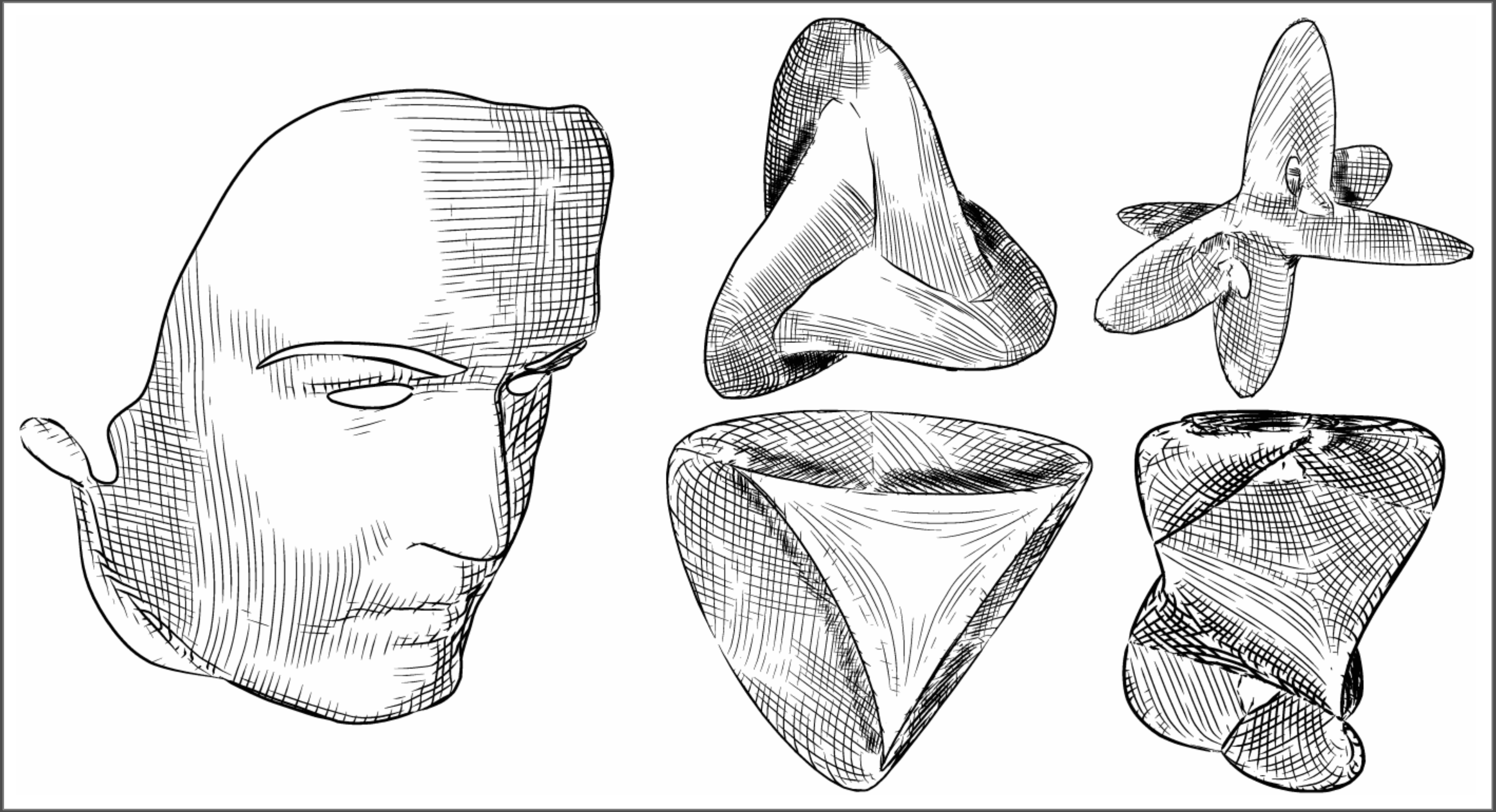




Examples:



Examples:



3D Hatching – “High Quality Hatching”

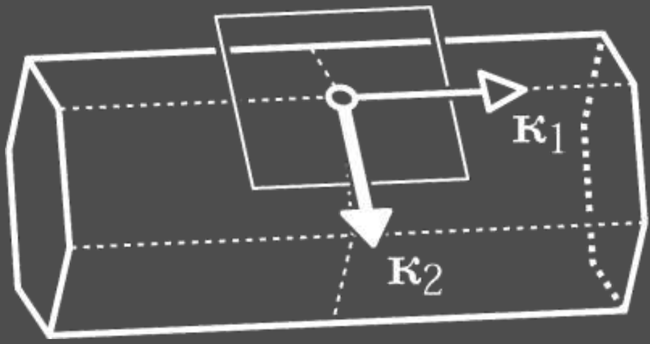
- previously: line tracking in 2D
  - projection necessary for each frame (interactivity)
  - shower-door effect due to 2D processing
- idea: line tracking on the 3D surface
  - no 2D effects, can be pre-computed
  - interactive exploration and stylization
  - but: hidden line removal necessary
- now:
  - general process
  - line shading
  - color hatching

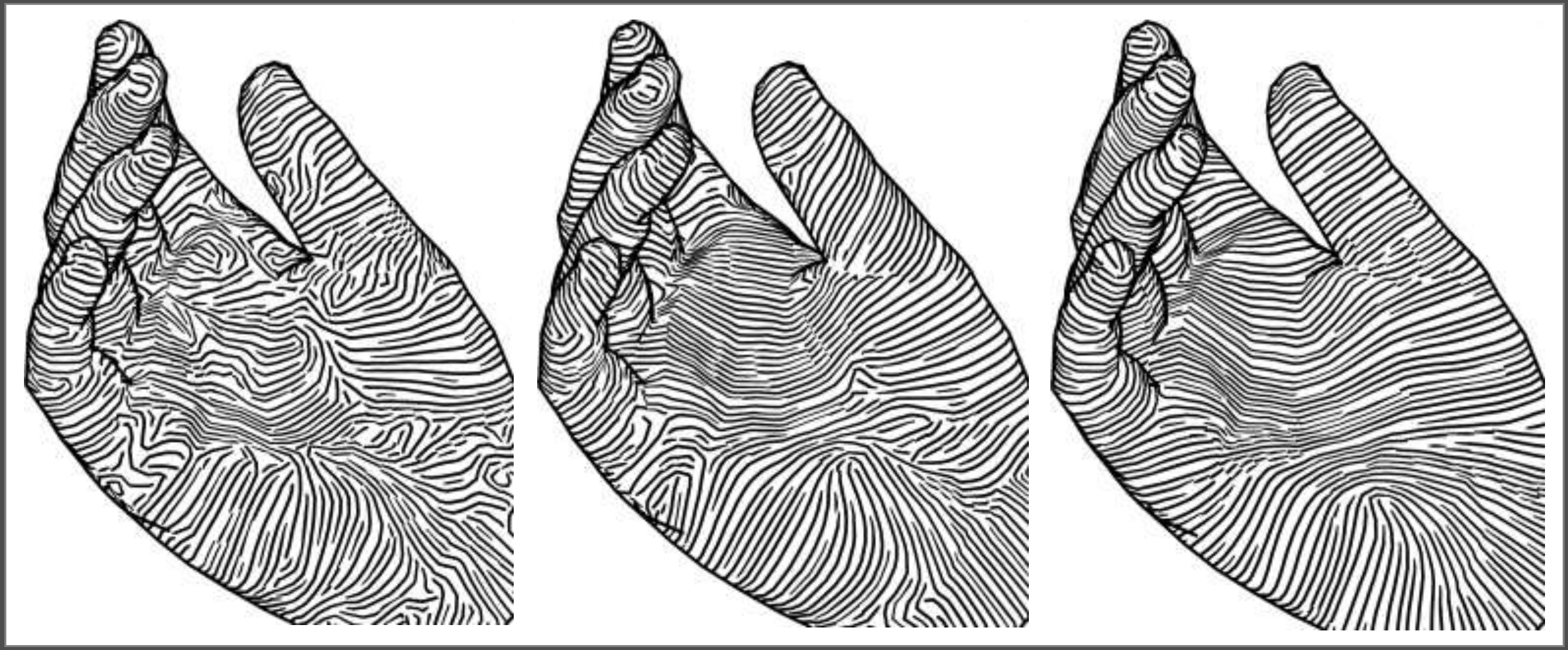
3D Hatching – General Process

- derive direction field based on principal curvatures
- optimize direction field to reduce artifacts
- compute streamlines following direction field
- perform hidden line removal
- compute line shading according to formula
- correct for perceived brightness of 3D hatching lines
- (repeat for several layers if desired)
- render to screen or render to vector graphics

3D Hatching – Deriving Vector Field

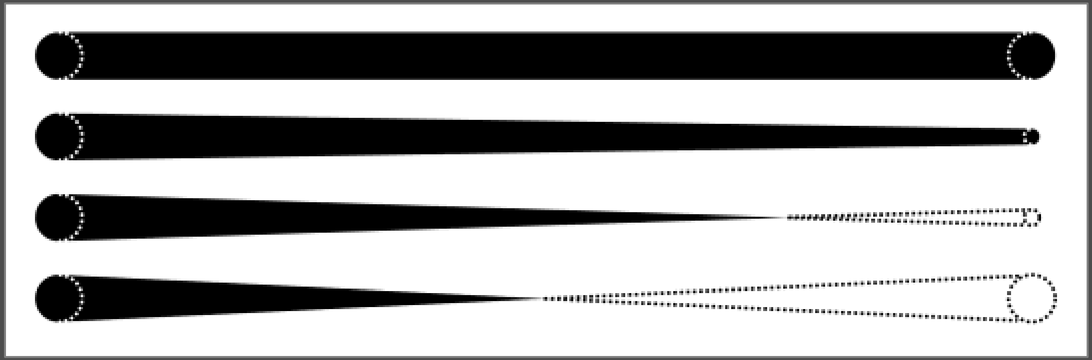
- based on principle directions
- chose the strongest absolute
- apply field smoothing based on local neighborhood

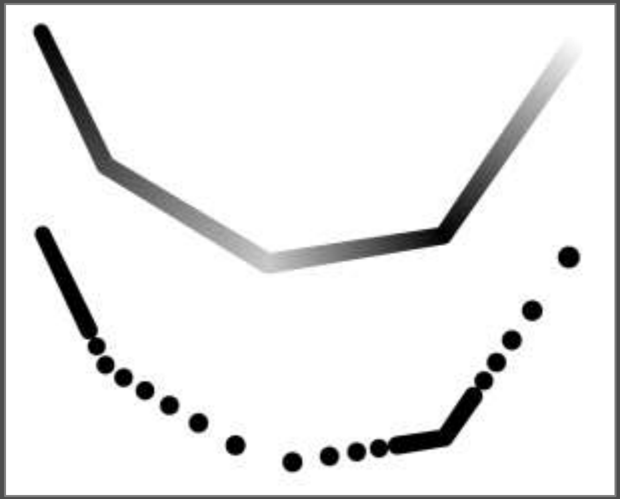




3D Hatching – Line Control

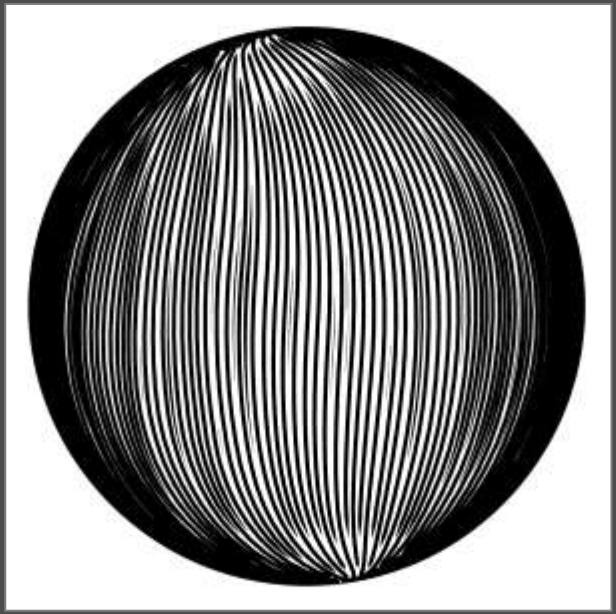
- problem 1: polyline tessellation controls line ends
- solution: negative line widths → line ends not restricted to vertices anymore
- problem 2: line shading also to be displayed in true black-and-white rendering
- solution: use line stippling to portray gray scale

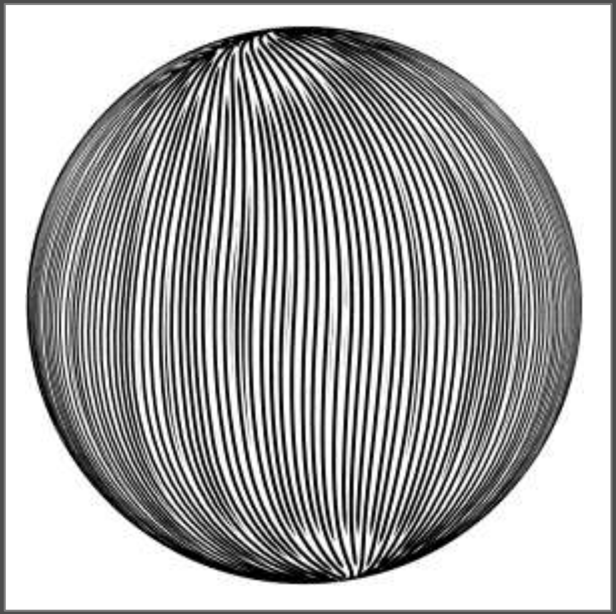




3D Hatching – Perceived Brightness

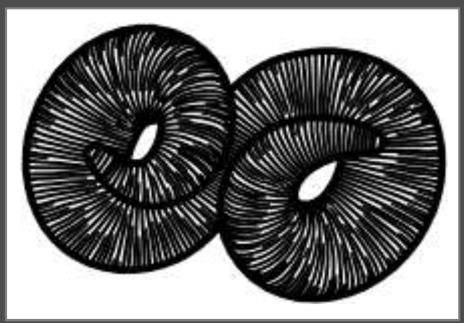
- new problem from hatching lines in 3D:
  - lines on surfaces facing the viewer are nicely spaced
  - lines on surfaces along the viewing direction are not
- this needs to be corrected:



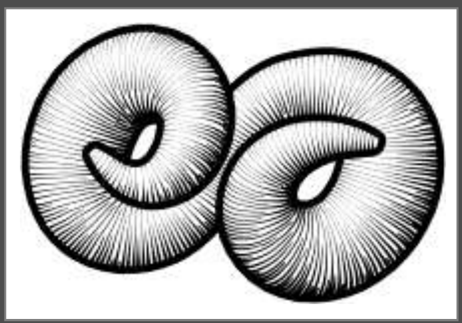


3D Hatching – Line Shading

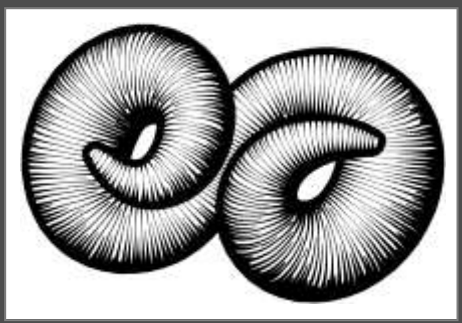
- control of line parameters using direct shading function, based on illumination and rim effect
- interactive control at run-time (virtual machine)



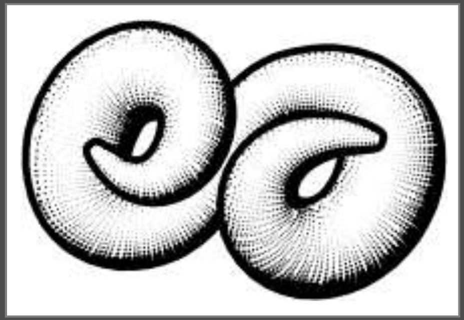
line width = 1
line density = 1



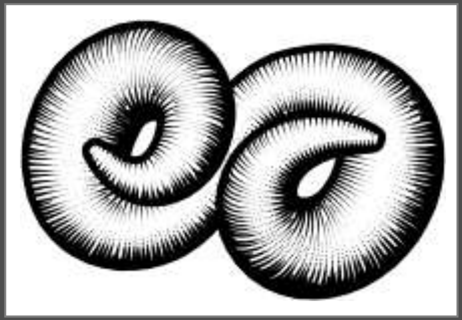
line width = *light*
line density = 1



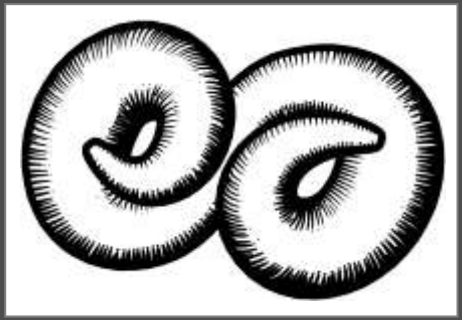
line width = *light* + *rim*
line density = 1



line width = *light* + *rim*
line density = *light*

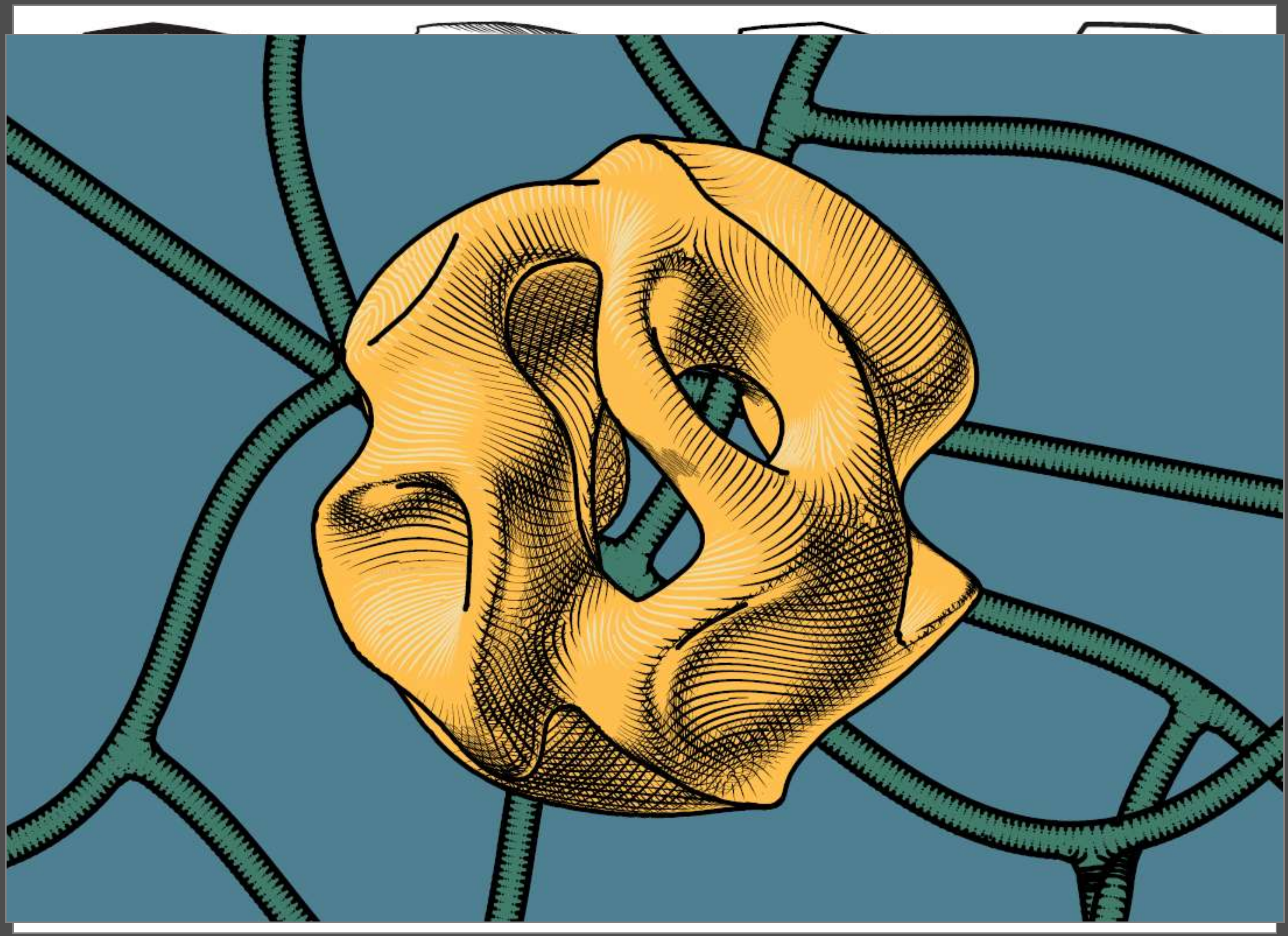


line width = *light* + *rim*
line density = 5 *light*



line width = *light* + *rim*
line density = (5 *rim*)^{50}

High Quality Hatching: Examples



Pen-and-Ink Rendering

Summary

Pen-and-Ink and B/W Rendering

- properties of black-and-white NPR techniques
  - only two “colors” (or spot colors): needed in printing
  - convert “normal” images to b/w images
  - render 3D shapes directly in b/w
  - get inspired by artistic techniques and illustration
  - precision and detail, abstraction and emphasis
  - can be computationally expensive (what to display?)
- many techniques
  - (artistic) halftoning and screening
  - binary shading
  - stippling and hatching
  - …