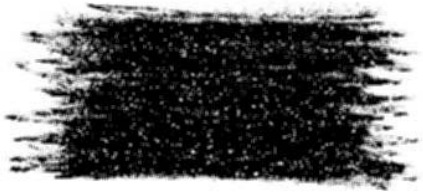
Computer Graphics

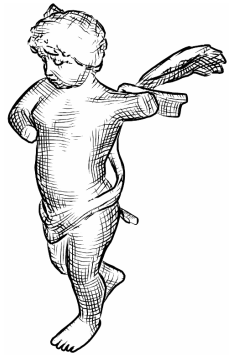
Non-Photorealistic Rendering

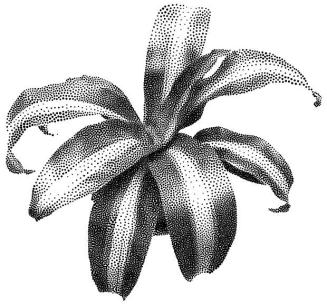
Tobias Isenberg

Overview

- introduction
- pencil rendering
- watercolor rendering
- oil paint
- stippling
- hatching
- deep learning







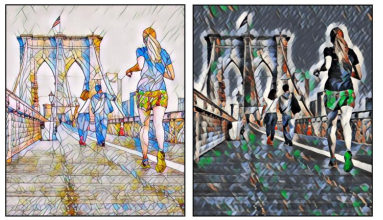


Photo “realistic” rendering?

Photo “realistic” rendering?

Photo “realistic” rendering?

Photo “realistic” rendering?

Ralph Goings:
Hot Fudge Sundae Interior, 1972
(oil on canvas)

Photorealism is a recent development

- photographic camera
  - first permanent picture in 1826
  - video cameras and digital cameras dominate today's visual world
  - "dictate of the photographic camera"

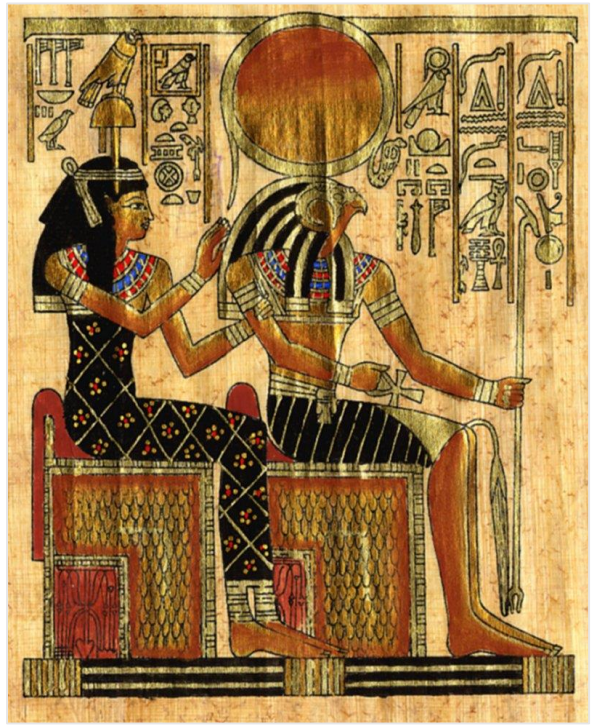
Other forms of depiction: Painting

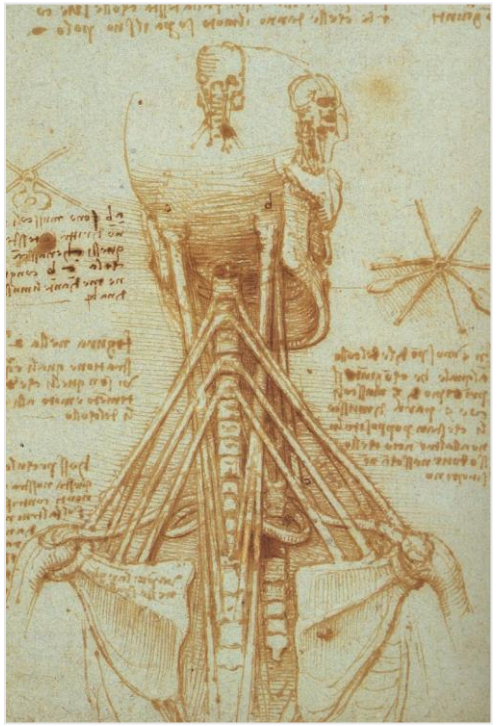
- up to 32 000 years old (French cave paintings)
- dominated visual depiction up to quite recently

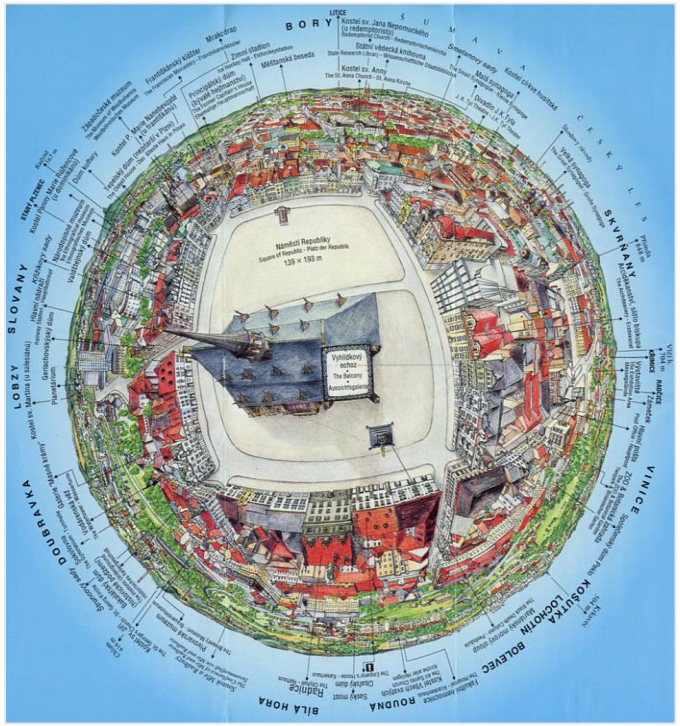


Other forms of depiction: Drawing etc.

- less visually vivid depiction, possibly with color
- abstraction and emphasis

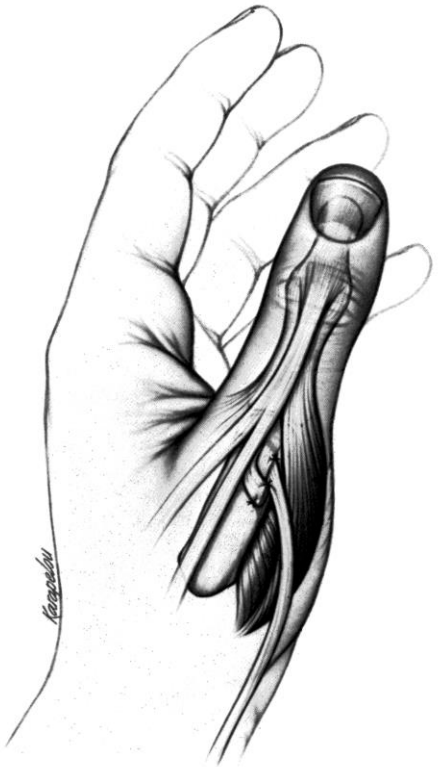


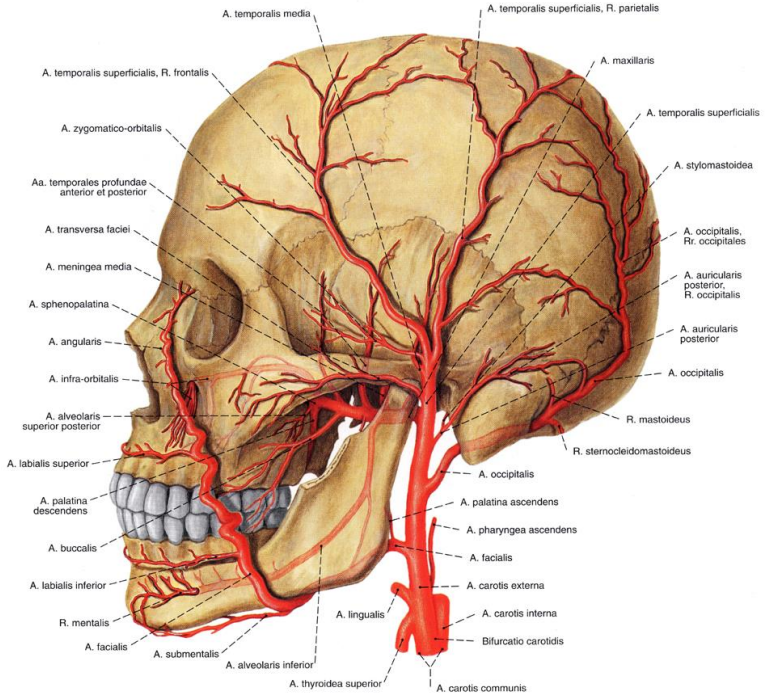


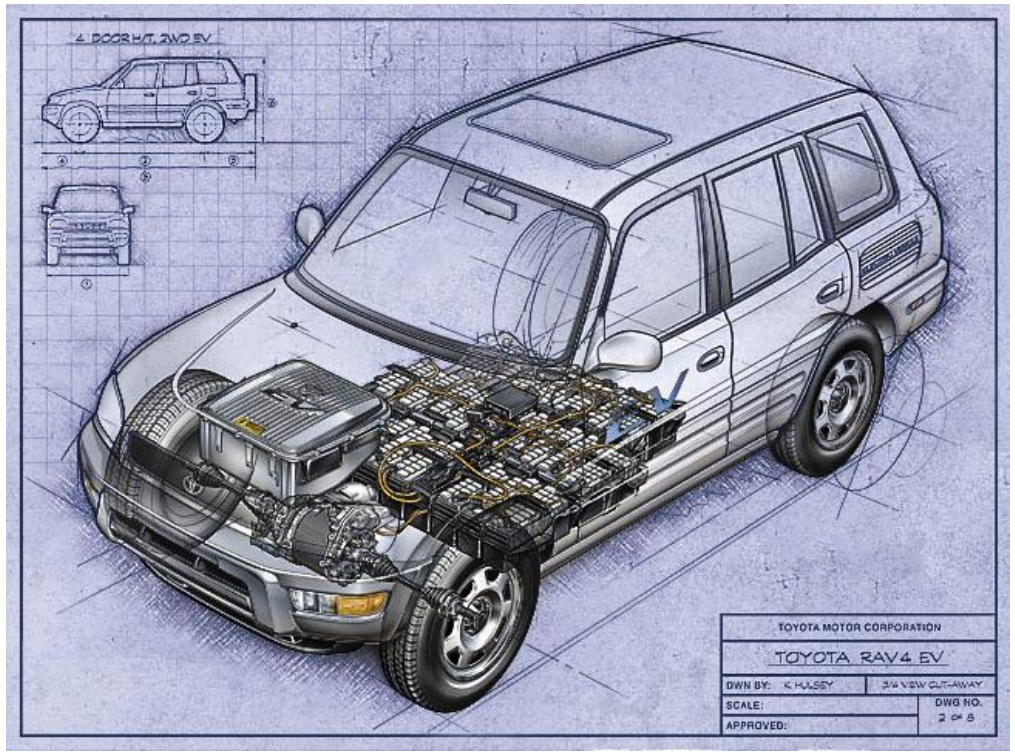


Other forms of depiction: Illustration

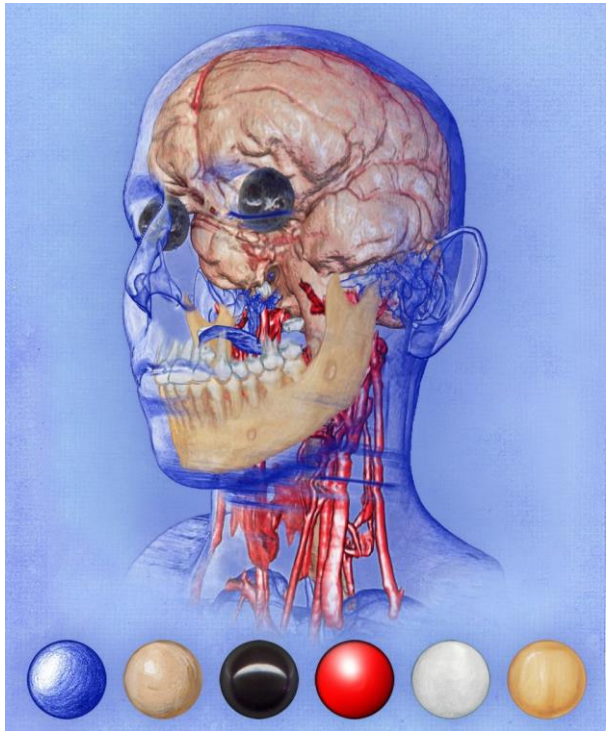
- uses abstracting forms of depiction; e.g., drawing techniques
- e.g., in medical and technical contexts

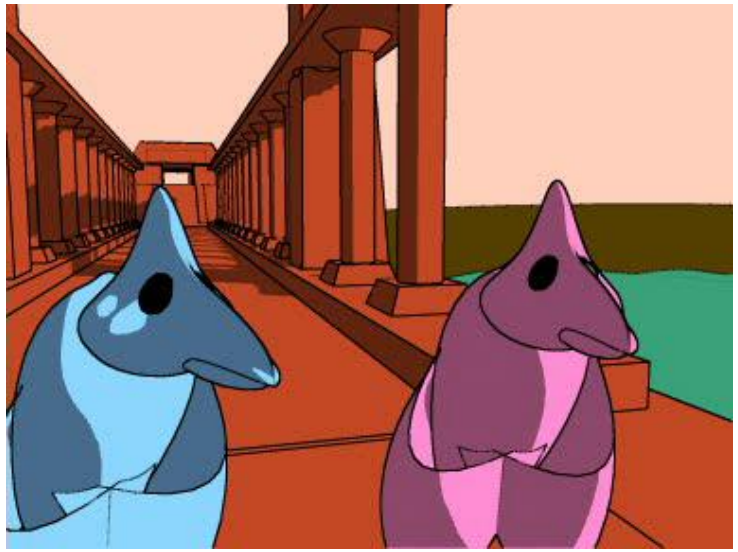


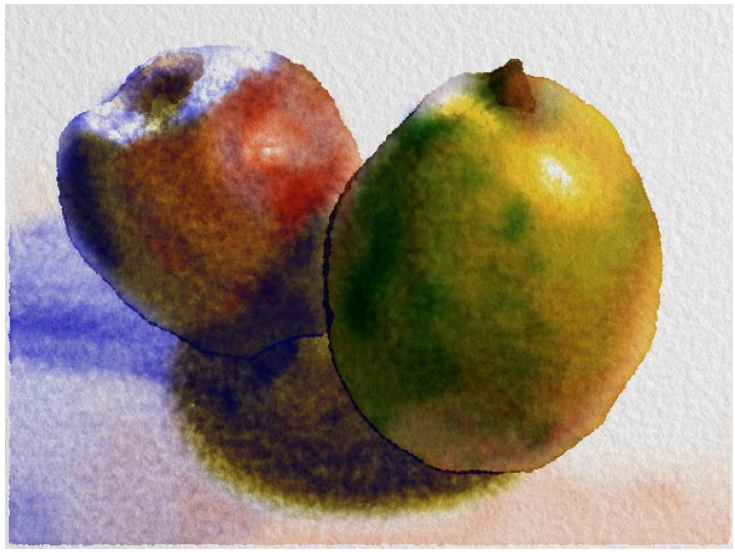


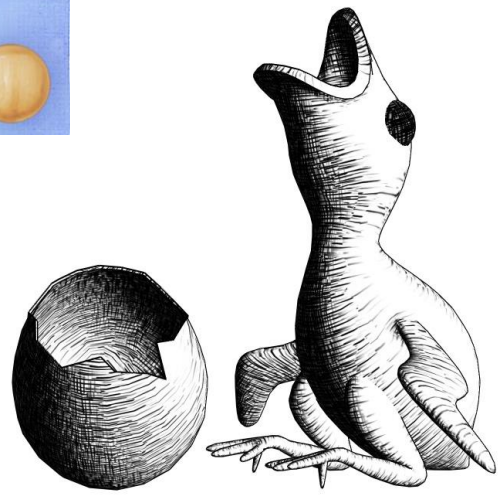


Non-photorealistic rendering (NPR)

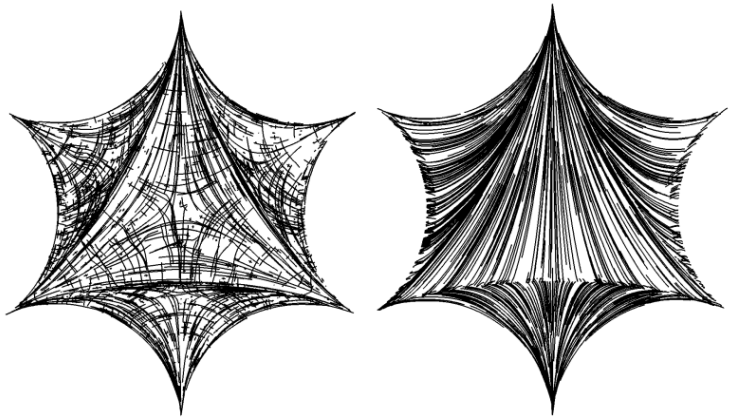


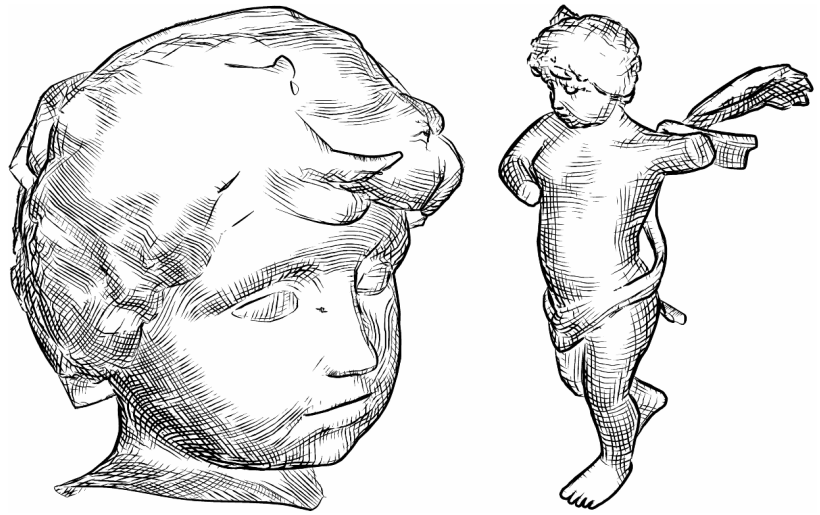


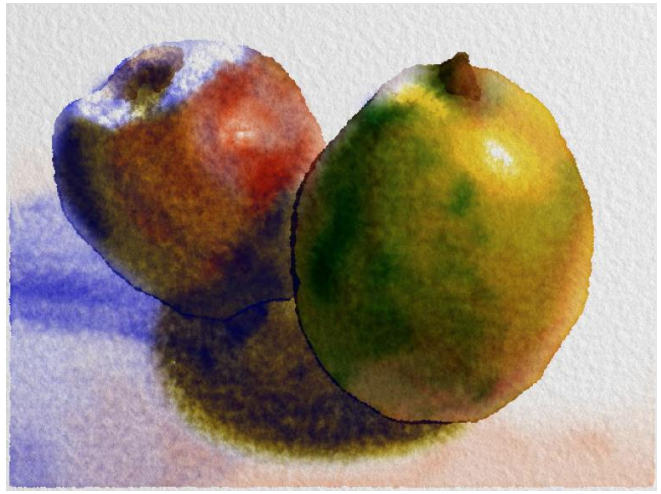




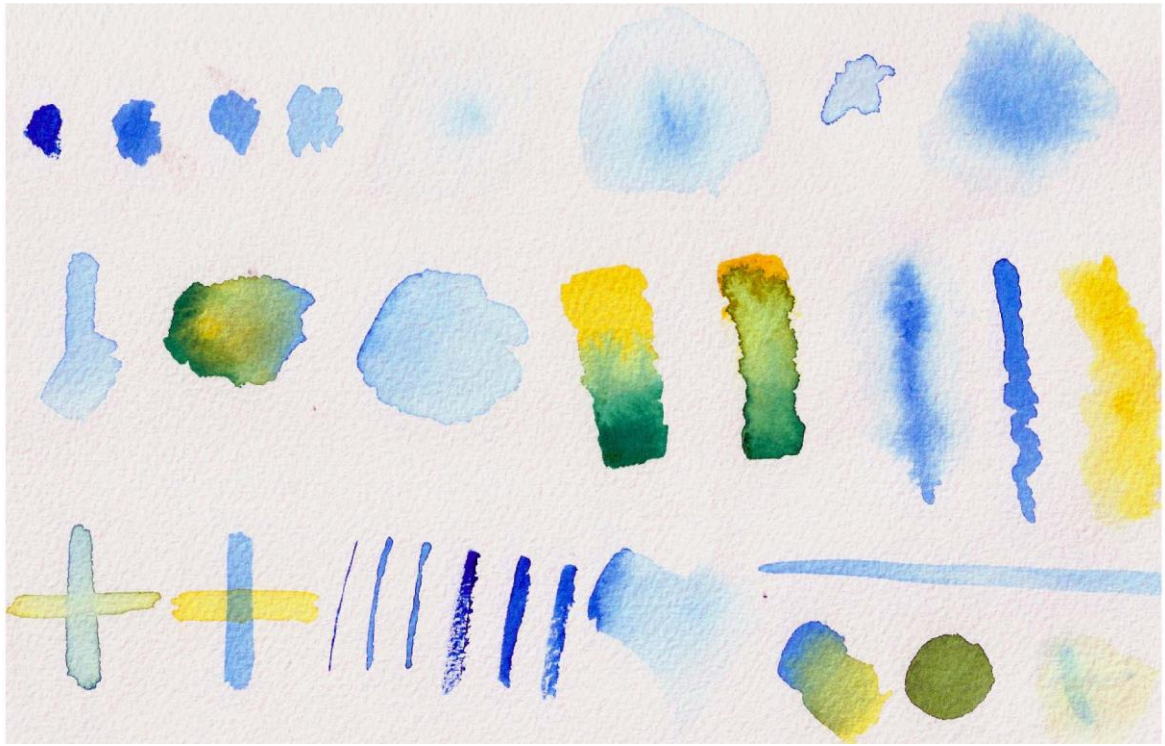
Quest for realism in non-photorealism

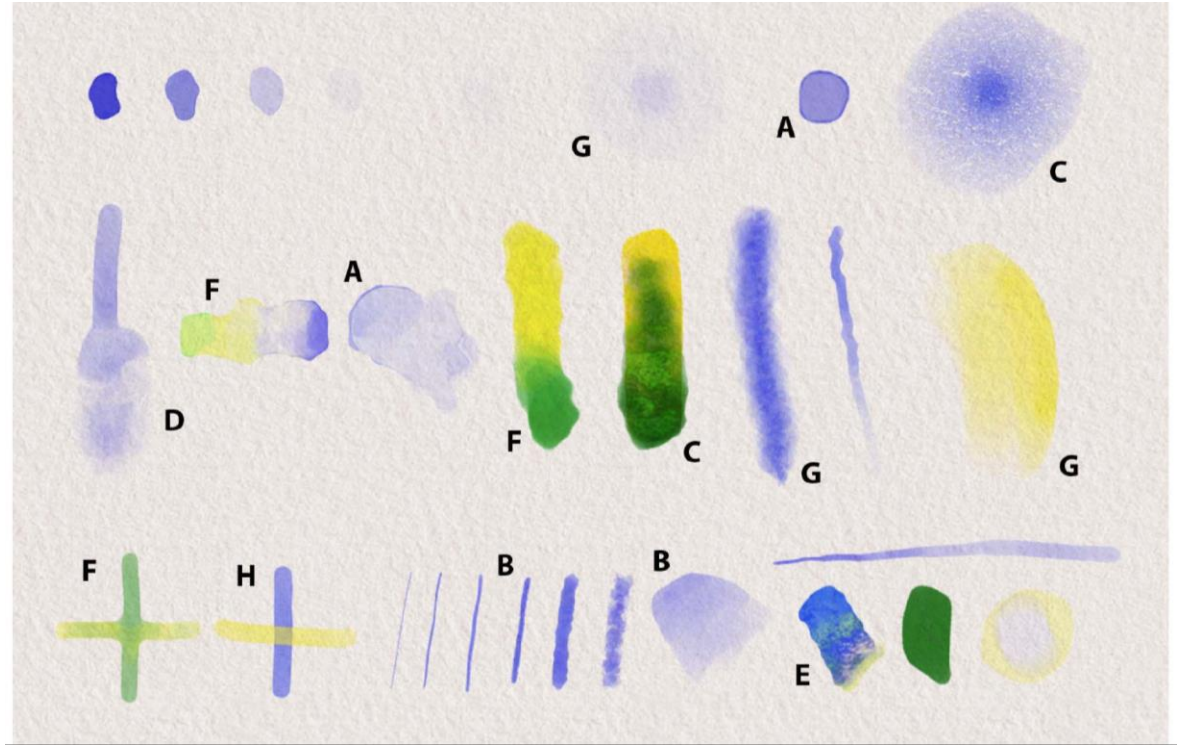






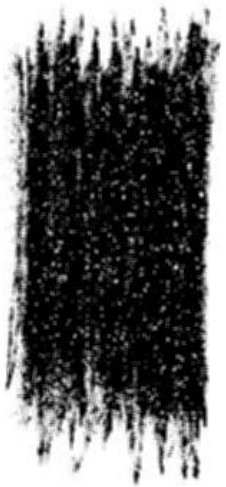
Quest for realism in non-photorealism

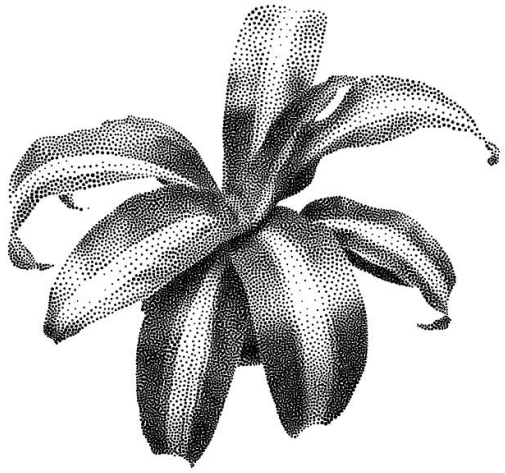
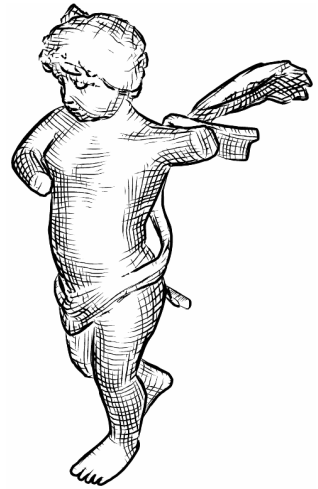


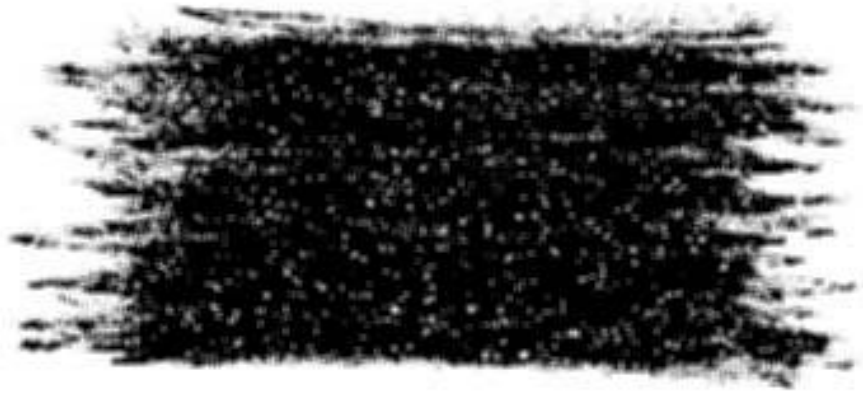


A few examples for NPR techniques

- media simulations
  - typically simulation of physical reality
- human low-level artistic techniques
  - model of the low-level technique
  - dedicated machine learning; recently, also deep learning



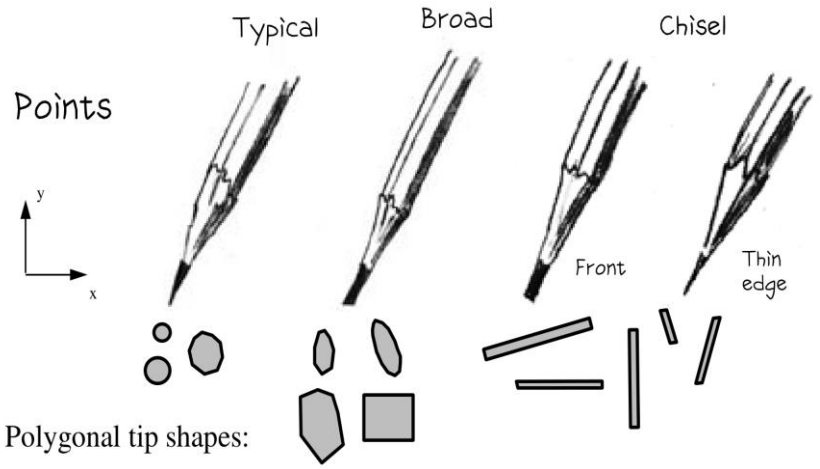




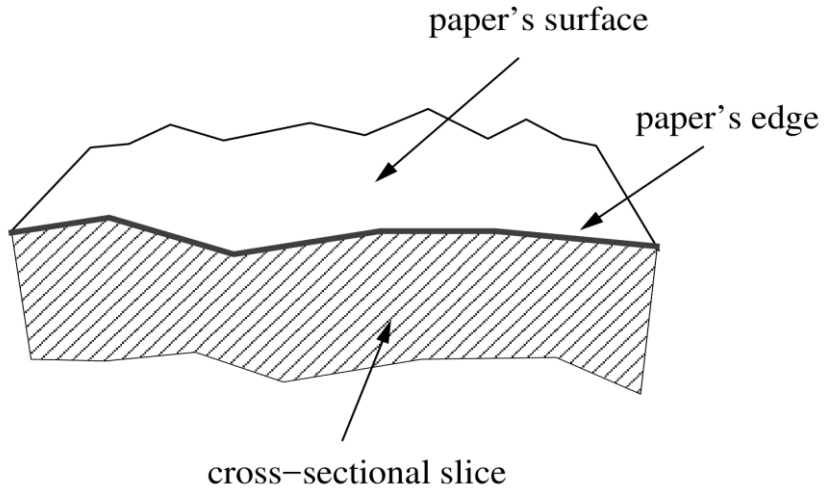


Pencil rendering

Pencil on paper rendering

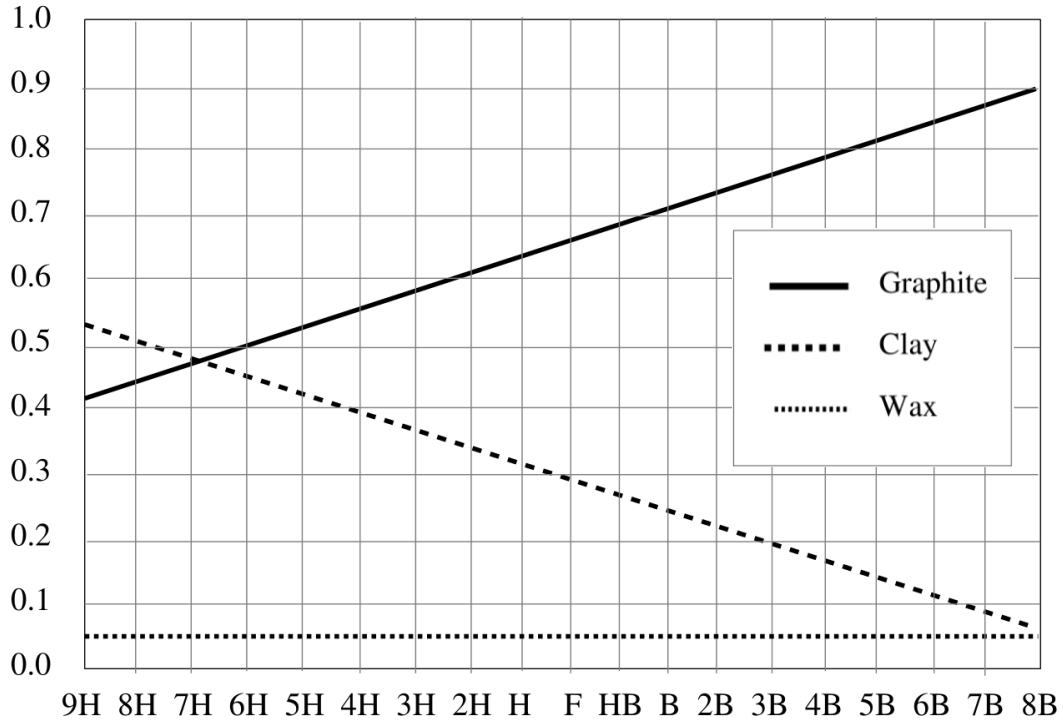
- pencil: graphite pigment mixed with clay and wax
- paper: medium composed of fibers, held together by a glue
- drawing: physical abrasion of the pencil's graphite, which is deposited onto the paper





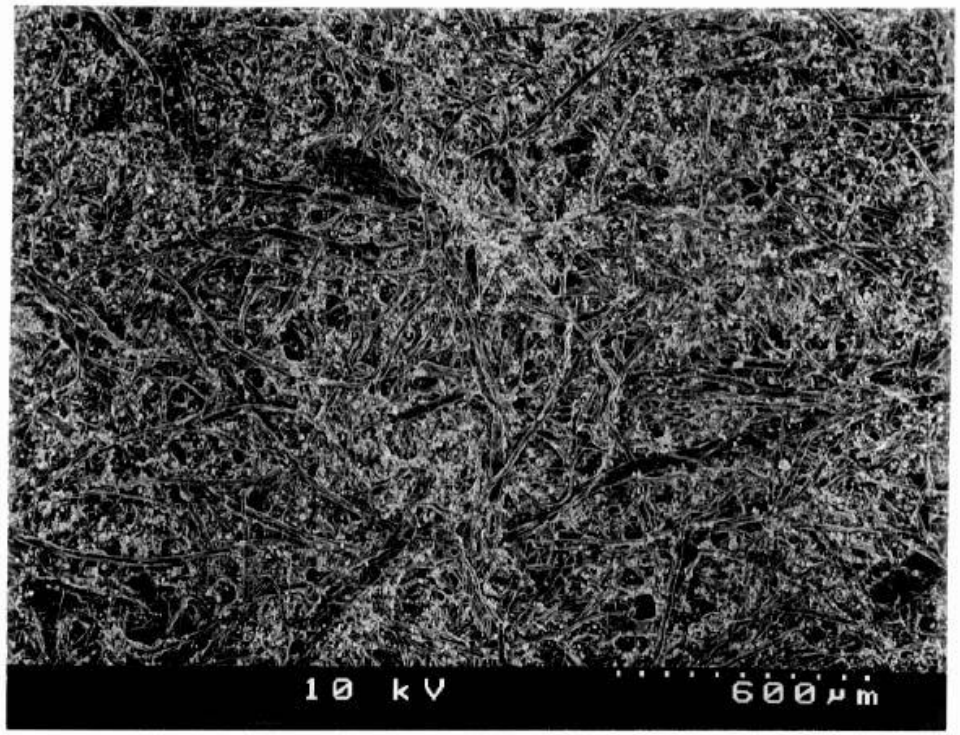
Different types of pencils

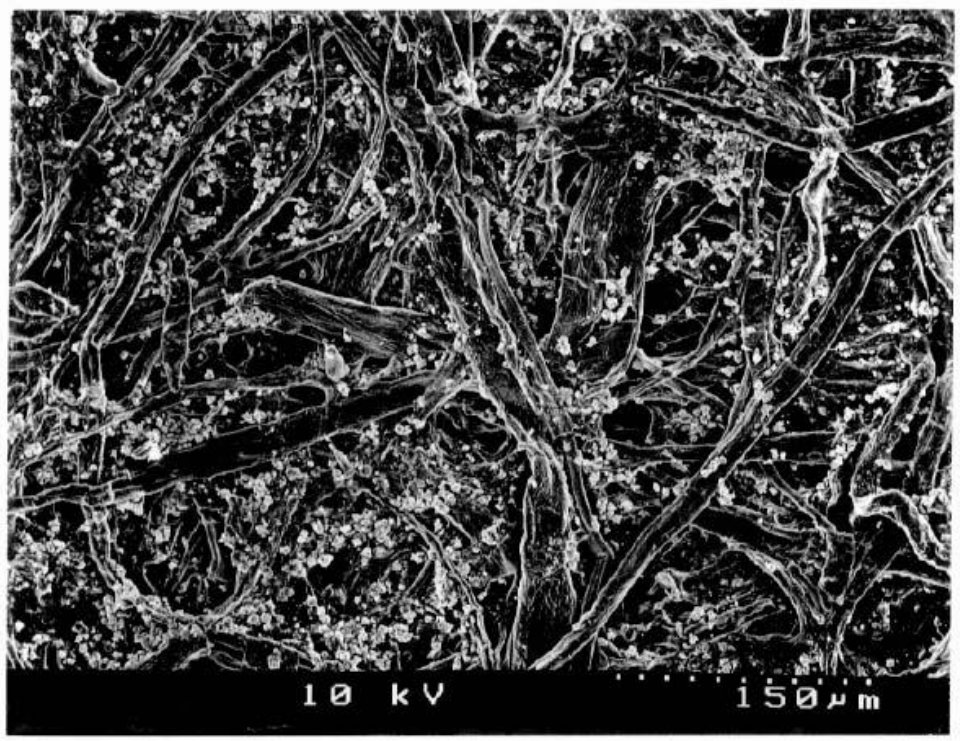
- hardness depends on the mix of graphite, clay, and wax



Paper structure

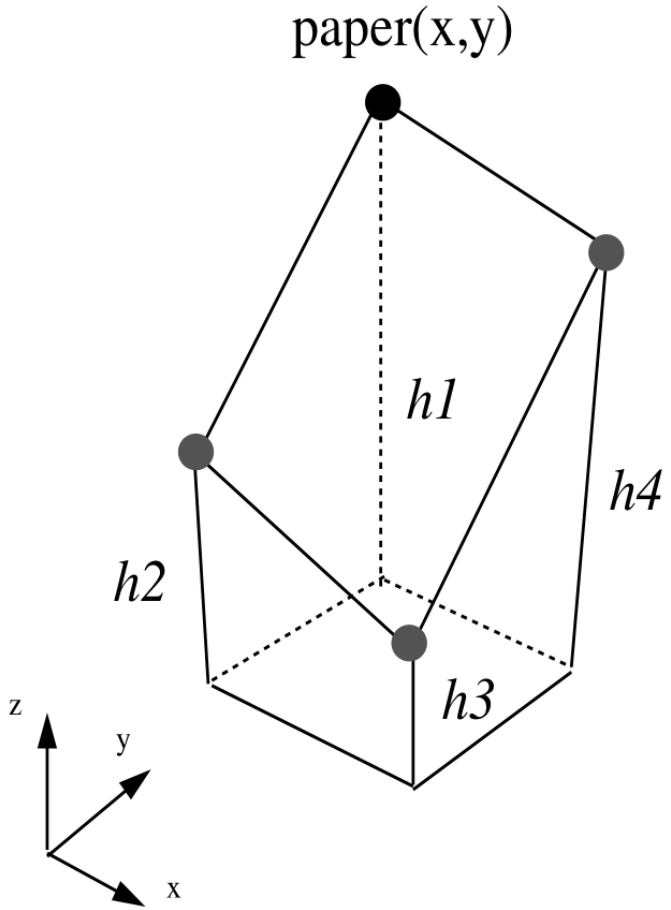
- fibers form peaks and valleys, which “bite” material out of the pencil’s tip

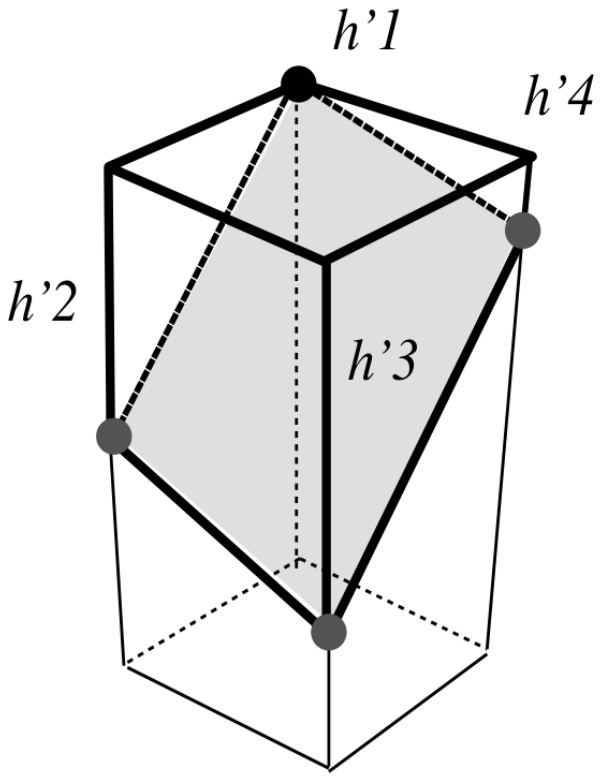




Model of the paper surface

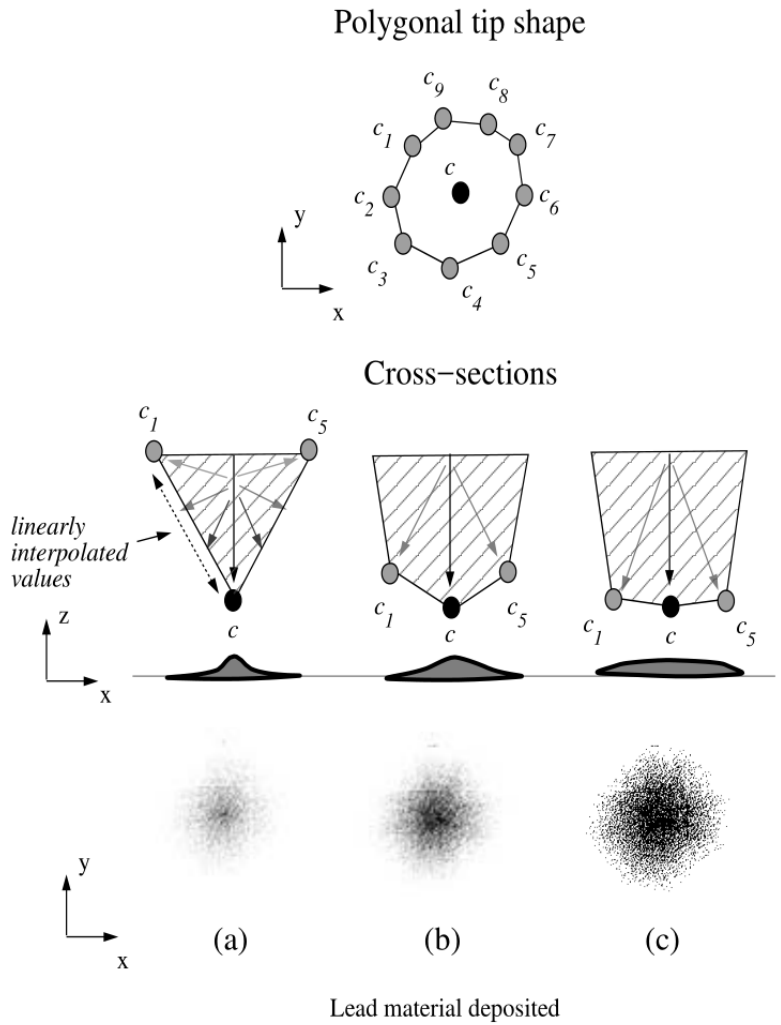
- paper modeled as height map
- volume above the paper can pick up pigment (porous volume)





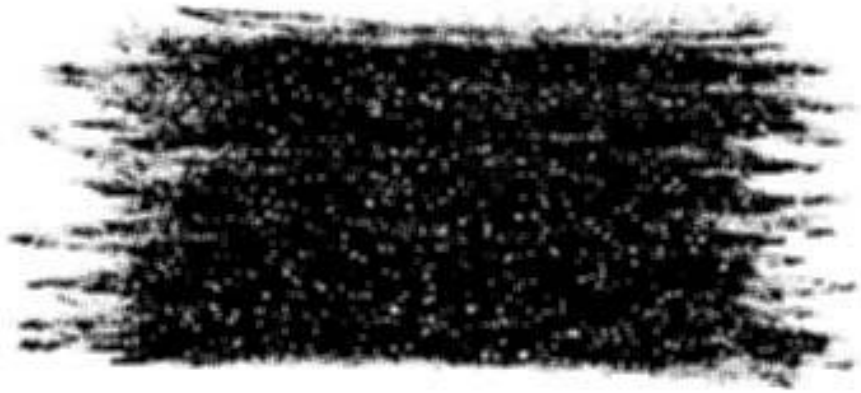
Model of the pencil tip

- model of the tip shape
- model of the tip cross-section
- model of the composition of the tip's core (mix of graphite, wax, and clay)
- also take pressure and pencil angle into account



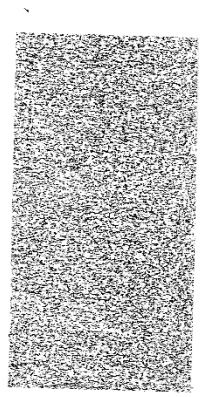
Pencil drawing process

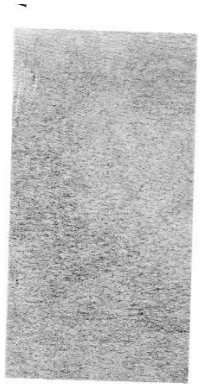
1. Evaluate the polygonal tip shape of the pencil tip
2. Distribute pressure applied to the pencil across the tip shape
3. Compute the paper porous volume
4. Process the grain biting the lead
5. Compute damage caused by the lead to the paper grains



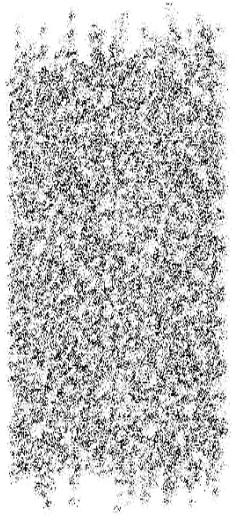
Comparison to real scans

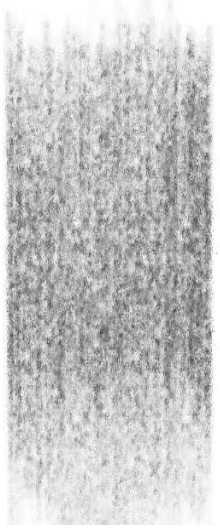
real

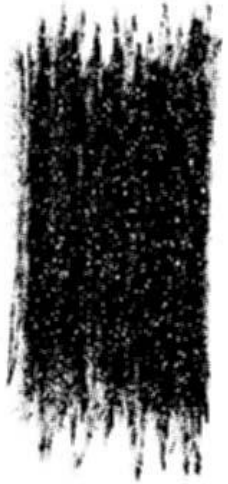




simulated



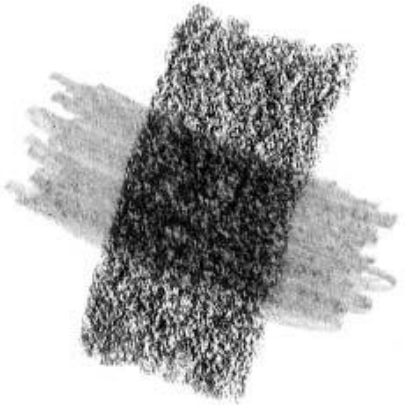


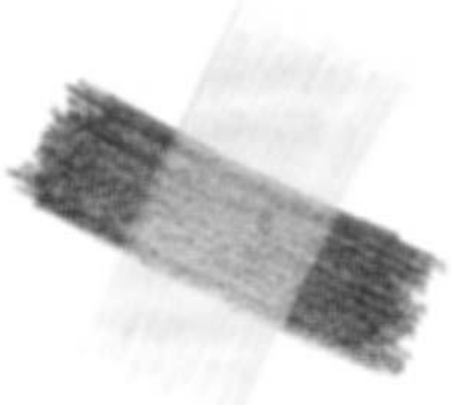


4B, medium pressure | 4B, heavy pressure | 6B, medium pressure | 6H, light pressure

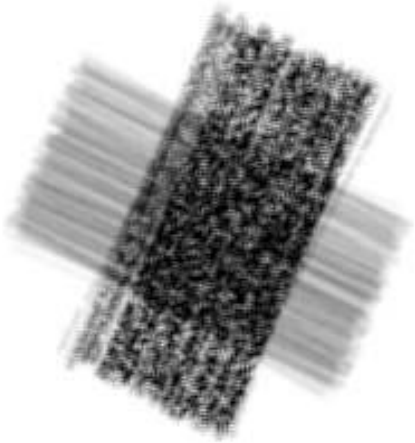
Combinations of several layers

real





simulated



first 4B (dark), then 4H (light)

first 8H (light), then 3B (dark)

Use in automatic (high-level) rendering

Watercolor rendering

Watercolor rendering

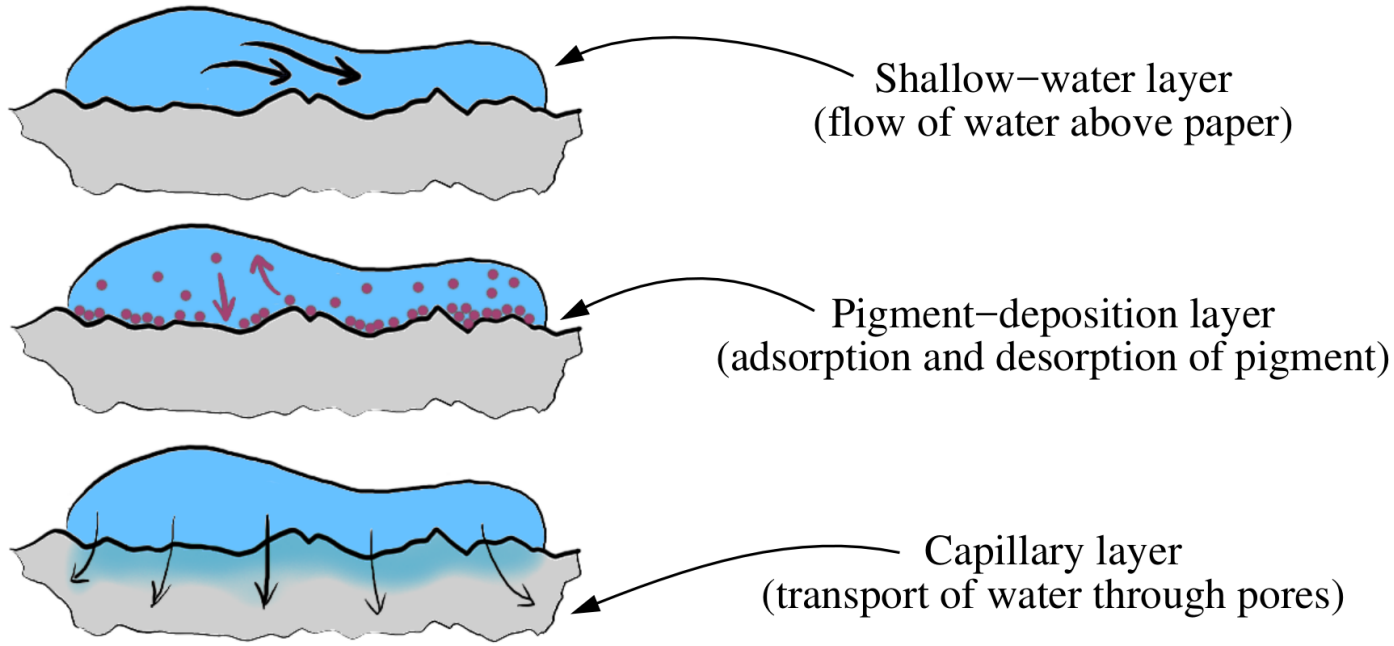
- watercolor: suspension of pigment particles in a solution of water, binder, and surfactant
- paper: made from linen or cotton rags pounded into small fibers, absorbs the water
- sizing layer: controls absorption of water into the paper

Watercolor effects

- dry brush technique, edge darkening, backruns
- granulation, flow effects, glazing

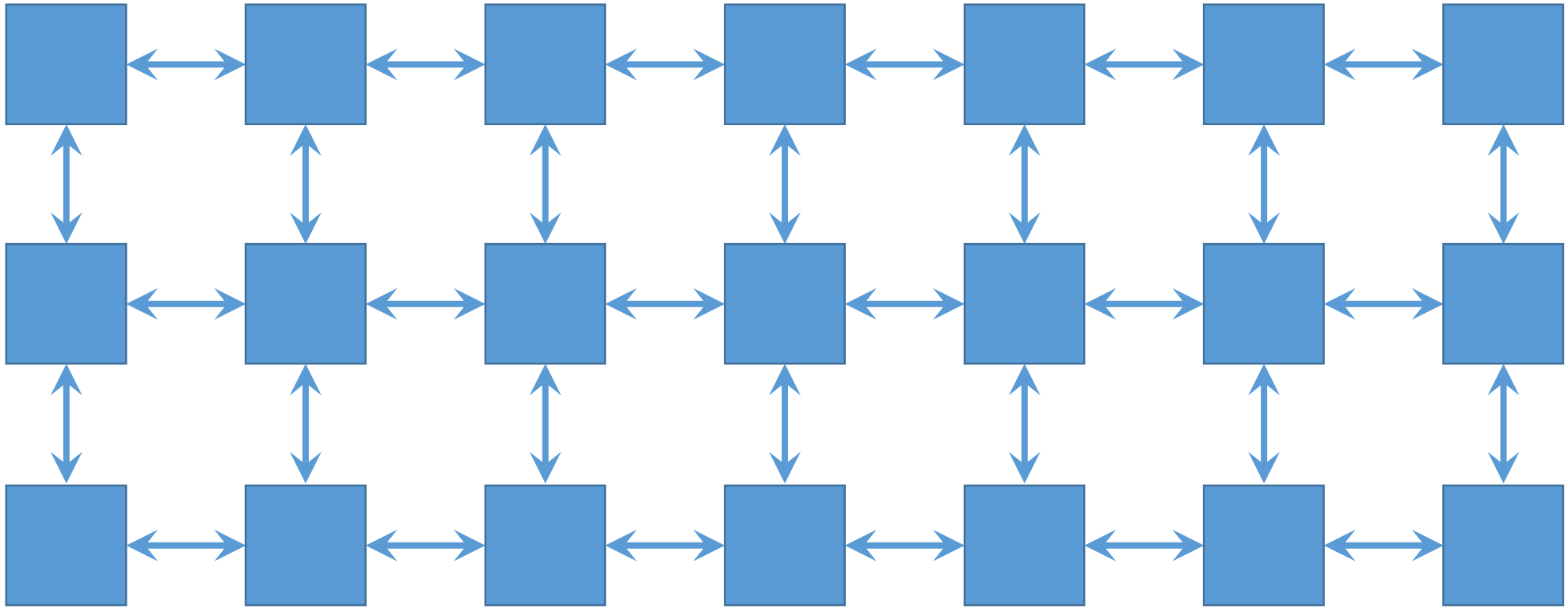
Simulation: A three-layer model

- model to cover the different physical processes during painting
- each layer captures and simulates the necessary physical properties (e.g., flow, pressure, material contents and deposits, etc.)



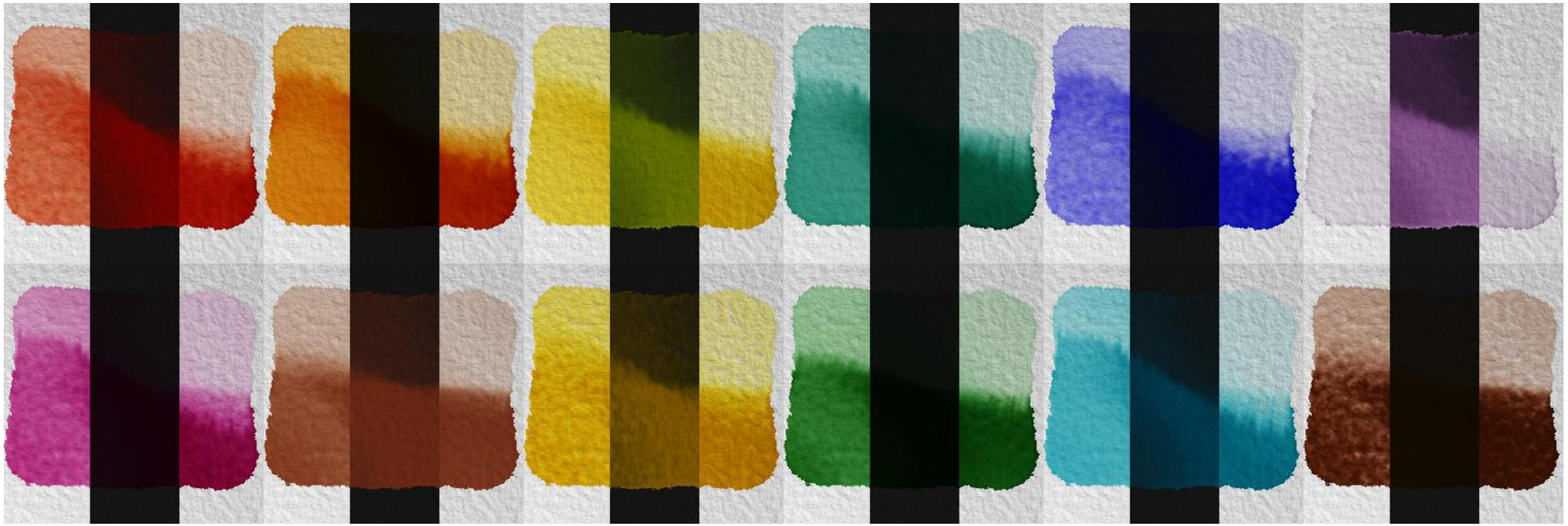
Layers arranged in a grid

- iterative simulation of property & material distribution

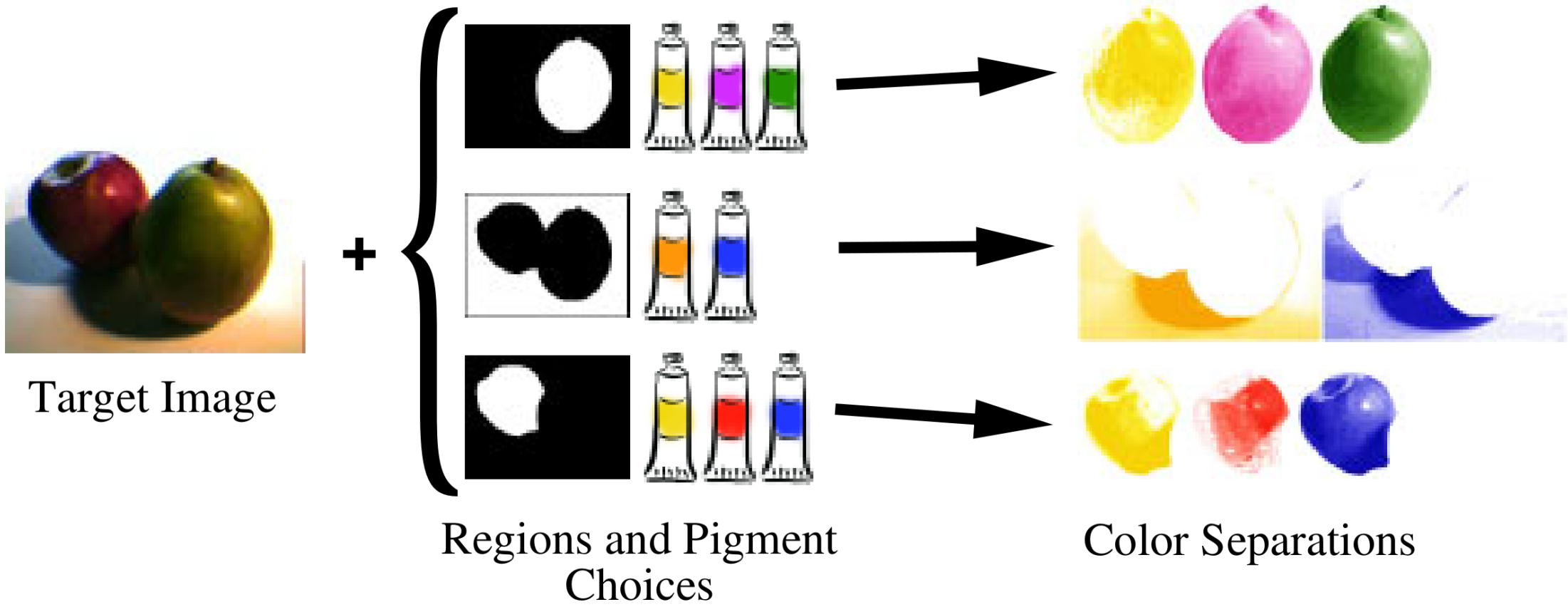


Use of Kubelka-Munk reflection model

- color reflection of color pigments, depending on pigment properties



(Semi-)Automatic painting

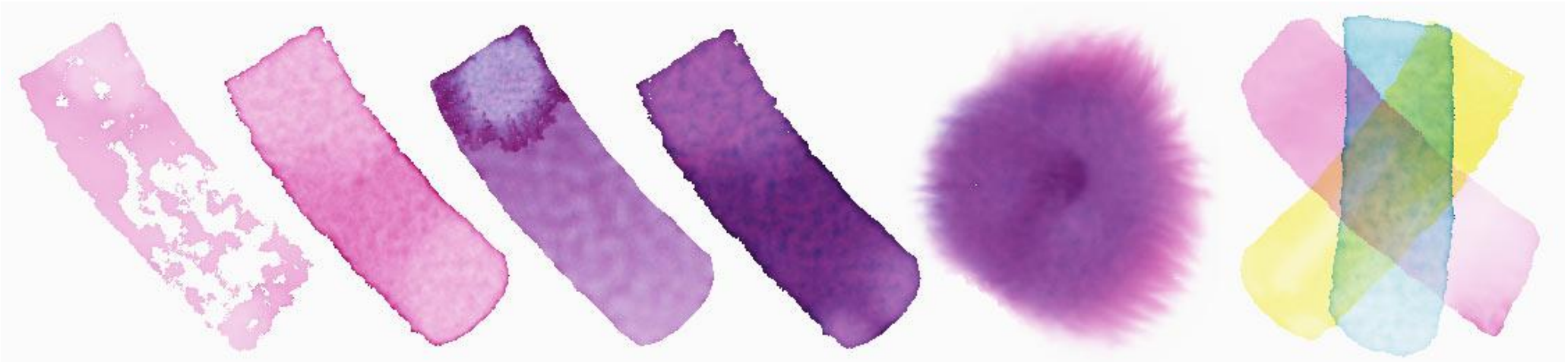


Result

Comparison of simulation with reality

real:

simulated:



Newer approaches: Control of abstraction

comparison to previous result

watercolor rendering with image abstraction

direct watercolor simulation

Newer approaches: Temporal coherence

Newer approaches: Art direction of effects

Newer approaches: Art direction of effects

Expressive 2018

MNPR: A Framework for Real-Time Expressive Non-Photorealistic Rendering of 3D Computer Graphics

Santiago E. Montesdeoca
Nanyang Technological University
Interdisciplinary Graduate School, MAGIC

Hock Soon Seah
Nanyang Technological University
School of Computer Science and Engineering

Amir Semmo
Hasso Plattner Institute for Digital Engineering
University of Potsdam

Pierre Bénard
LaBRI (UMR 5800, CNRS, Univ. Bordeaux)
Inria

Davide Benvenuti
Nanyang Technological University
School of Art, Design and Media

Romain Vergne
Univ. Grenoble Alpes
CNRS, Inria, Grenoble INP, LJK

Joëlle Thollot
Univ. Grenoble Alpes
CNRS, Inria, Grenoble INP, LJK

Oil painting simulation

Oil painting simulation

- paint: pigment suspended in a drying oil
- typically painted on canvas
- paint usually viscous, i.e. little or no interaction between different paint strokes

- we do not need physical simulation, but low-level artistic technique of stroke placement (stroke-based rendering)

Stroke-based rendering & layers

- stroke samples color at start
- trace strokes along a chosen direction, e.g.:
  - gradient of the image luminance
  - edge extraction + direction field interpolation
- quit after maximum length or color difference too large

- layers of increasing detail: brush size divided by 2 each time

Stroke-based rendering: control

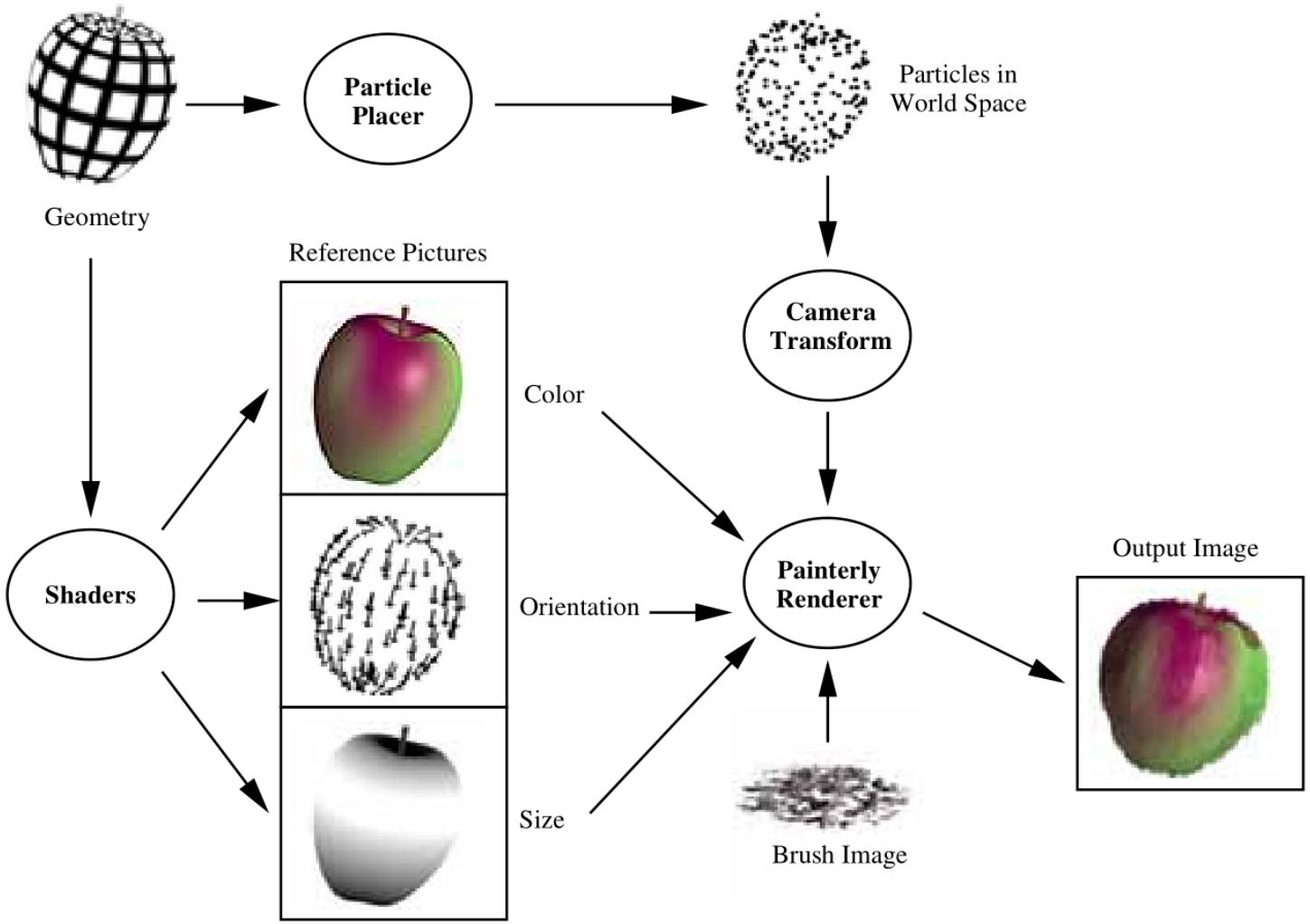
source image

interactively painted weight image w_{app}

result for one set of weights

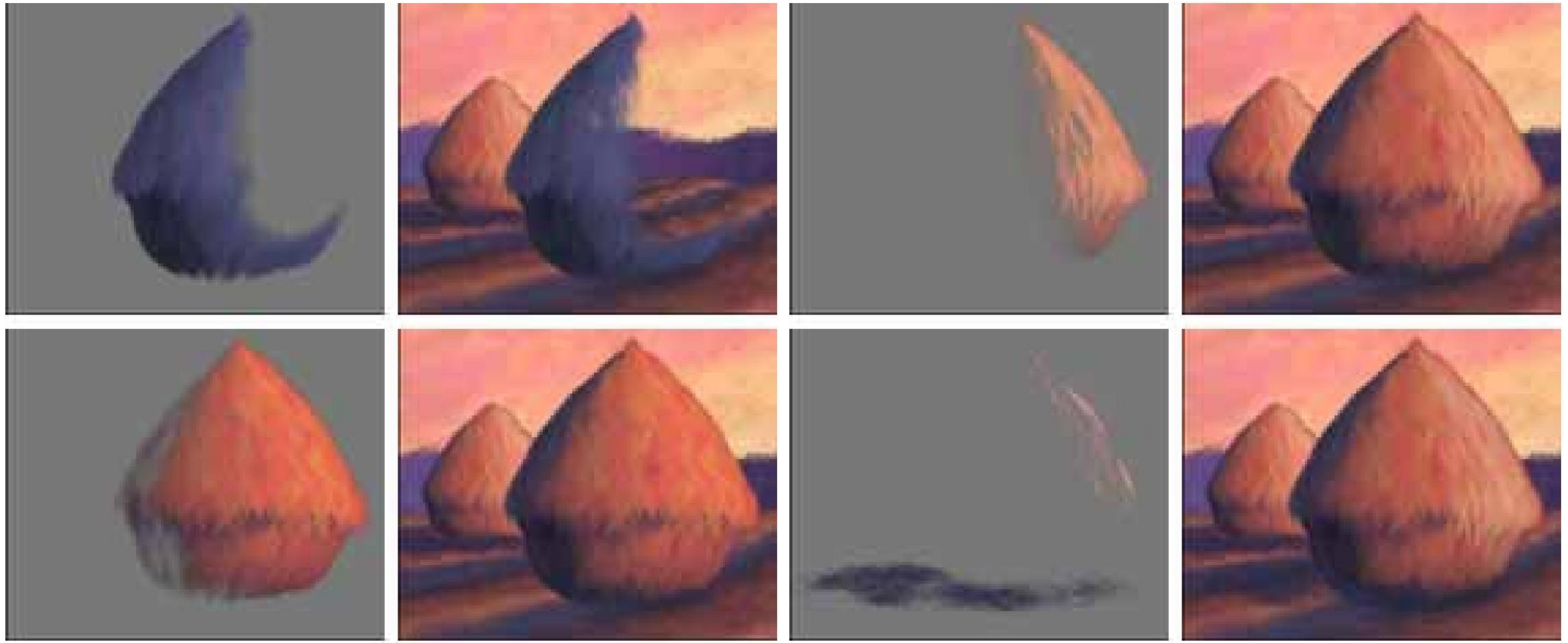
result for another set of weights

Particle-based model for 3D rendering



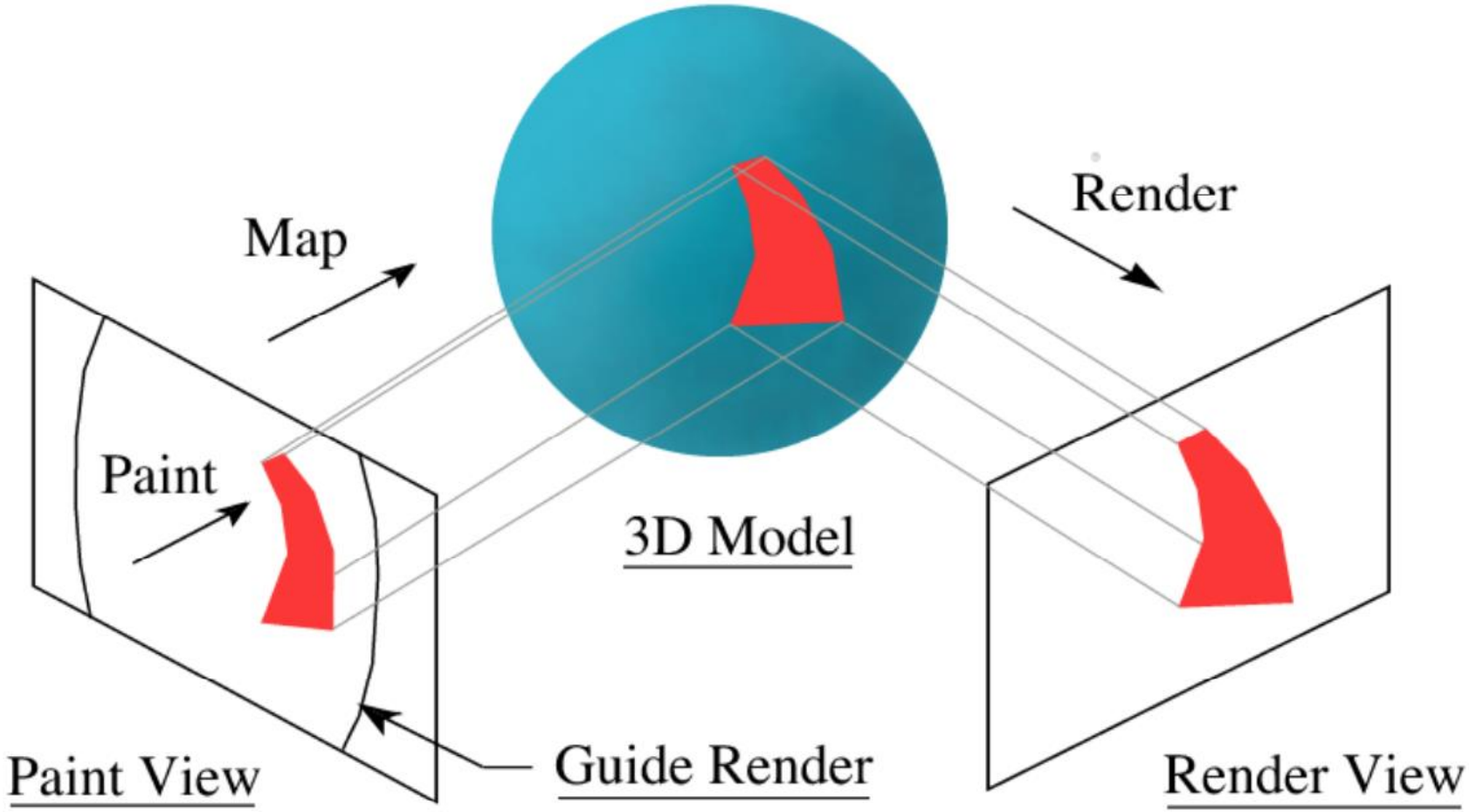
Layers to composit different effects

- cold colors, main (warm) illumination, highlights, shadows



Animation possible

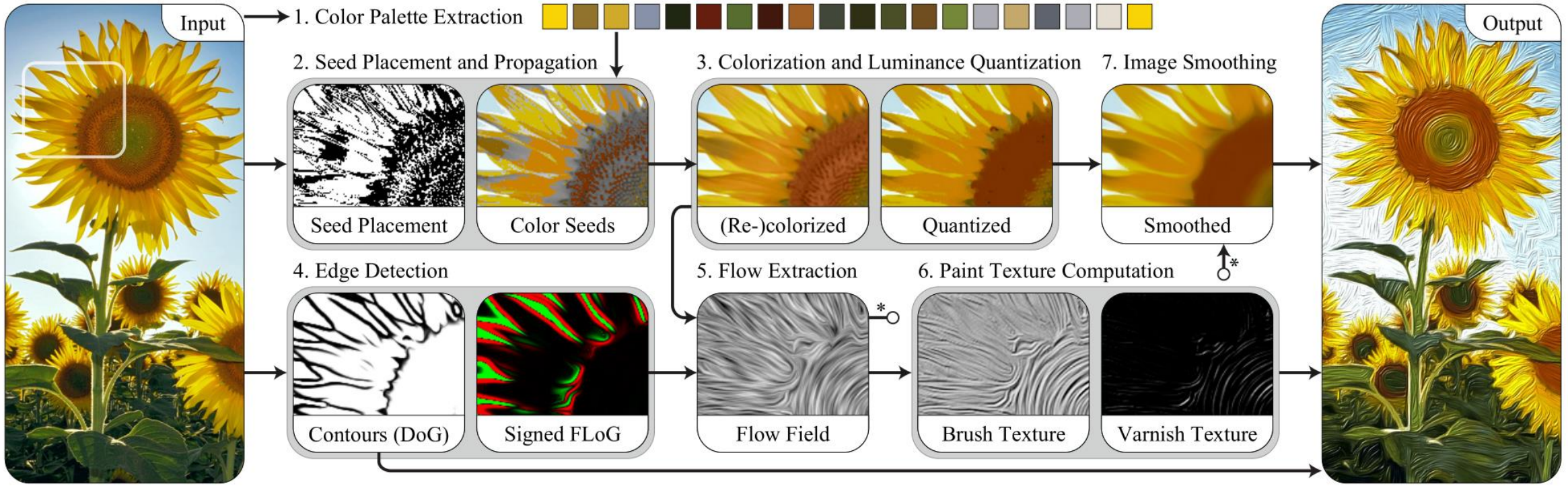
Disney's Deep Canvas



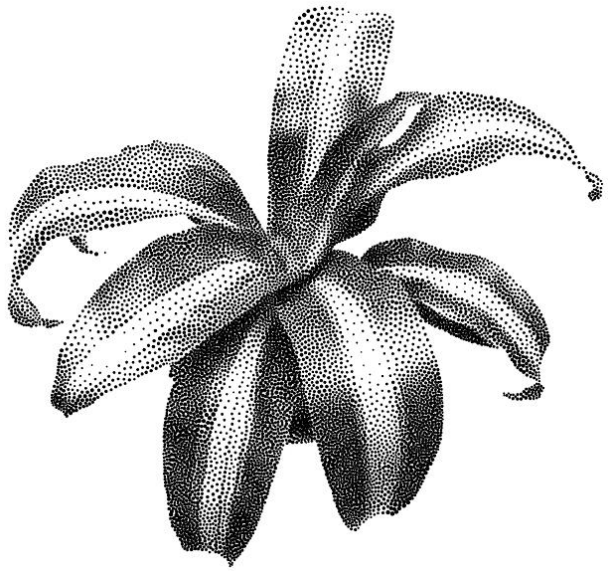
Disney’s Deep Canvas

Automatic stroke placement (etc.)

- more comprehensive and complex approaches exist
- e.g., fully automatic painterly rendering:



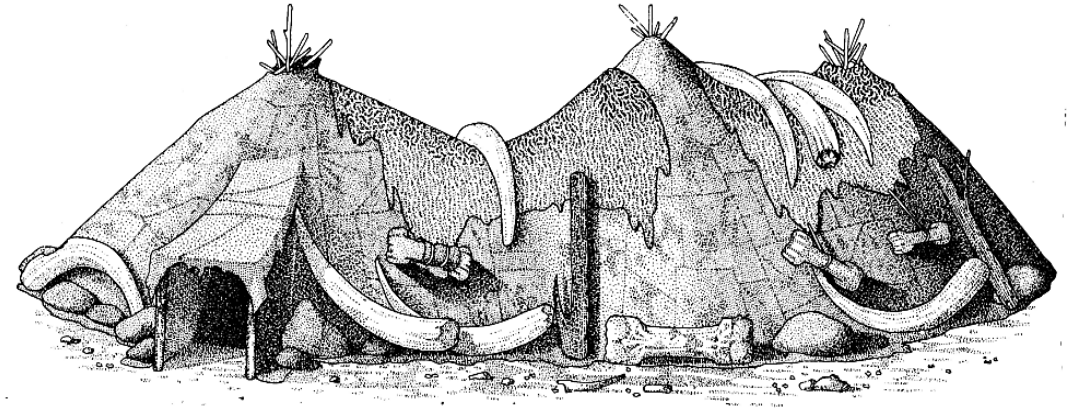
Example results

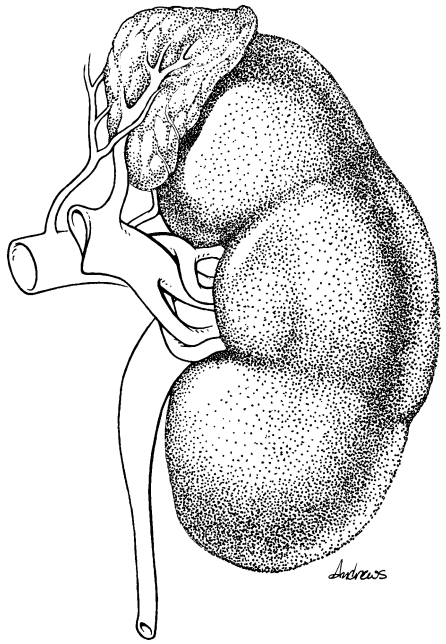


Stippling

Stippling: Rendering with points/dots

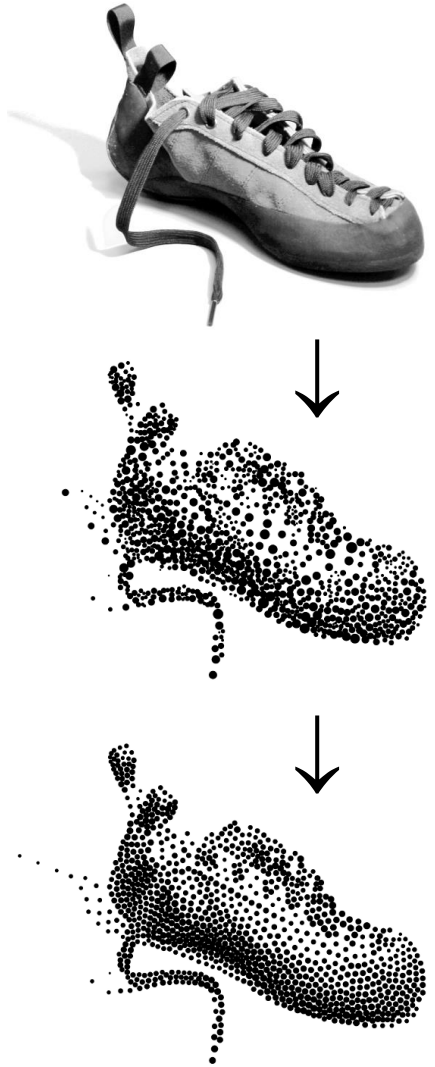
- suggest form in natural graduation of tone
- do not suggest directionality of a surface structure
- typically arranged so that they do not exhibit artifacts
- can also suggest form if placed along (curved) lines
- each stipple has a purpose





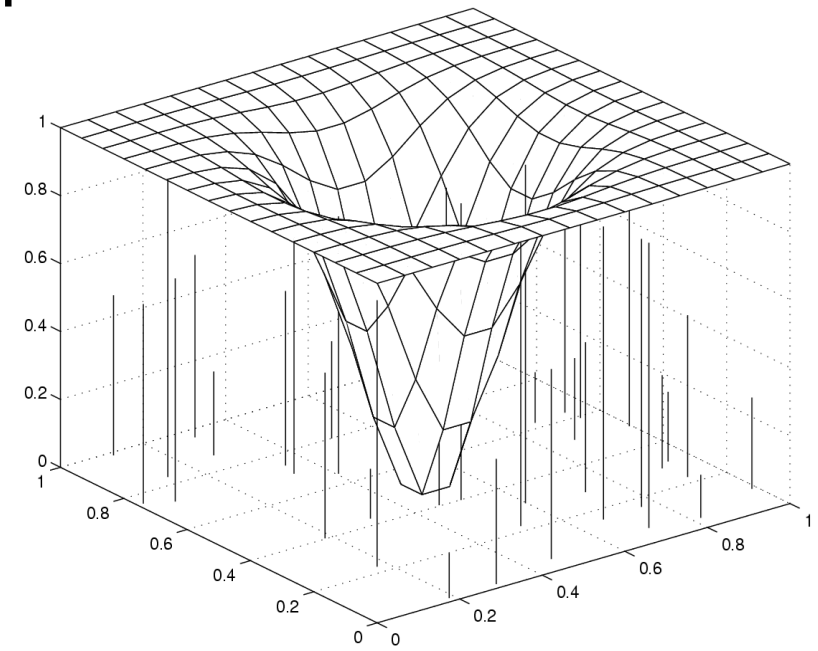
Rendering with points: General approach

- 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



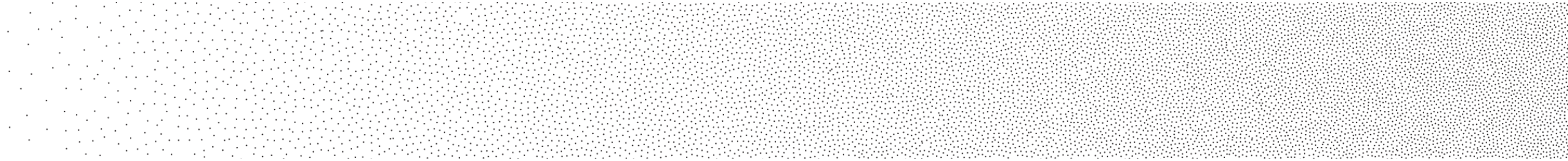
Initial point distribution: Height map

- 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 (high intensities) will have few points that pass
  - dark regions (low intensities) will have more points that pass



Goals for NPR stippling

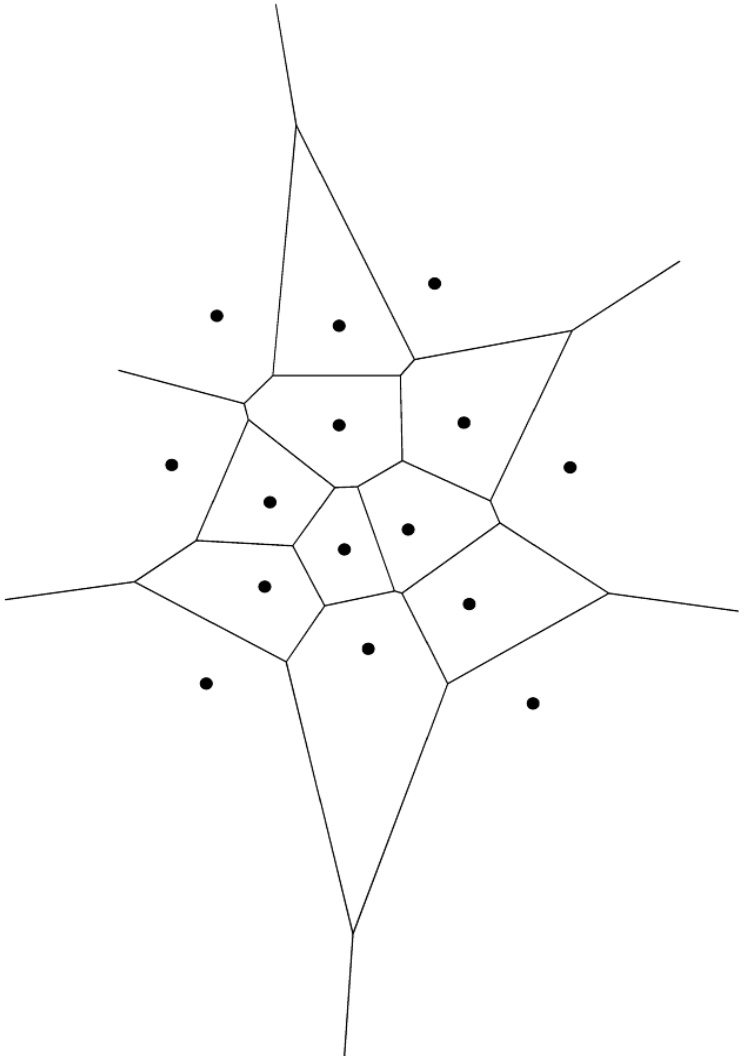
- no overlapping stipples, no visual patterns if possible
- even distributions: i.e., stipples should have about the same distance to their neighbors
- linear intensity response: the perceived gray values generated for a linear gray ramp should also behave linearly



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

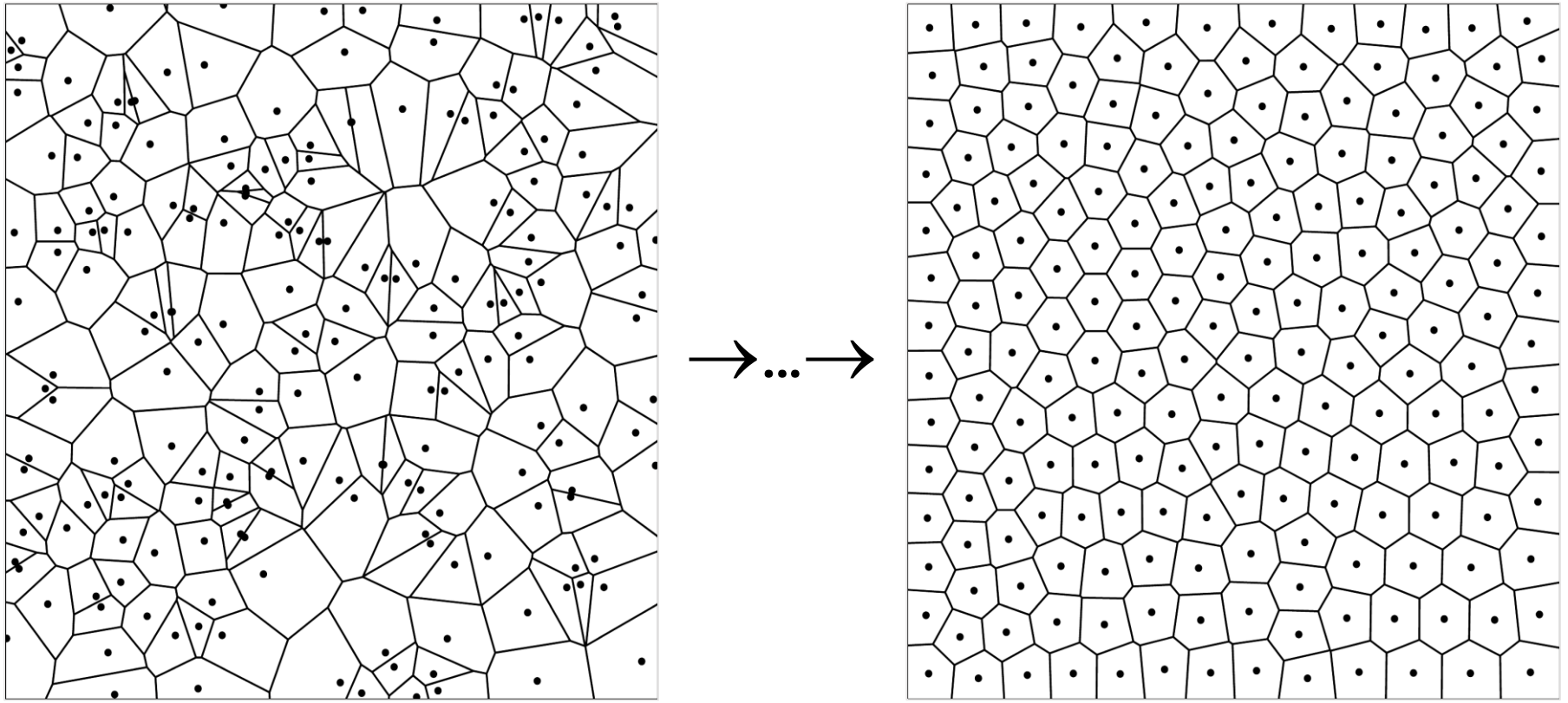
Relaxation to obtain even distributions

- based on Voronoi diagrams
- idea: iteratively 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”



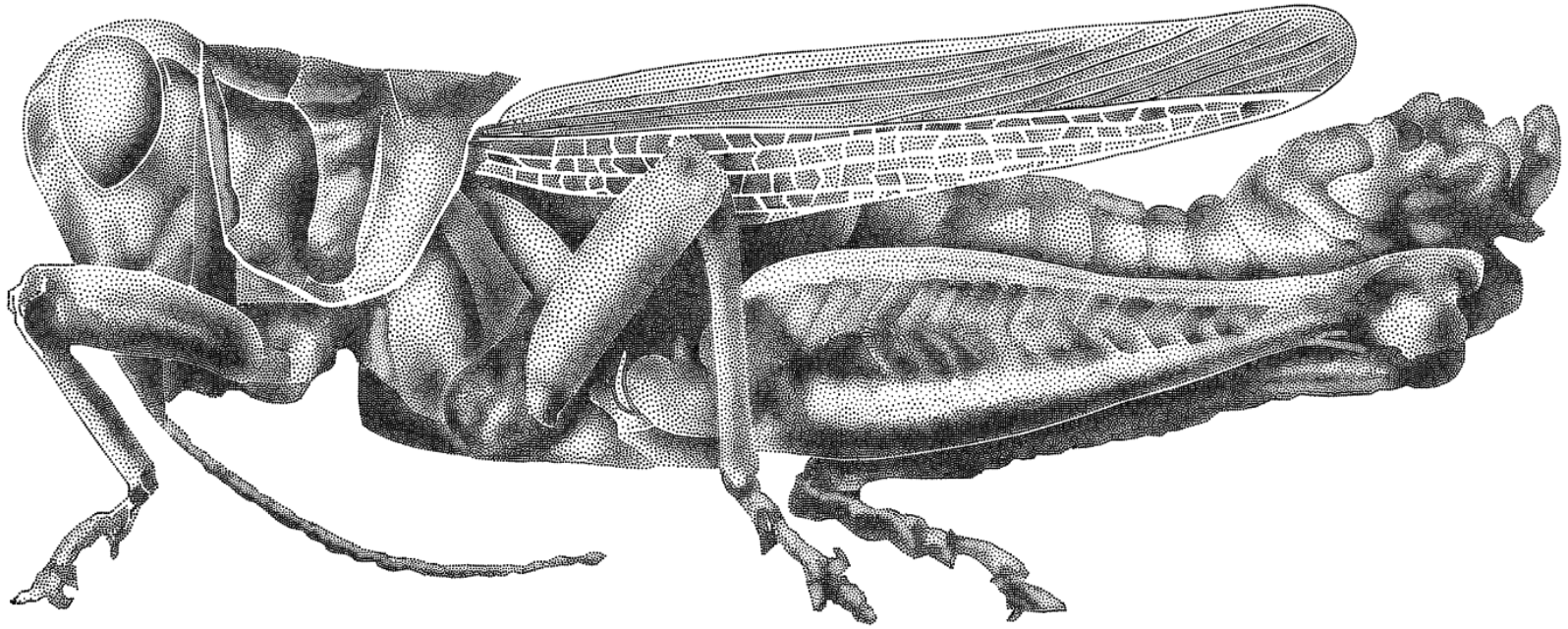
Iteration in Lloyd's method

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



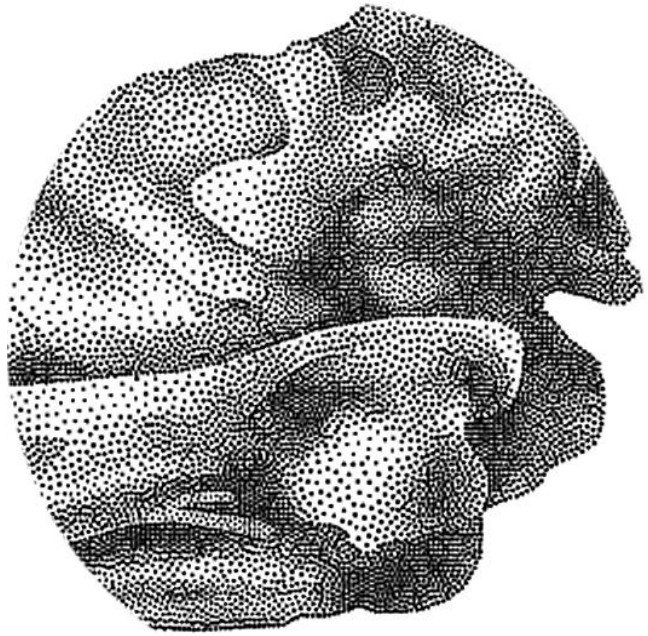
Interactive relaxation-based stippling

- 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)



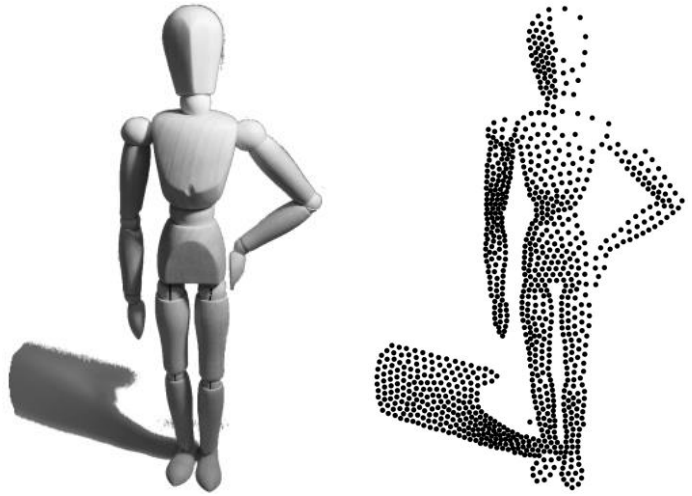
Interactive Relaxation-Based Stippling

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

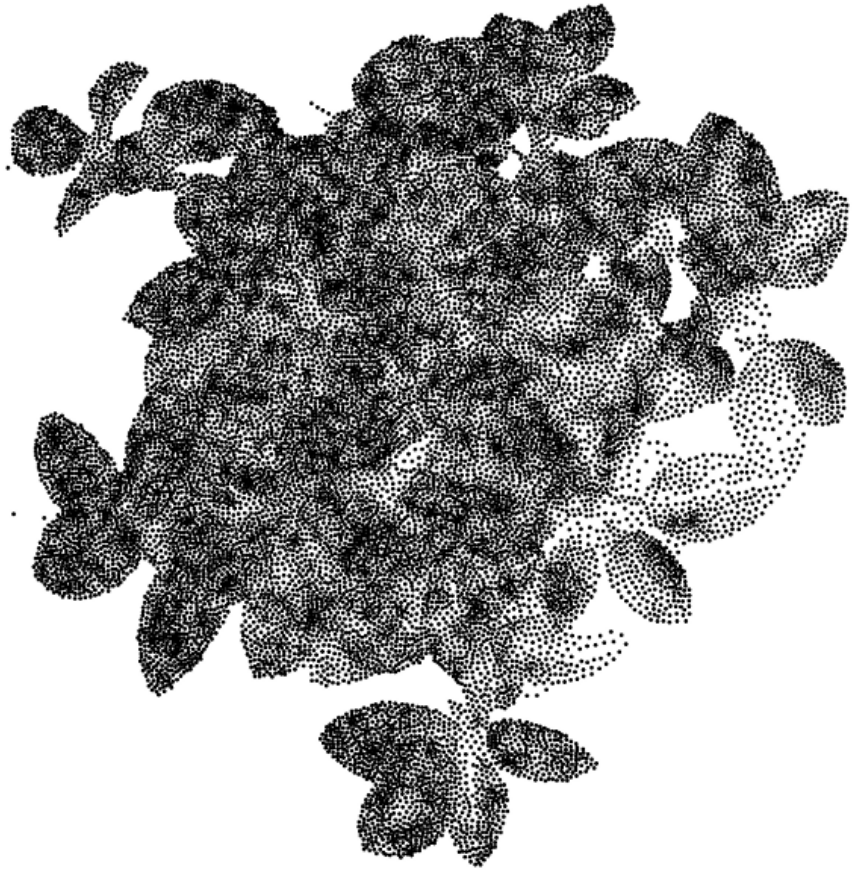


Automatic relaxation-based stippling

- 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



Automatic relaxation-based stippling

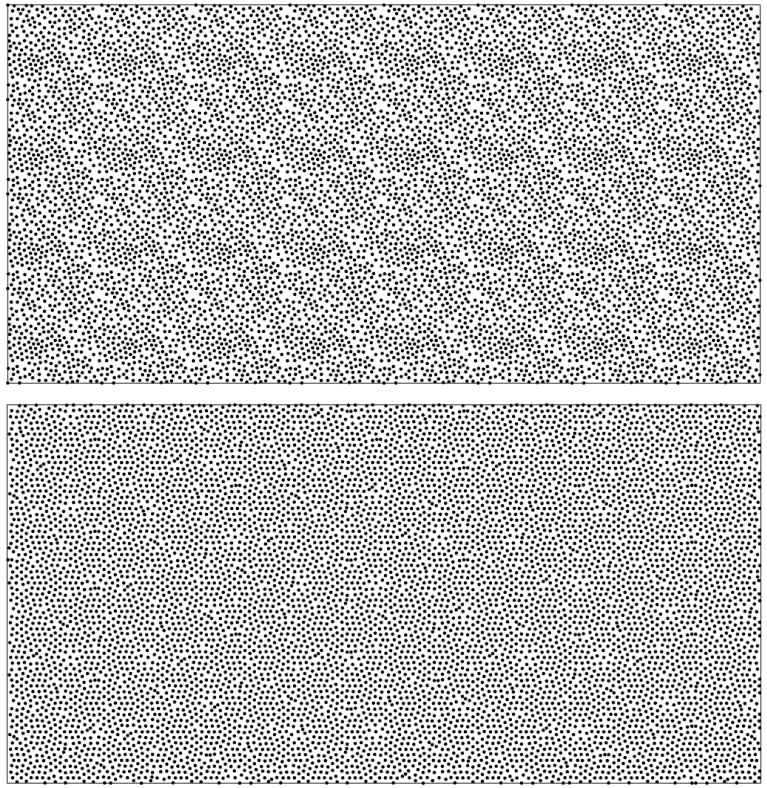


constant point size

Automatic relaxation-based stippling

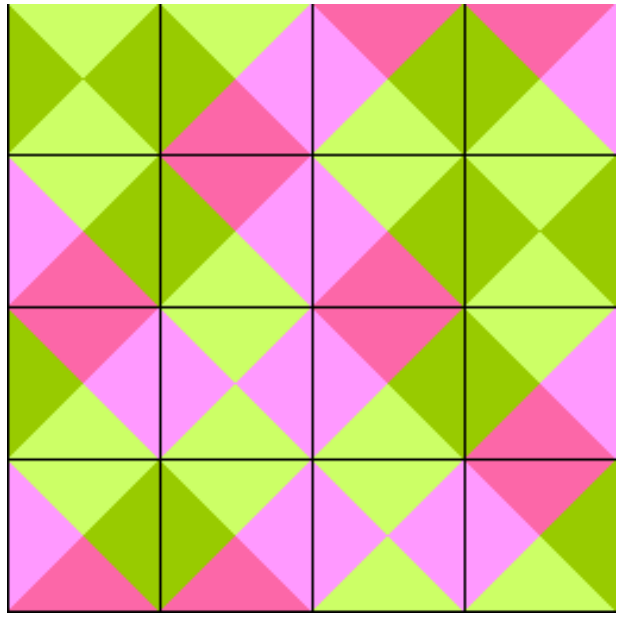
Stippling with recursive Wang tiles

- 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

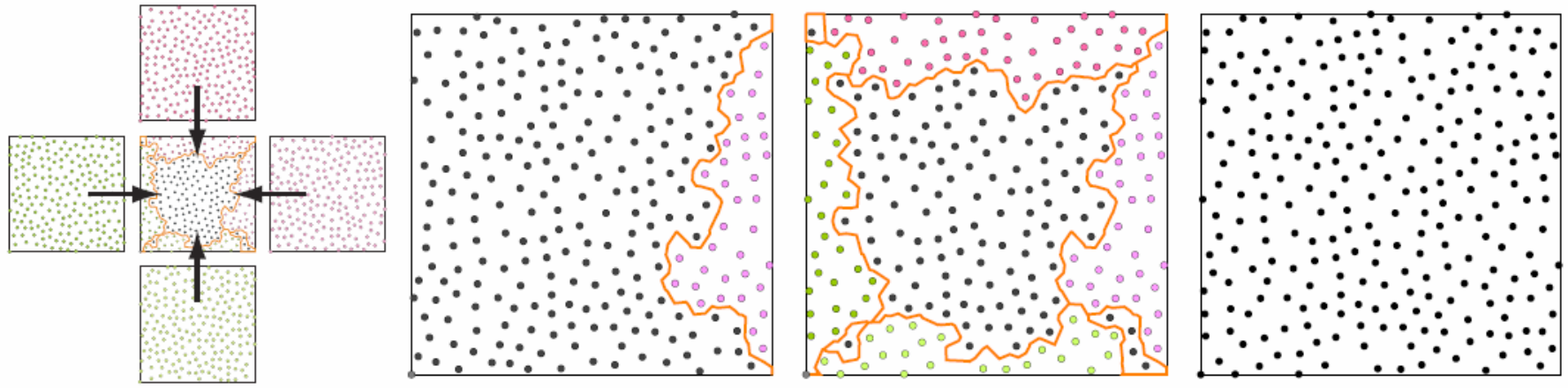
- idea: use Wang tiling
  - tiling of the plane using a set of tiles with color-coded edges
  - each edge can only be matched with another tile of the same color
  - depending on # of colors used yields large non-periodic tiling



- associate each tile with matching point distribution
  - need to ensure that each individual tile has “good” distribution
  - need to make sure that tiles match according to colors

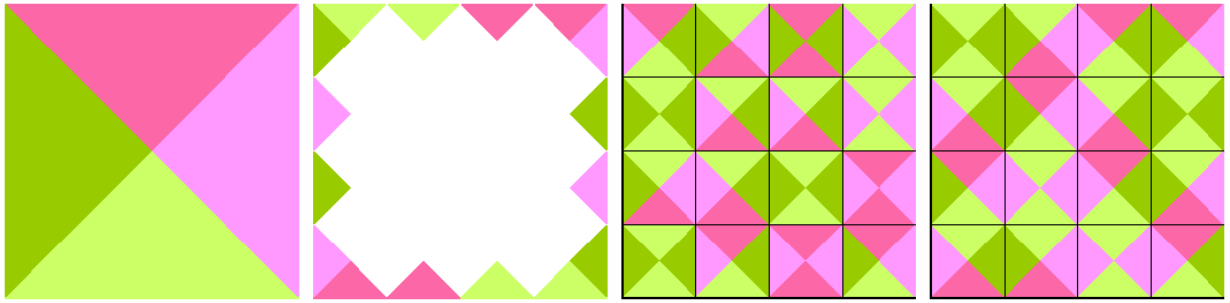
Stippling with recursive Wang tiles

- 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

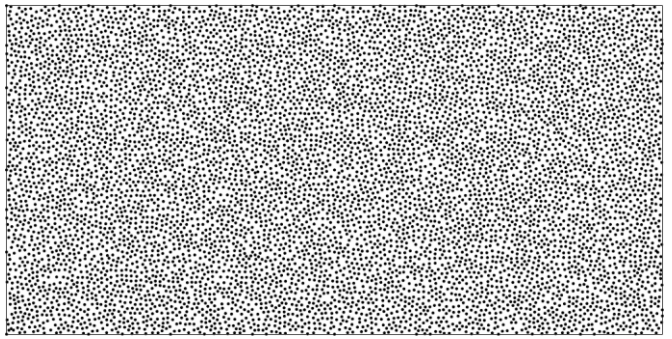
- recursion to be able to “zoom into” a distribution
  - need order in the stipple distribution of each tile
  - need a subdivision scheme for the tiling itself
- approach
  - each color is assigned a unique sequence of new colors
  - stochastic search of possible tiling with border constraints



  - relaxation to match base and detail point distribution

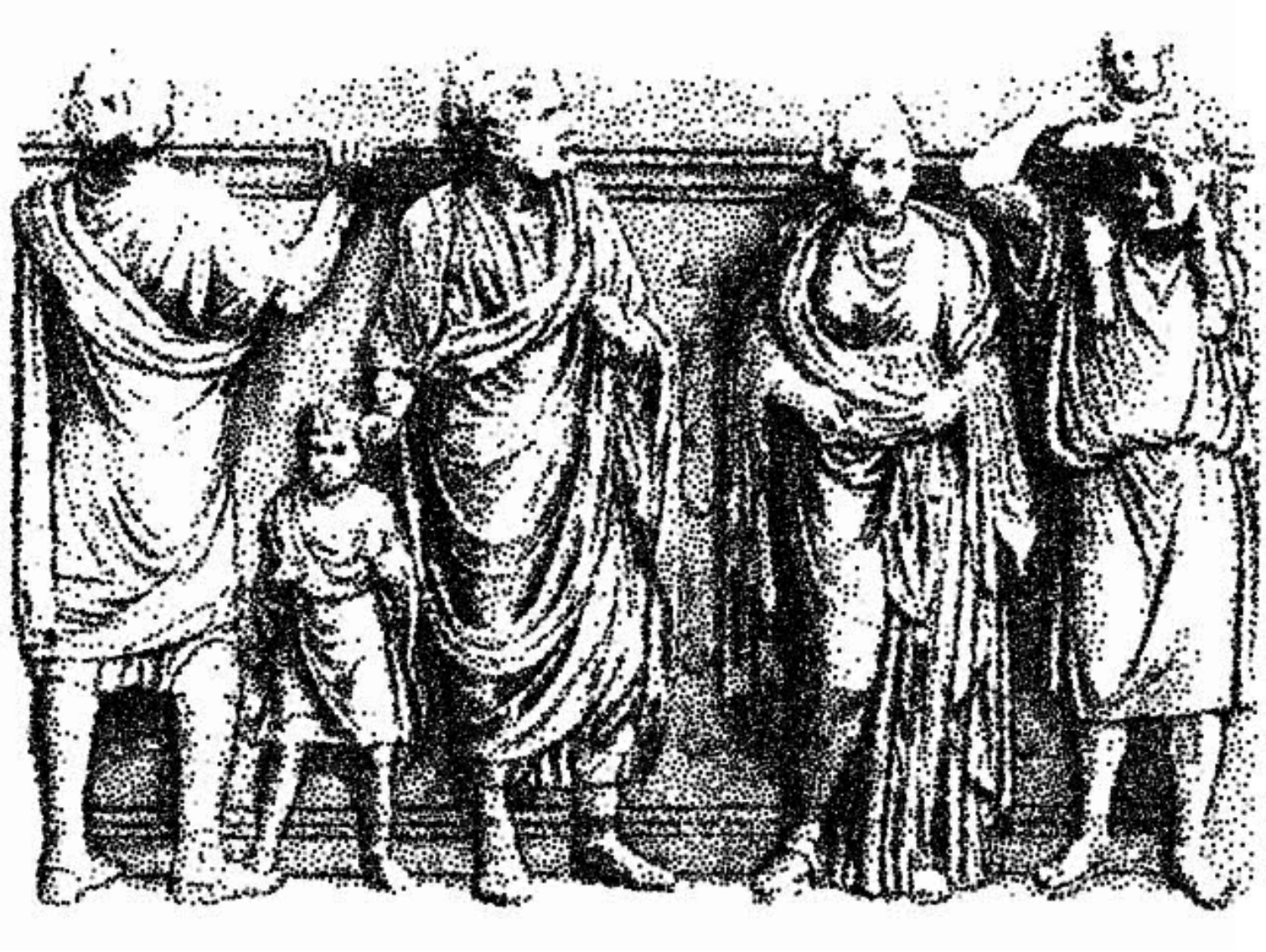
Stippling with recursive Wang tiles

- result: recursive, non-periodic tiling that can be applied to interactive stippling



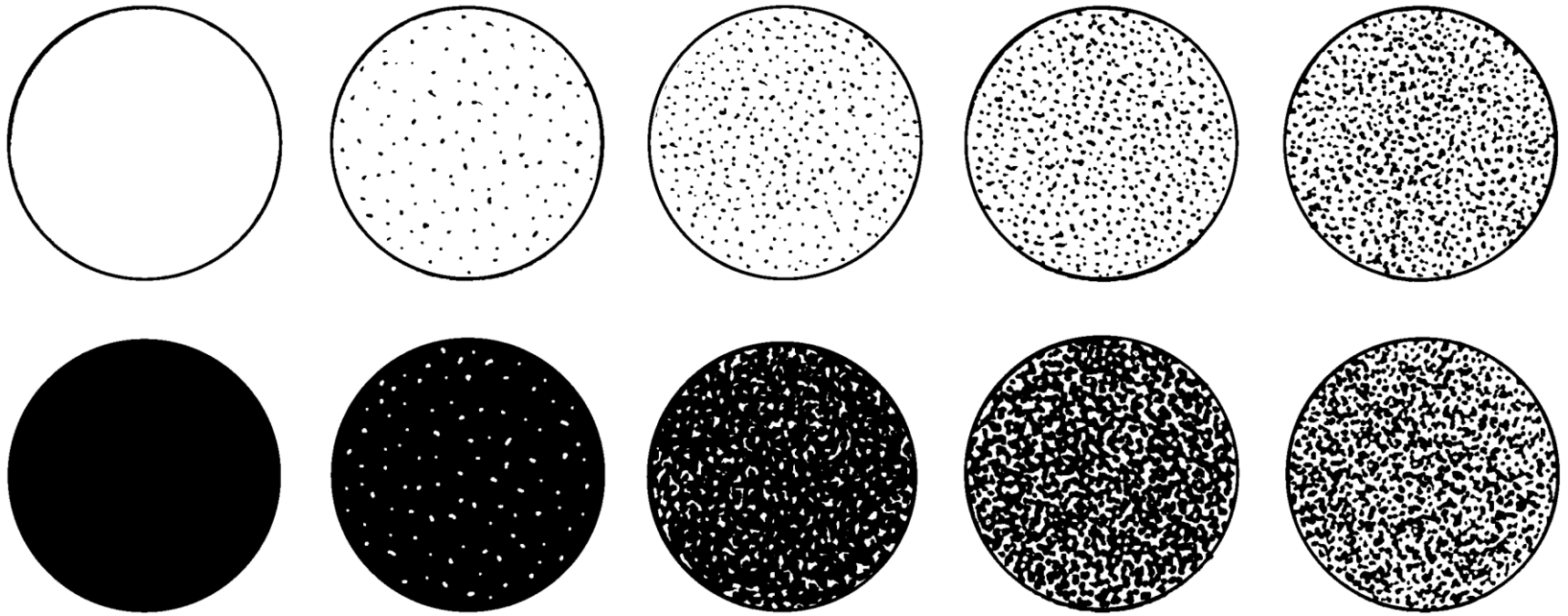
- ranking of each stipple can be used to determine which stipple is rendered first to approximate a grayscale image

Stippling with recursive Wang tiles

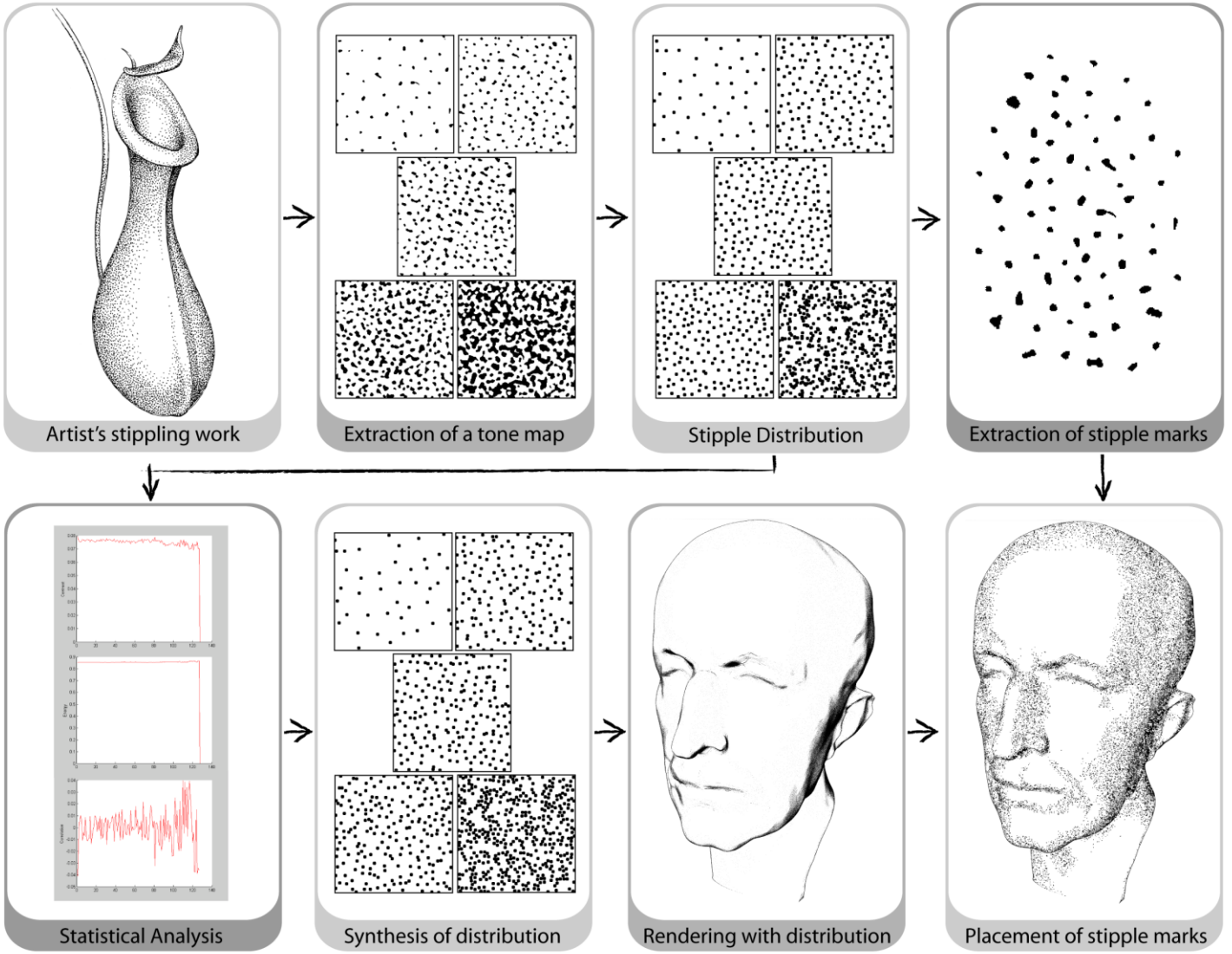


Example-based stippling

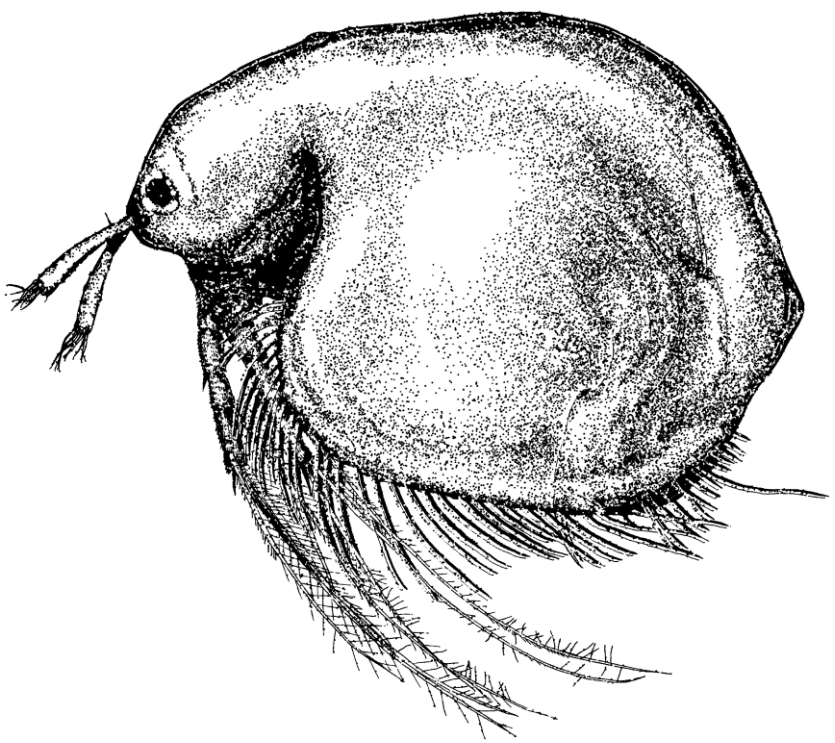
- remaining problem: results too sterile
- idea: base stippling on human-drawn exemplars
- 2 issues to address: placement & stipple dots

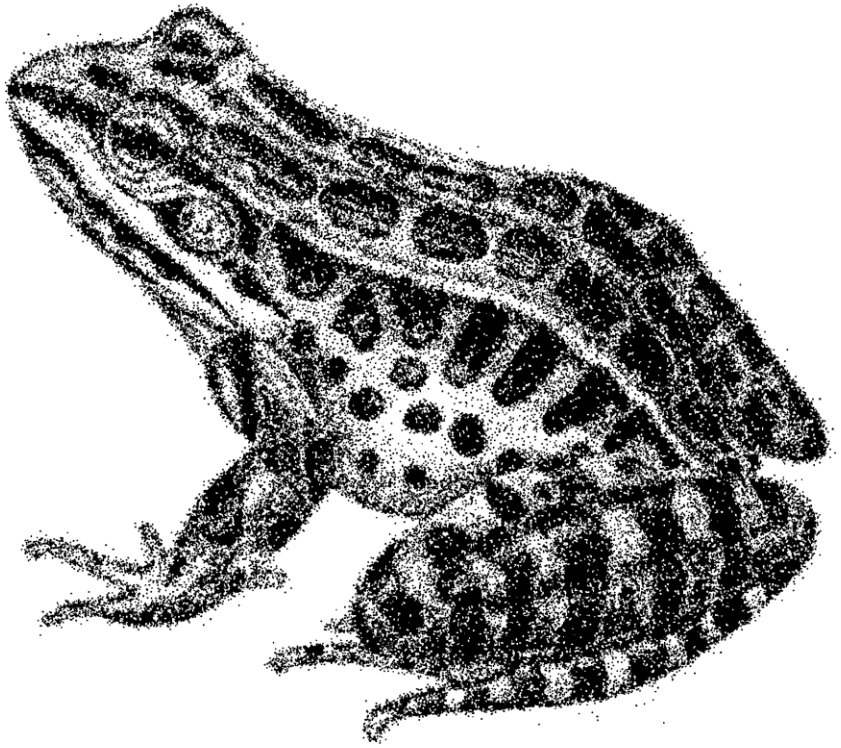


Example-based stipple placement

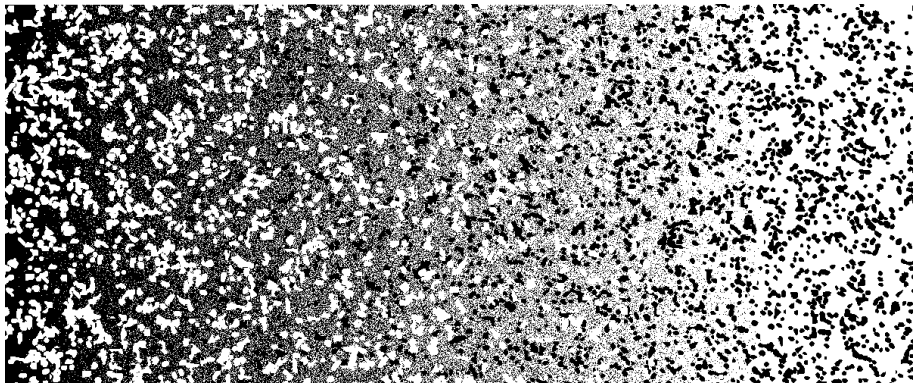


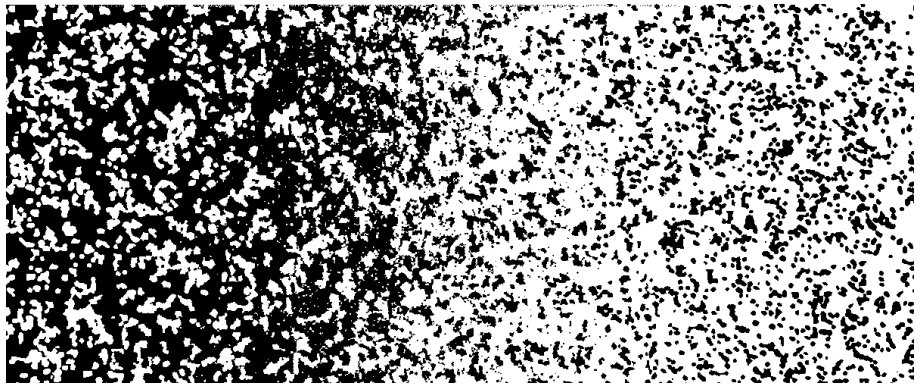
Example-based stipple placement

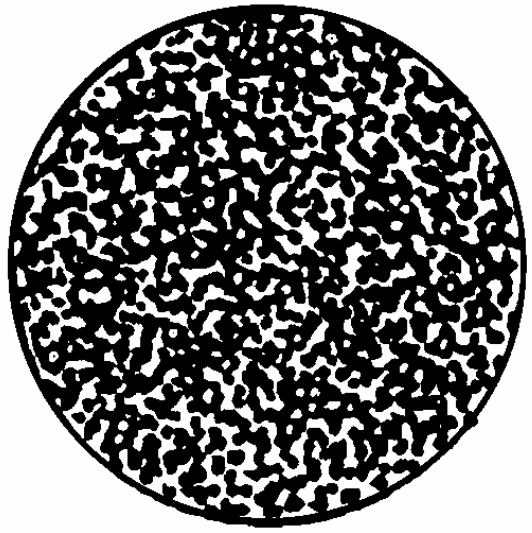


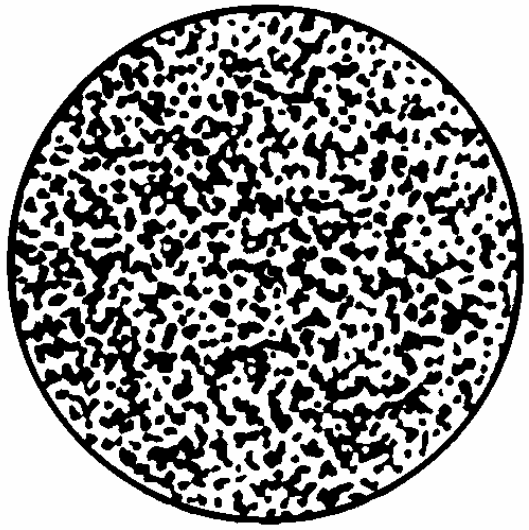


Example-based stippling: Problem



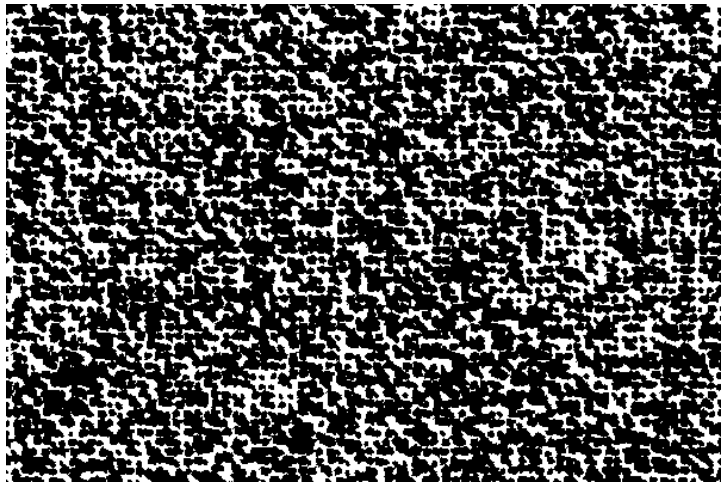


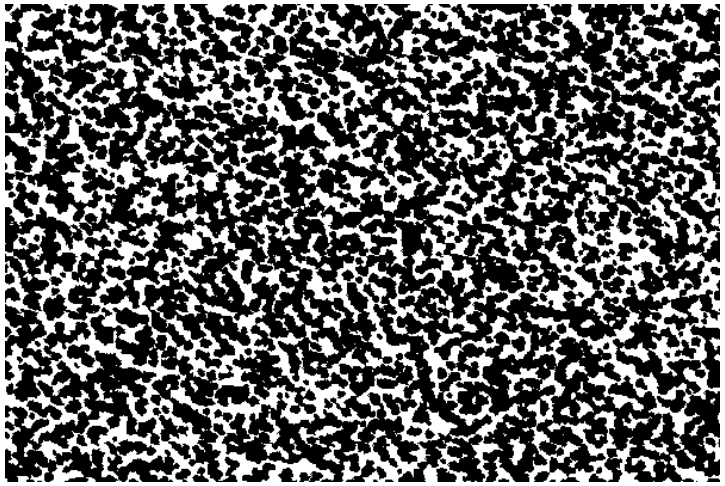


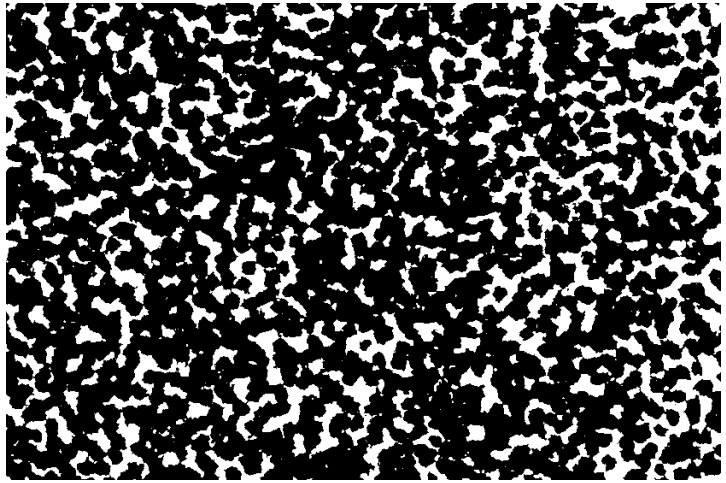


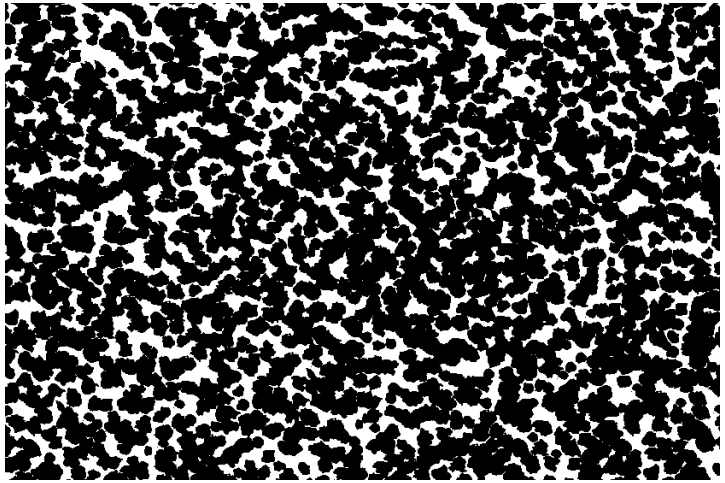
Solving the problem: Grayscale stippling

Grayscale stippling, halftoned distribution

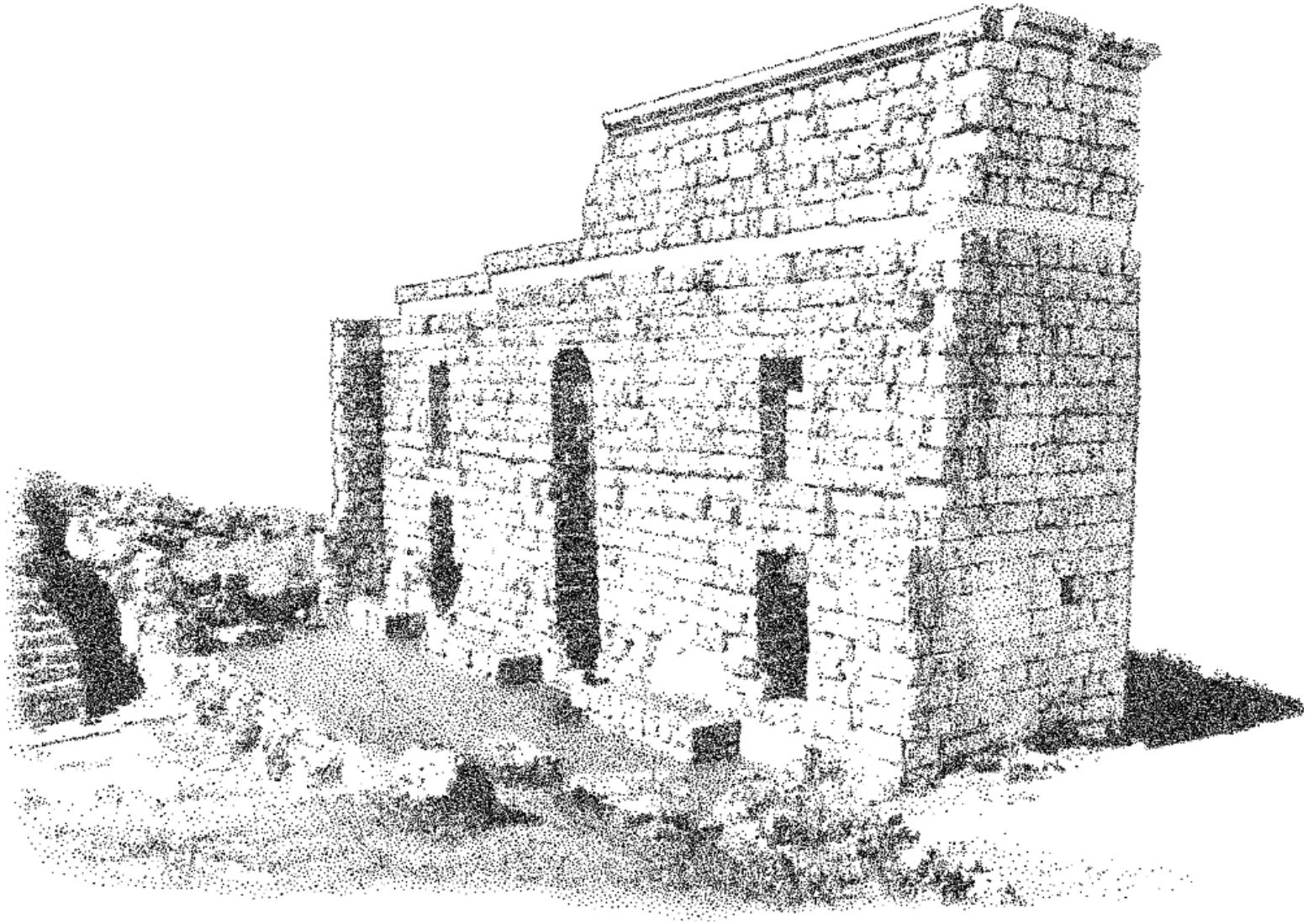




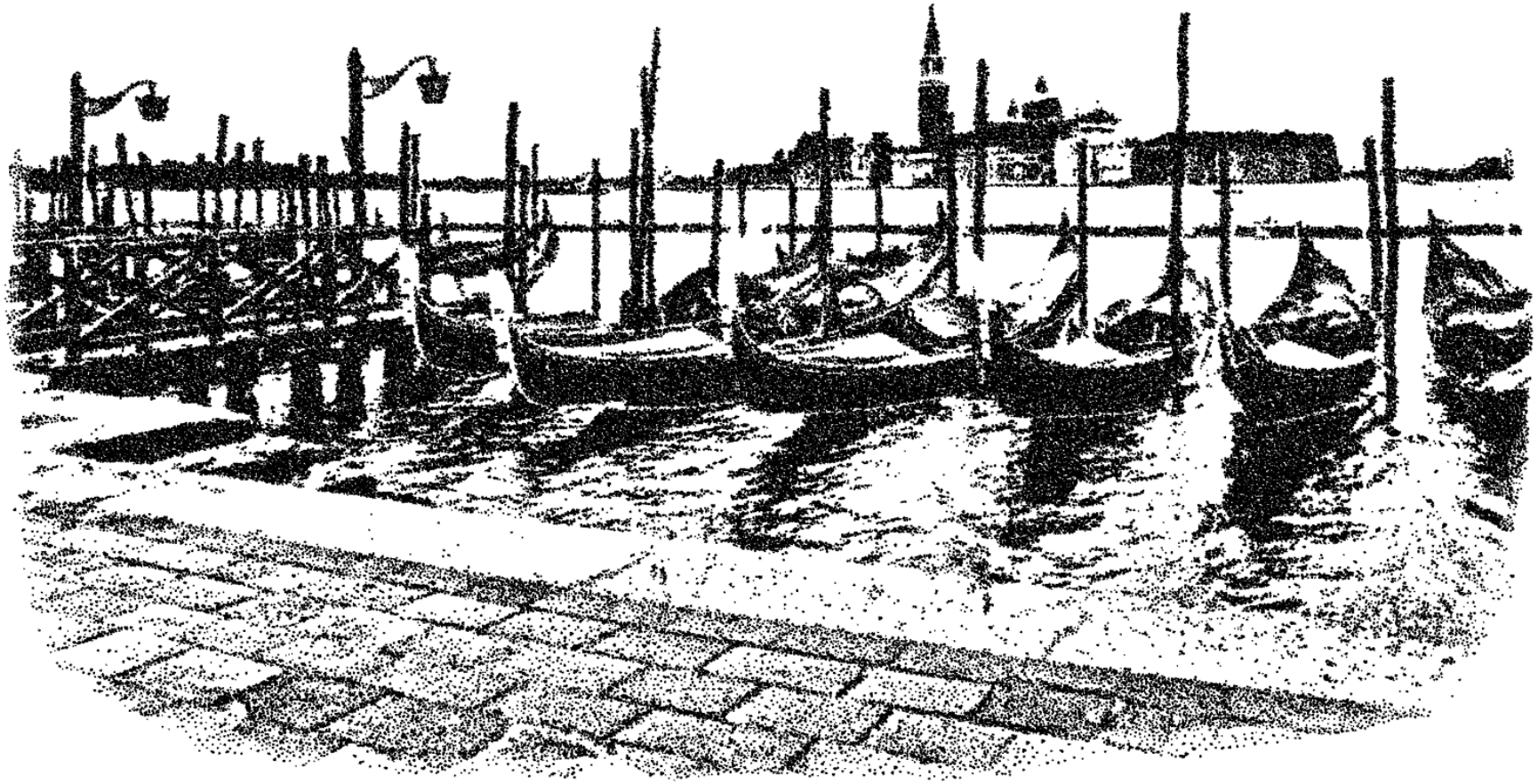


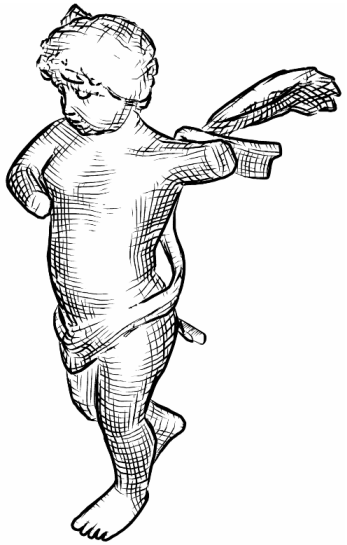


Example-based grayscale stippling: Result



Example-based grayscale stippling: Result

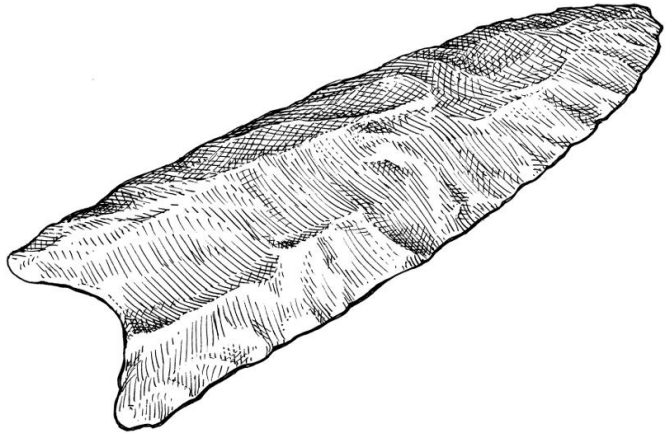




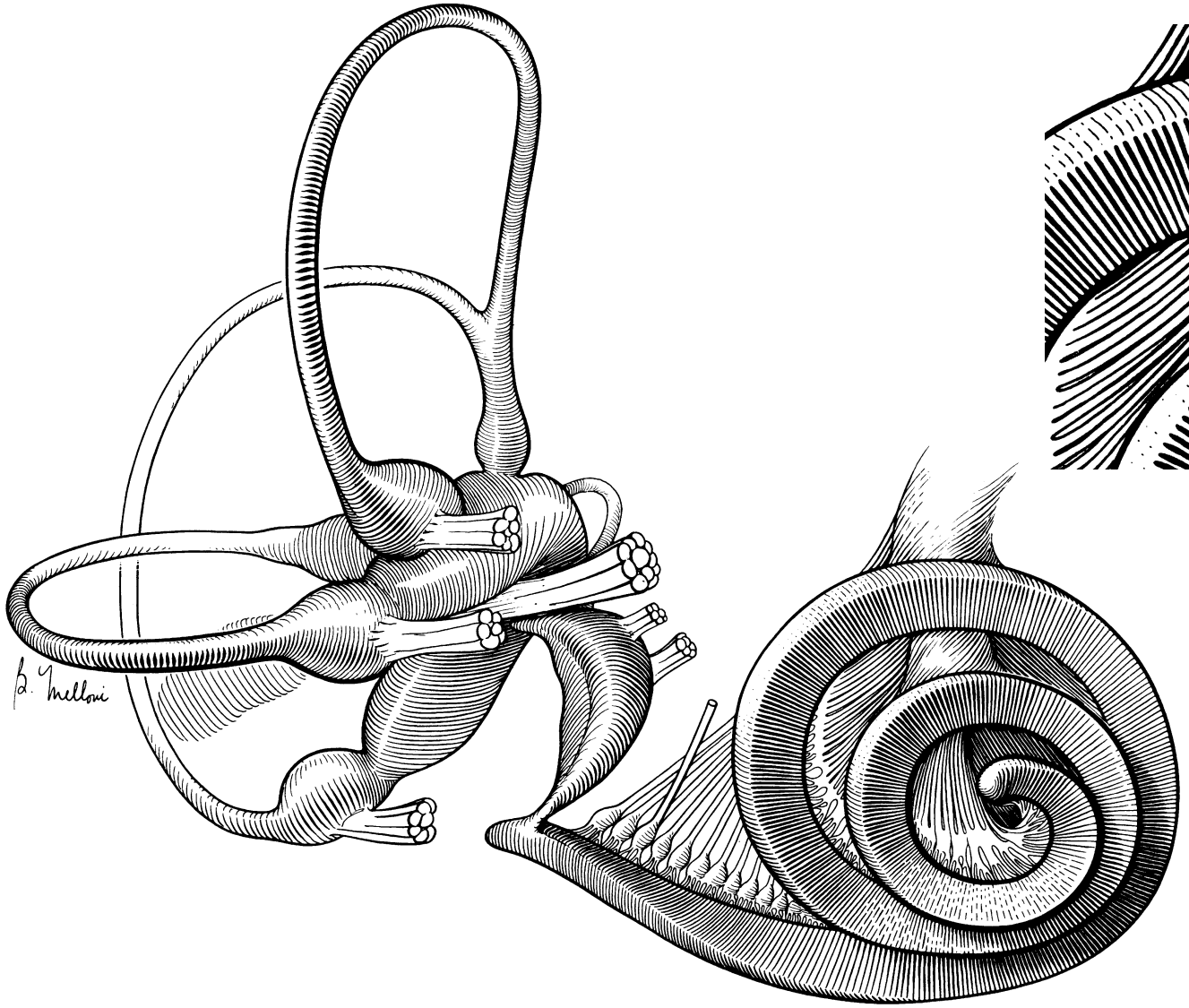
Hatching

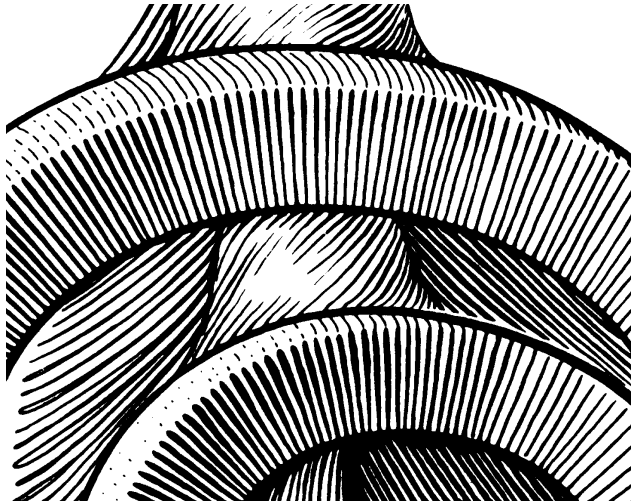
Hatching: Rendering with lines

- 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

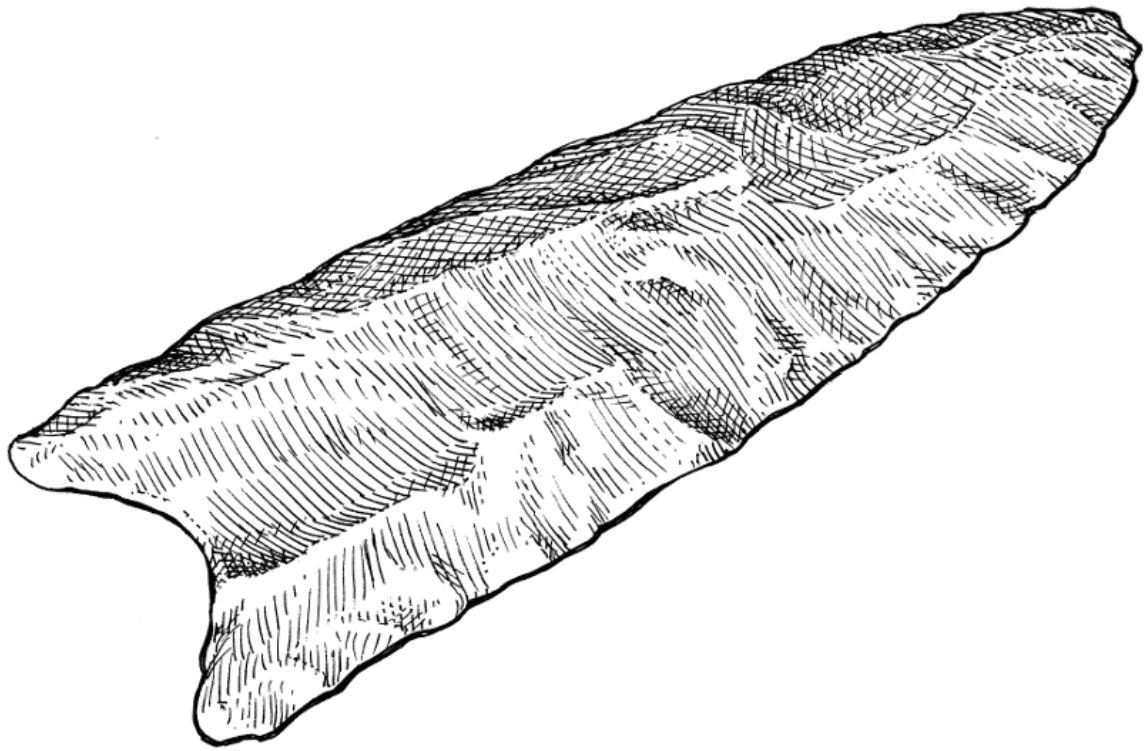


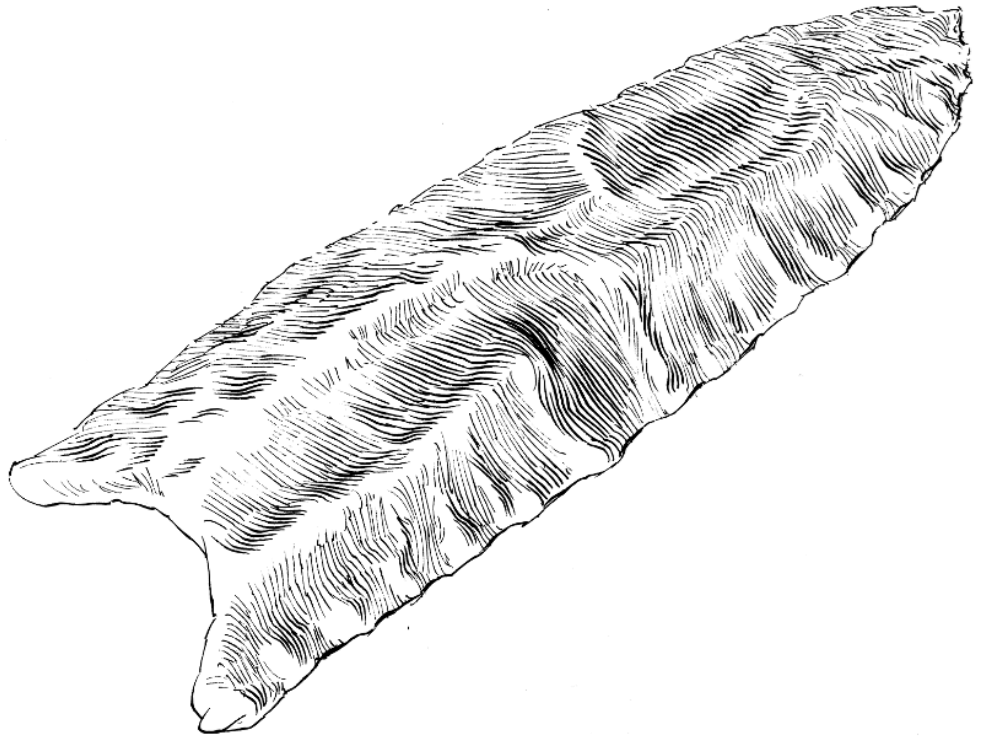
Hatching: Rendering with lines





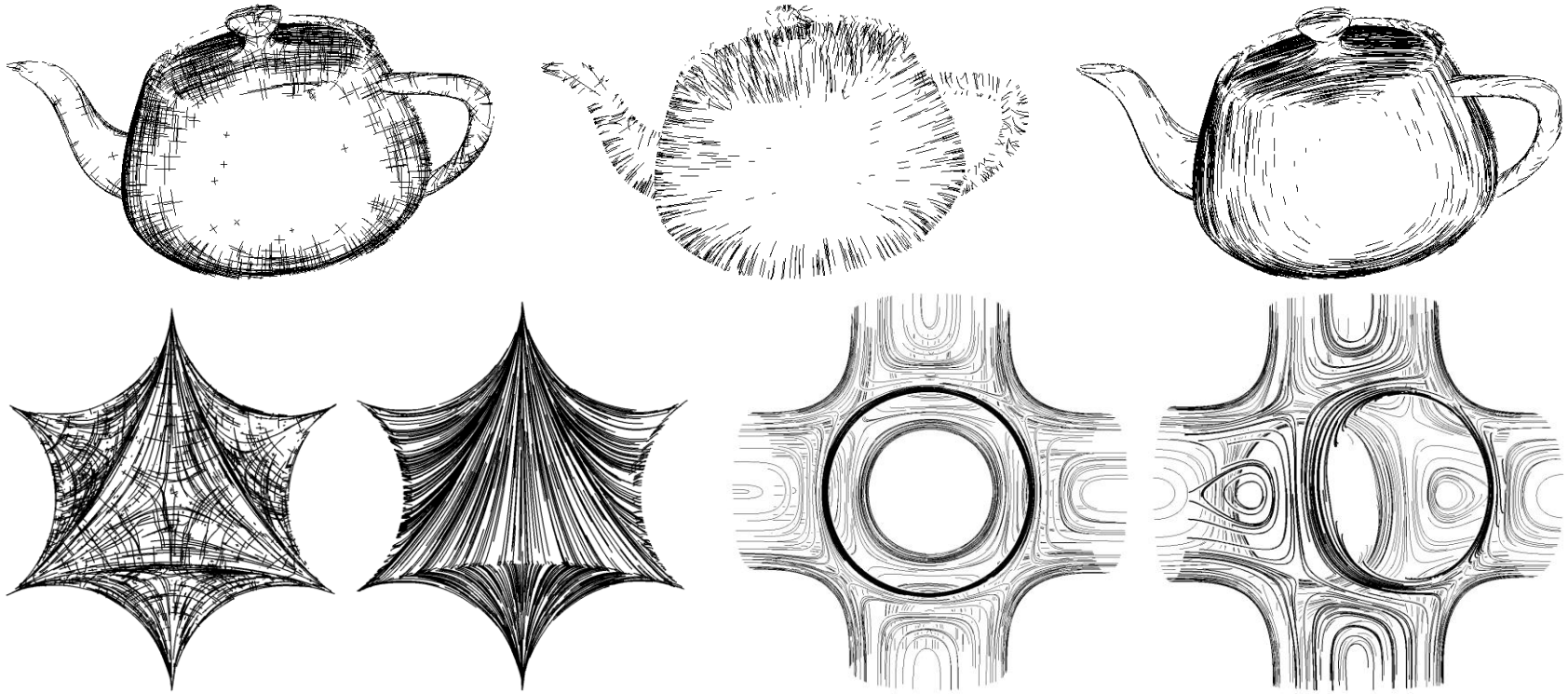
Hatching: Rendering with lines





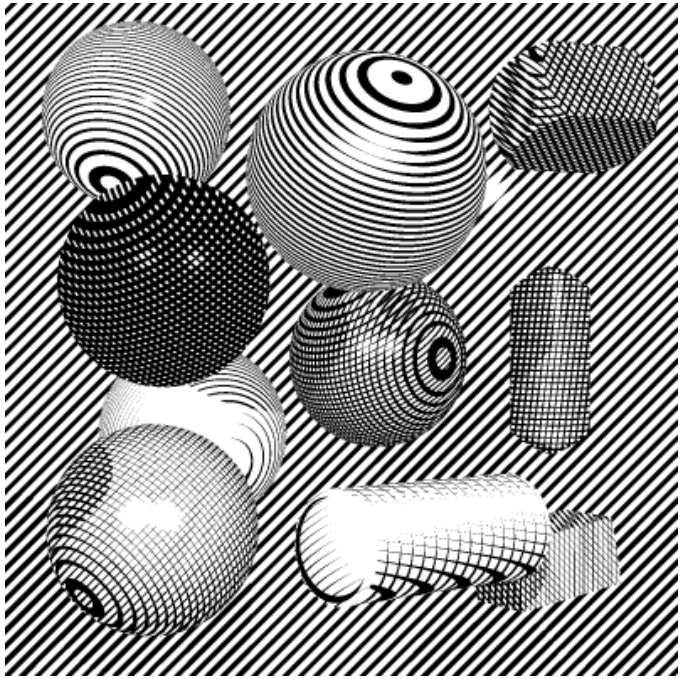
Representing shape and tonal values

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

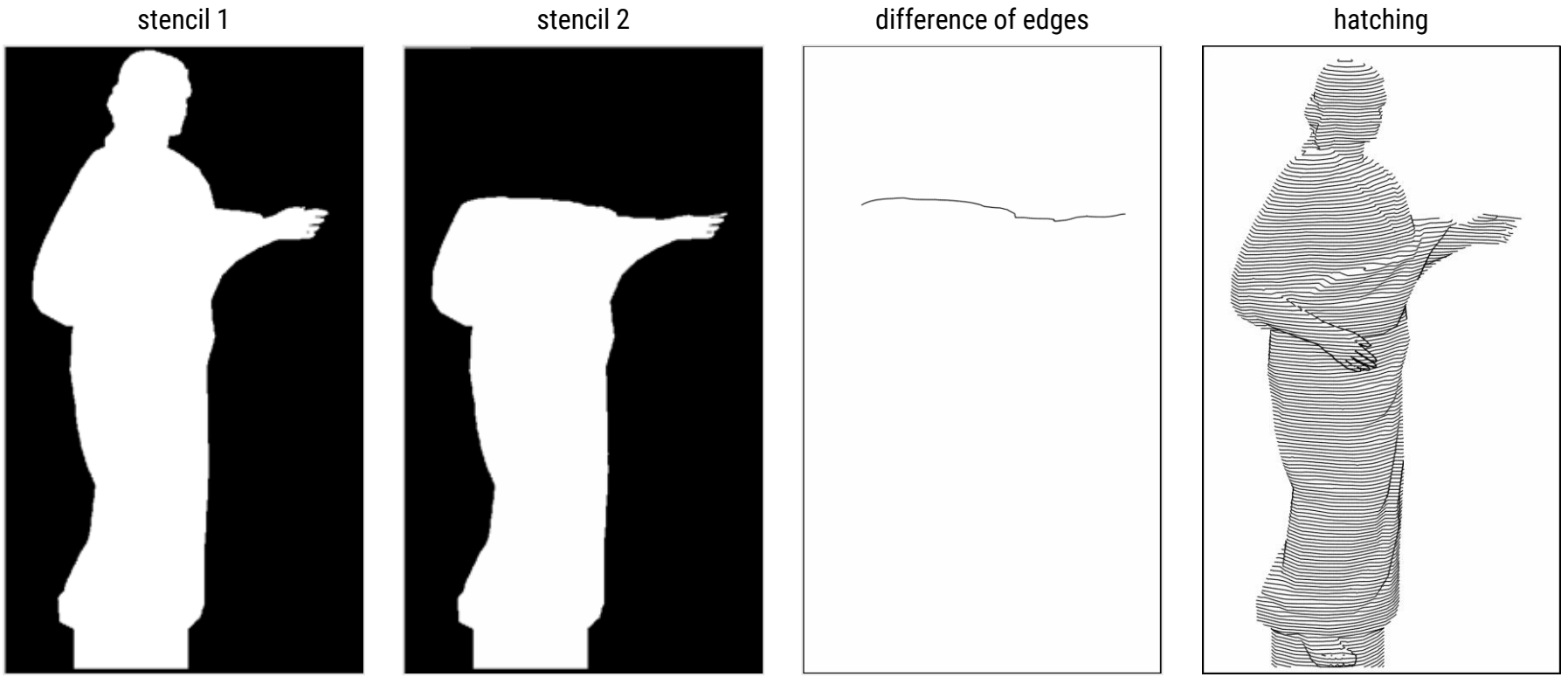


Simple hatching techniques

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



Slice-Based Hatching for Meshes

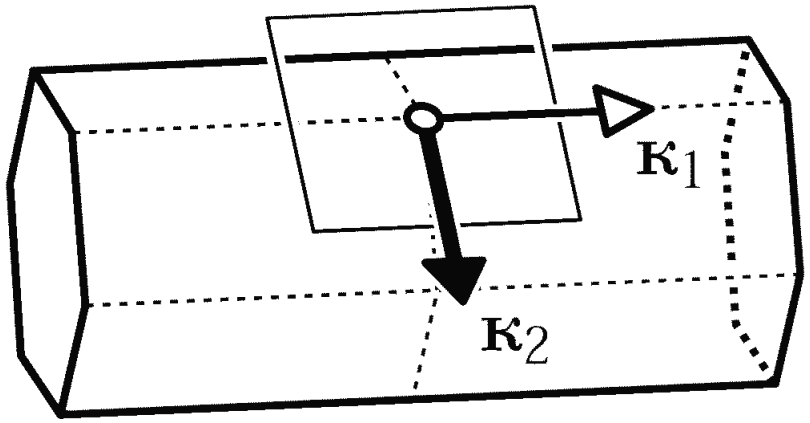


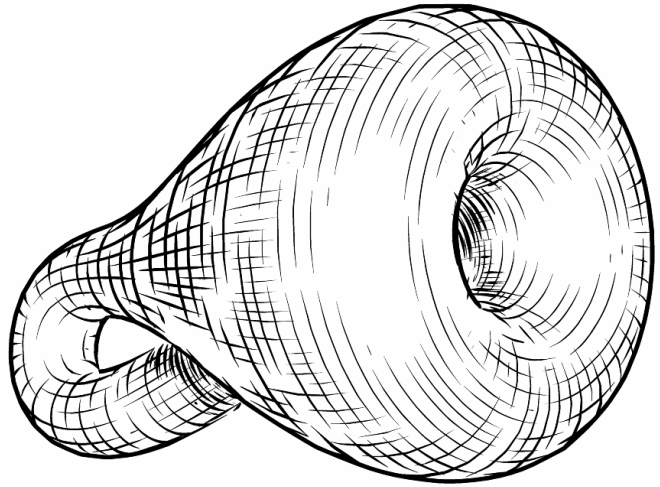
Deussen et al., 1999

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

Hatching to represent smooth surfaces

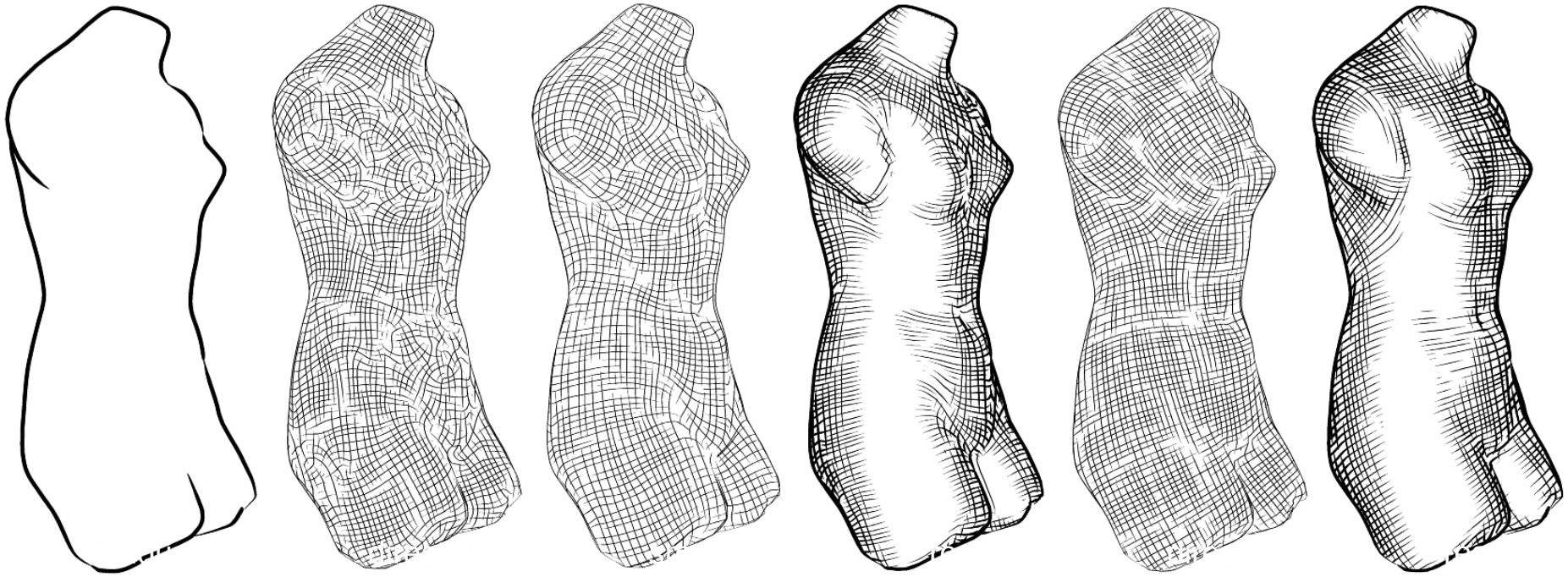
- general concept: meshes represent smooth surfaces
  - extract properties to the “ideal” smooth surfaces
  - principle directions (directions of principle curvatures)
  - direction field, use to guide line placement
  - direction field smoothing to reduce artifacts
  - projection to 2D, trace lines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





Hatching to represent smooth surfaces

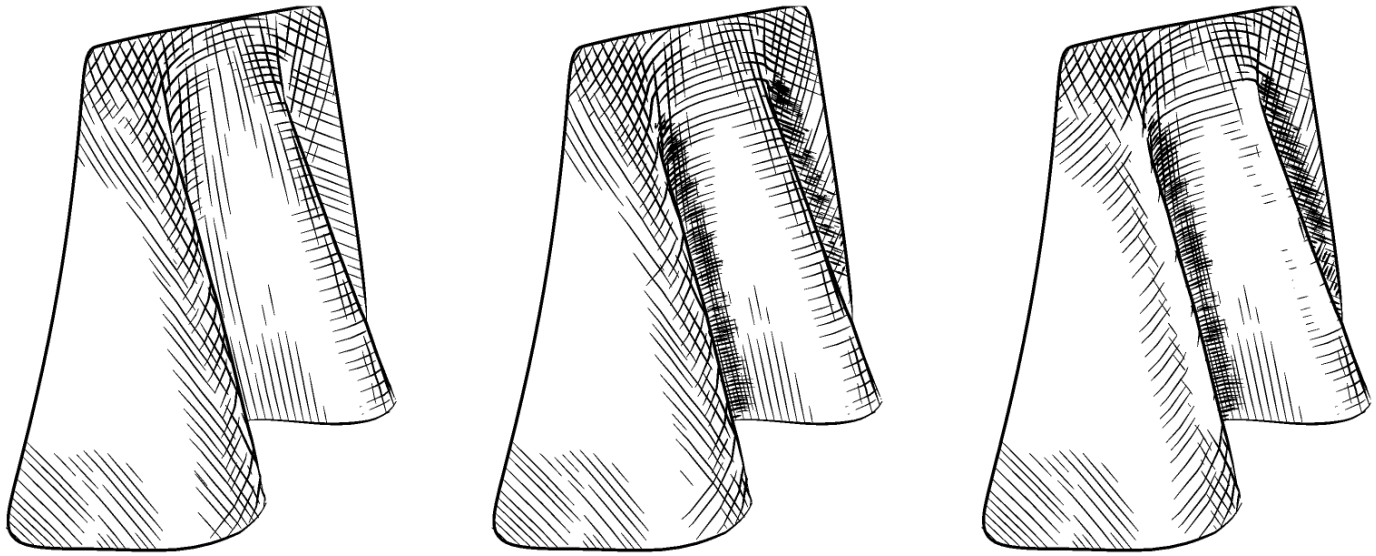
- the process illustrated:

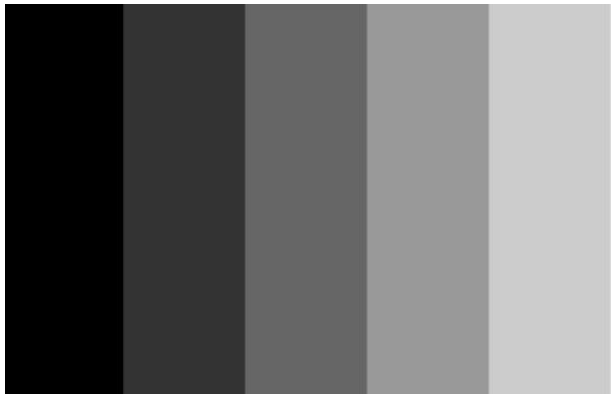


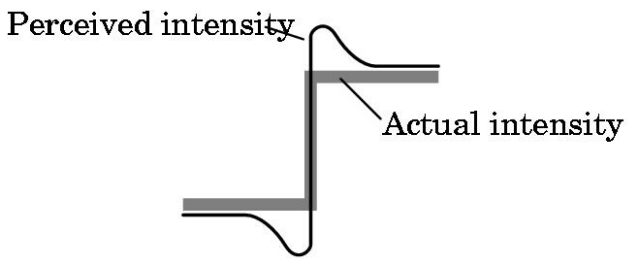
- smoothing based on stability of curvature directions and the variance of directions in local neighborhood

Cross-Hatching and Lighting Visualization

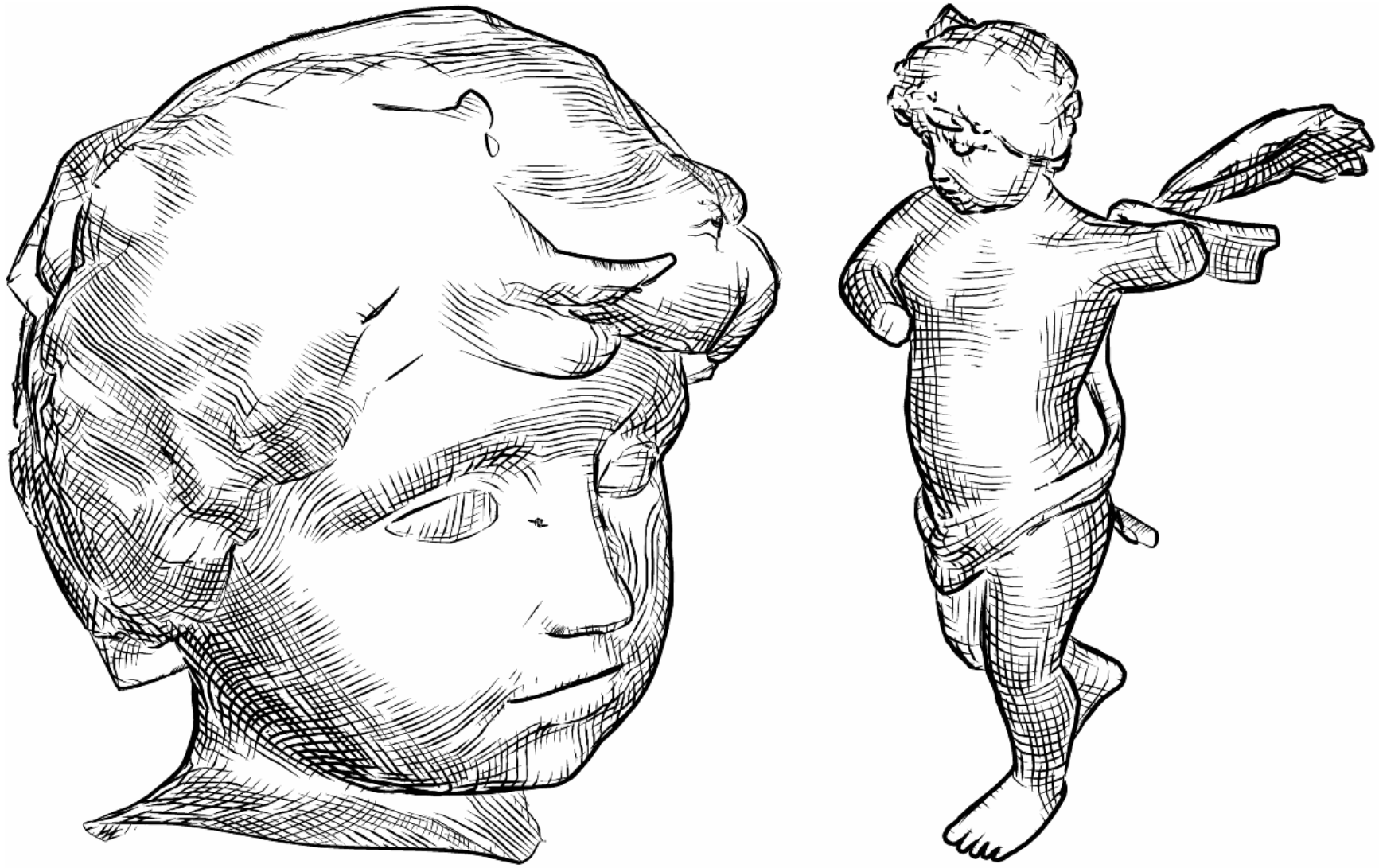
- cross-hatching: using more than one direction
- show illumination by no hatching, single- and cross-hatching
- emphasize illumination discontinuities: undercuts and Mach bands



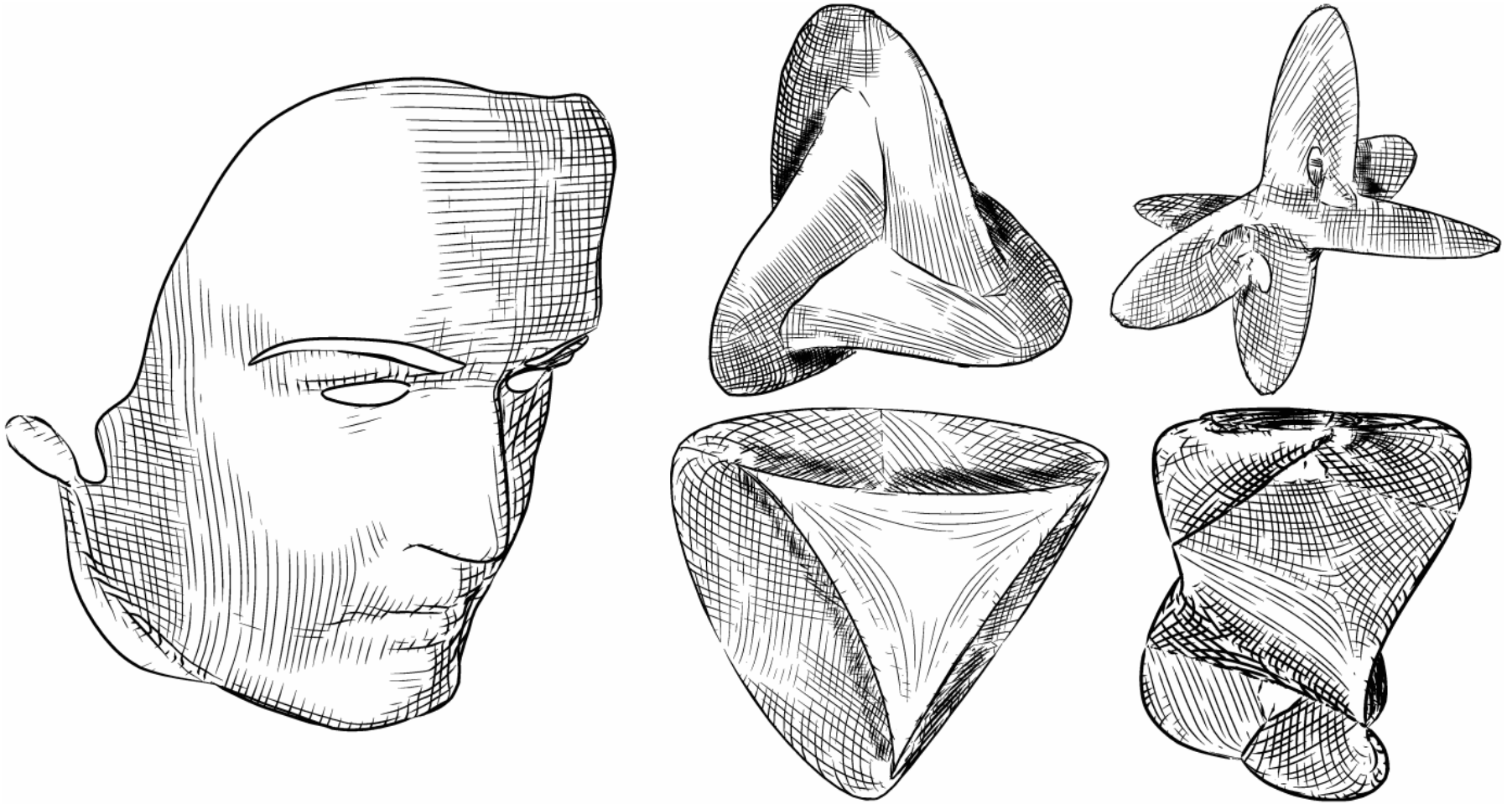




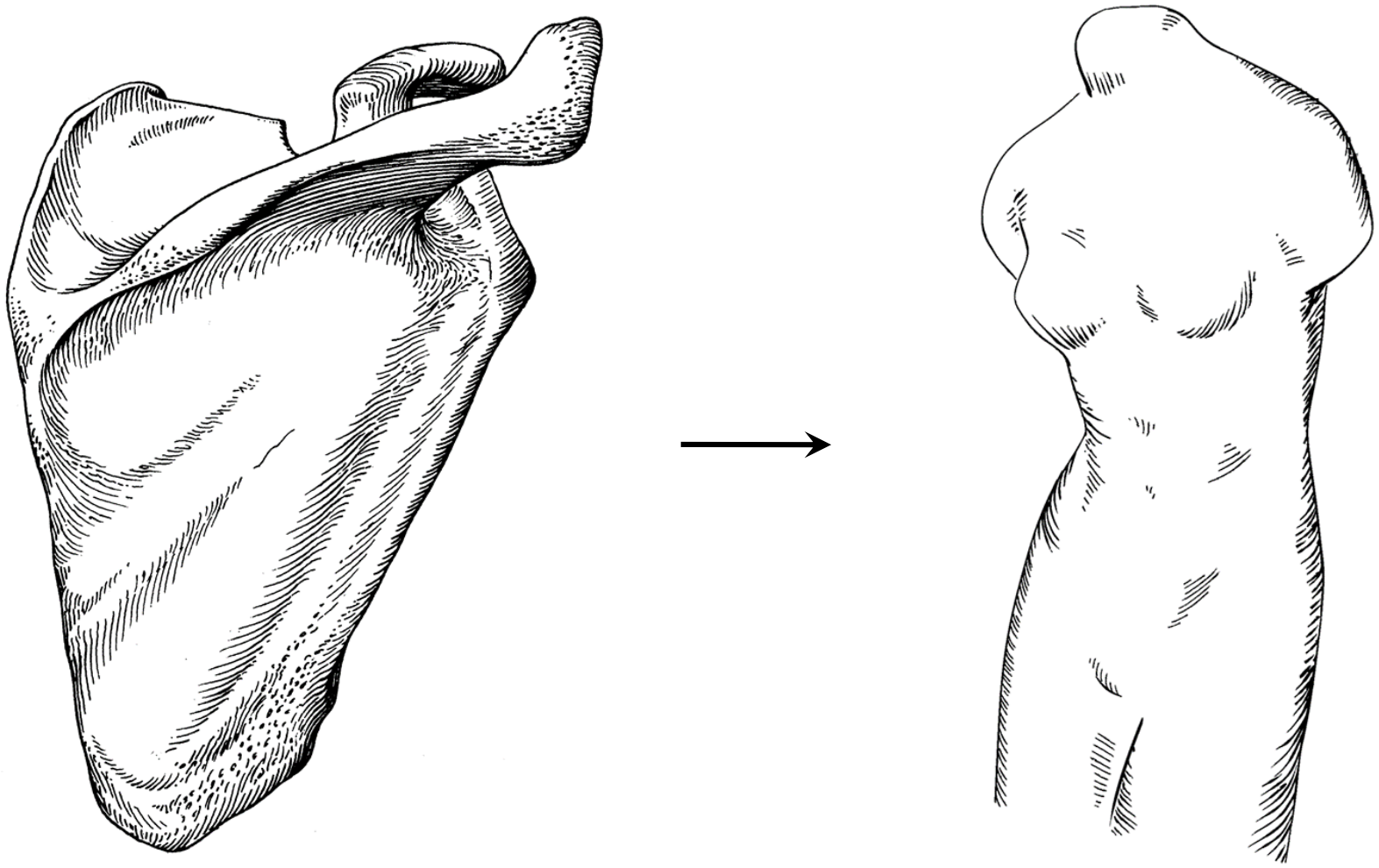
Example results:



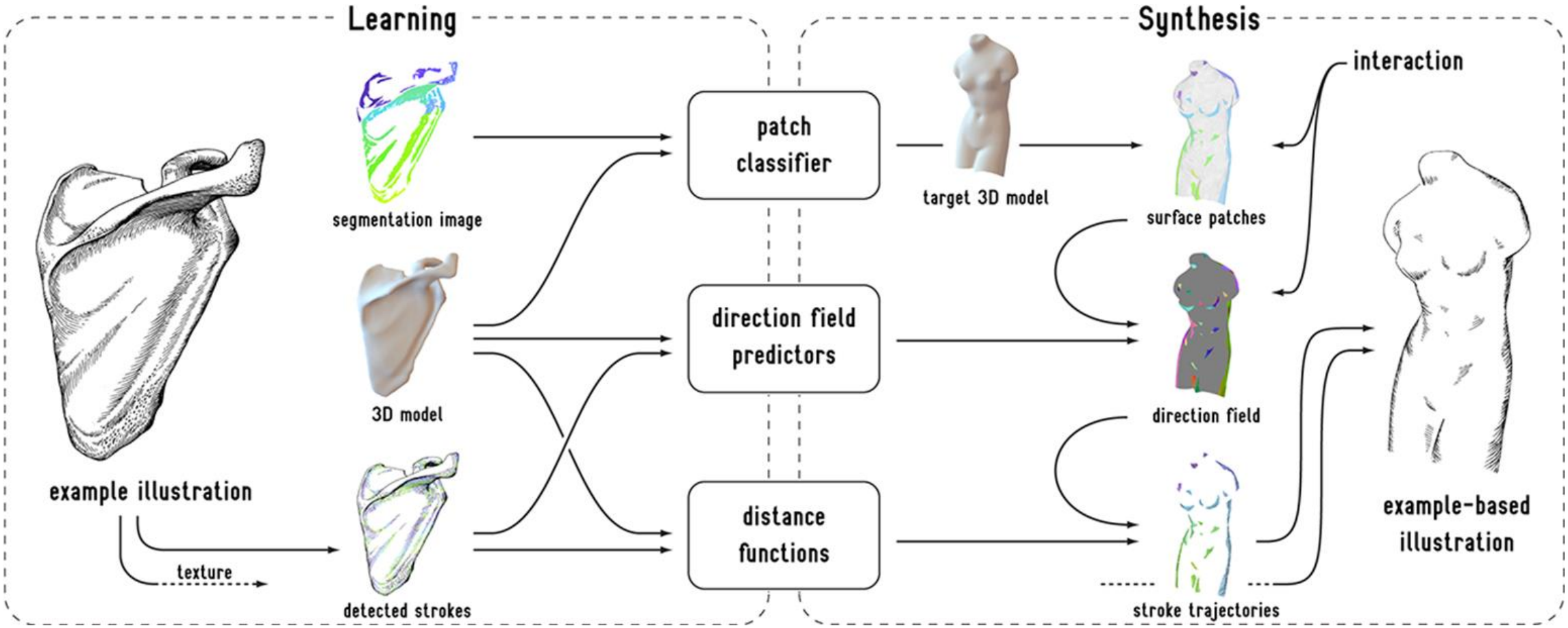
Example results:



Example-based hatching

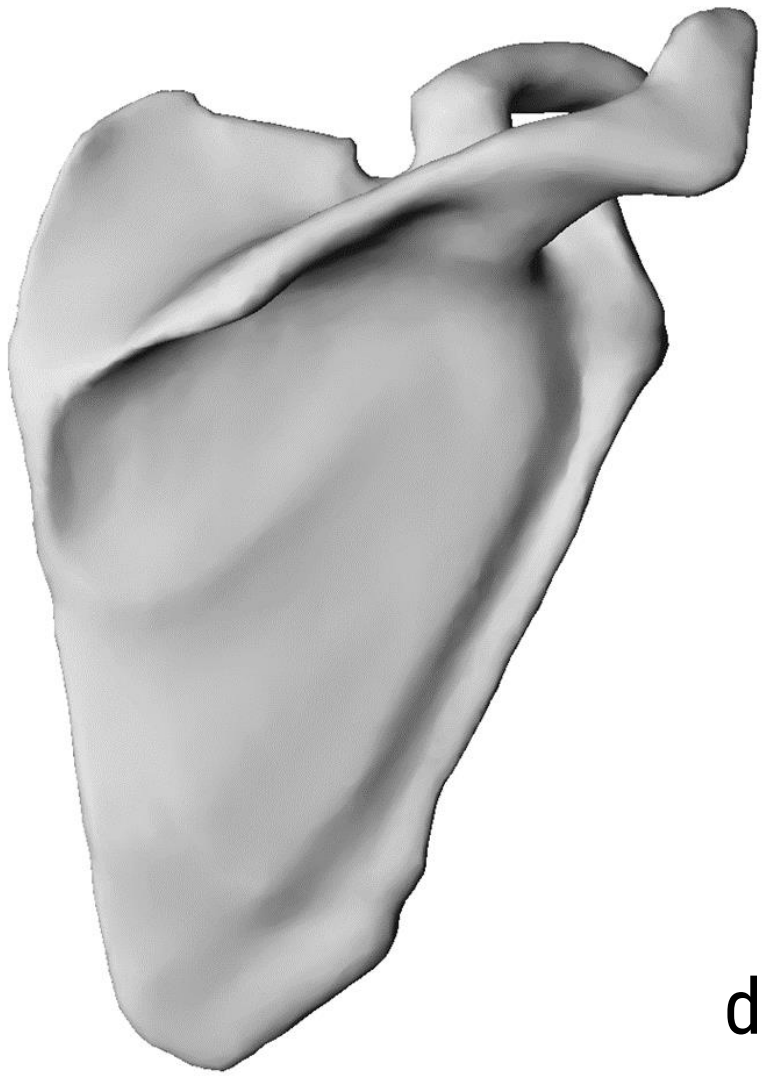


Overview


Learning
Synthesis
example illustration
texture
segmentation image
3D model
detected strokes
patch classifier
direction field predictors
distance functions
target 3D model
surface patches
direction field
stroke trajectories
interaction
example-based illustration

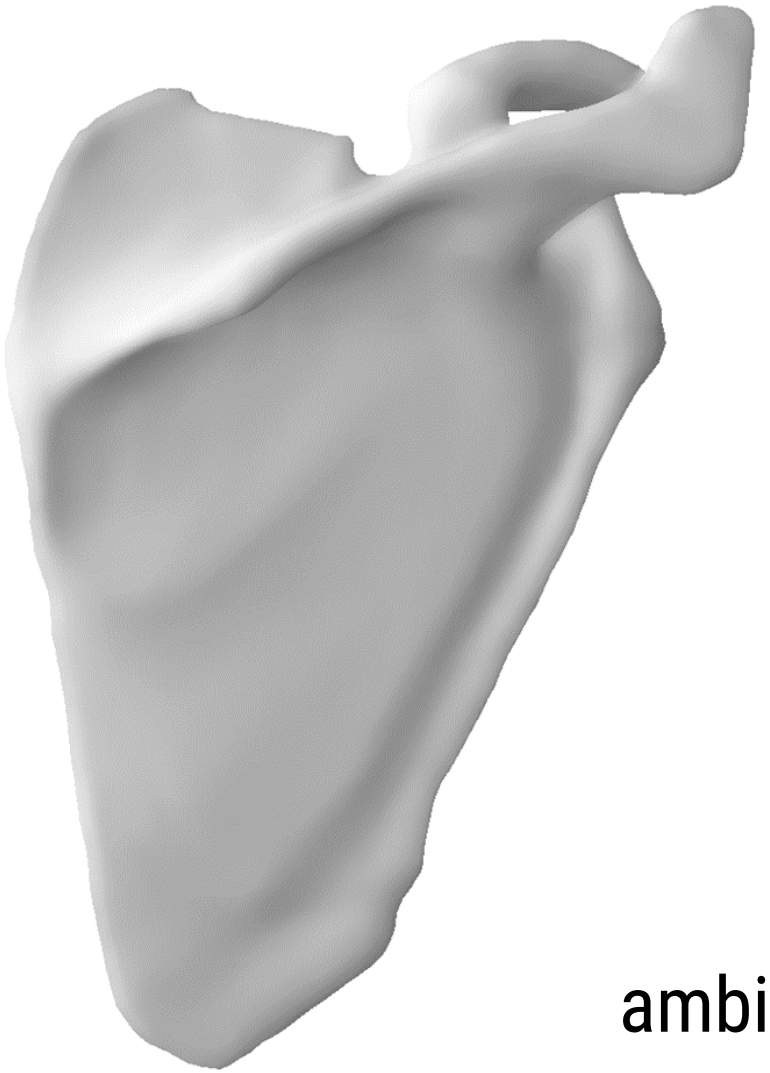
Input to learning

Features



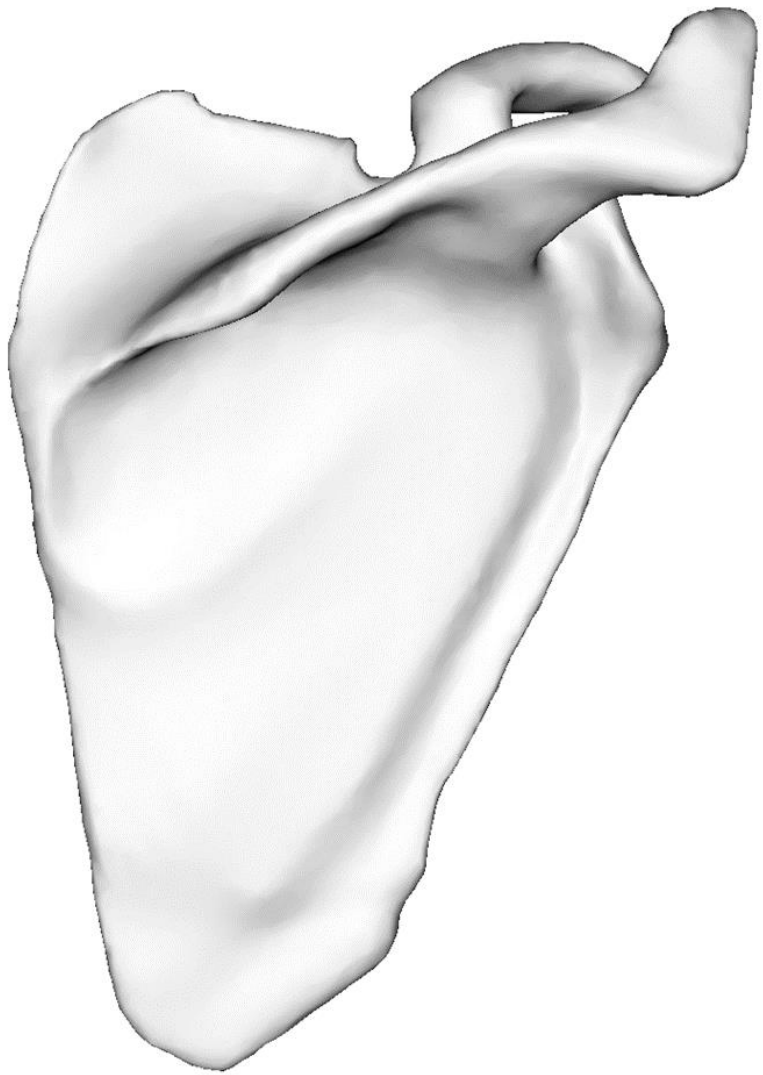
diffuse lighting

Features



ambient occlusion

Features



facing ratio

Features

curvature

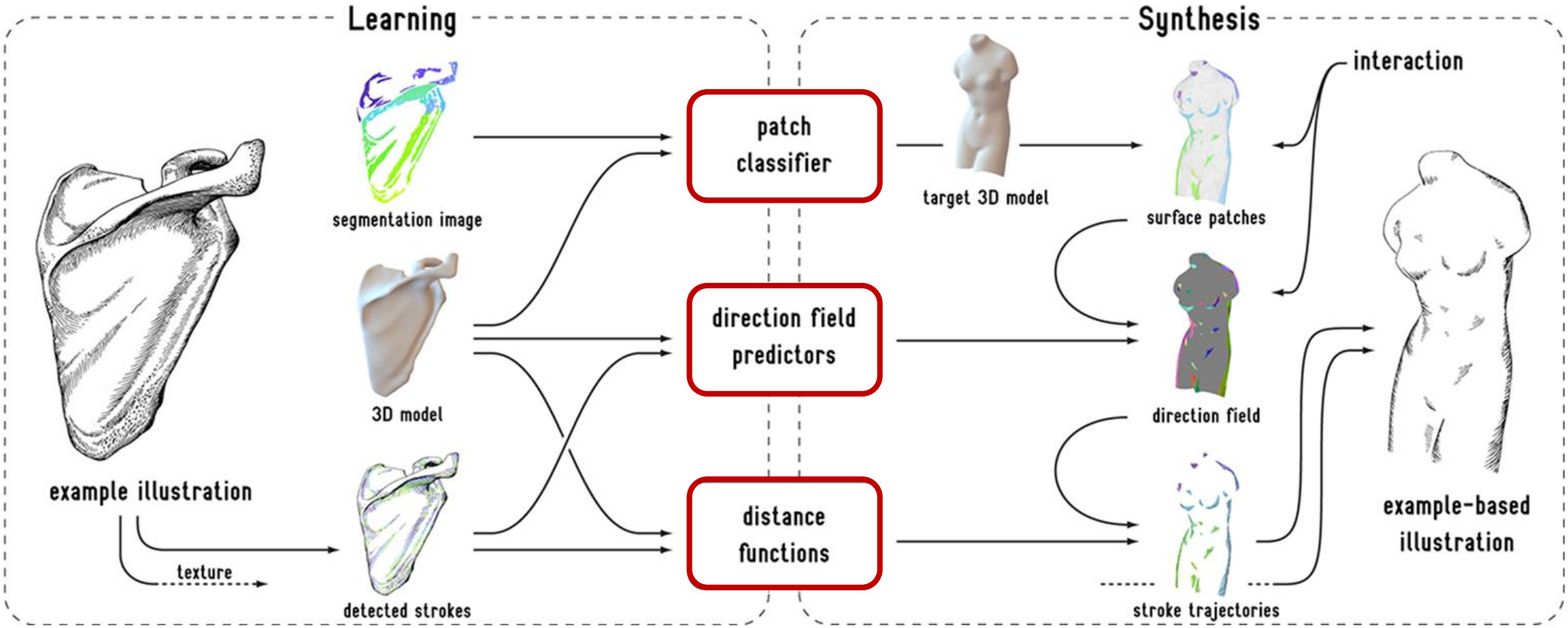
Features

normal

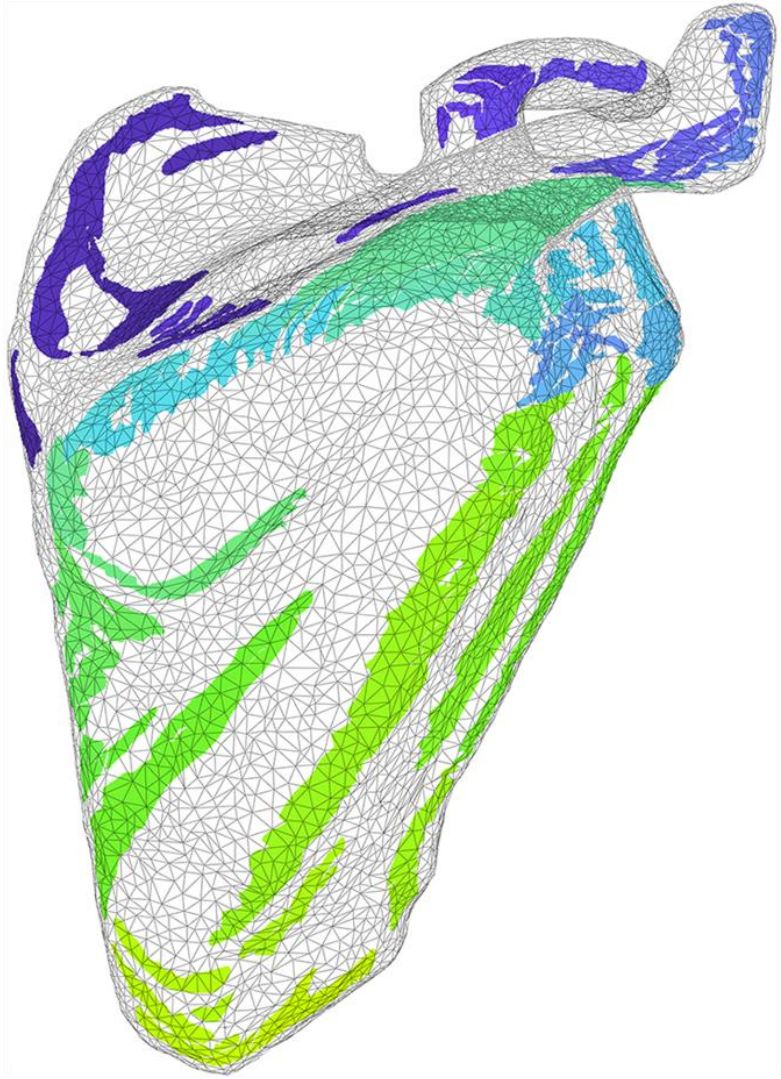
Features

depth

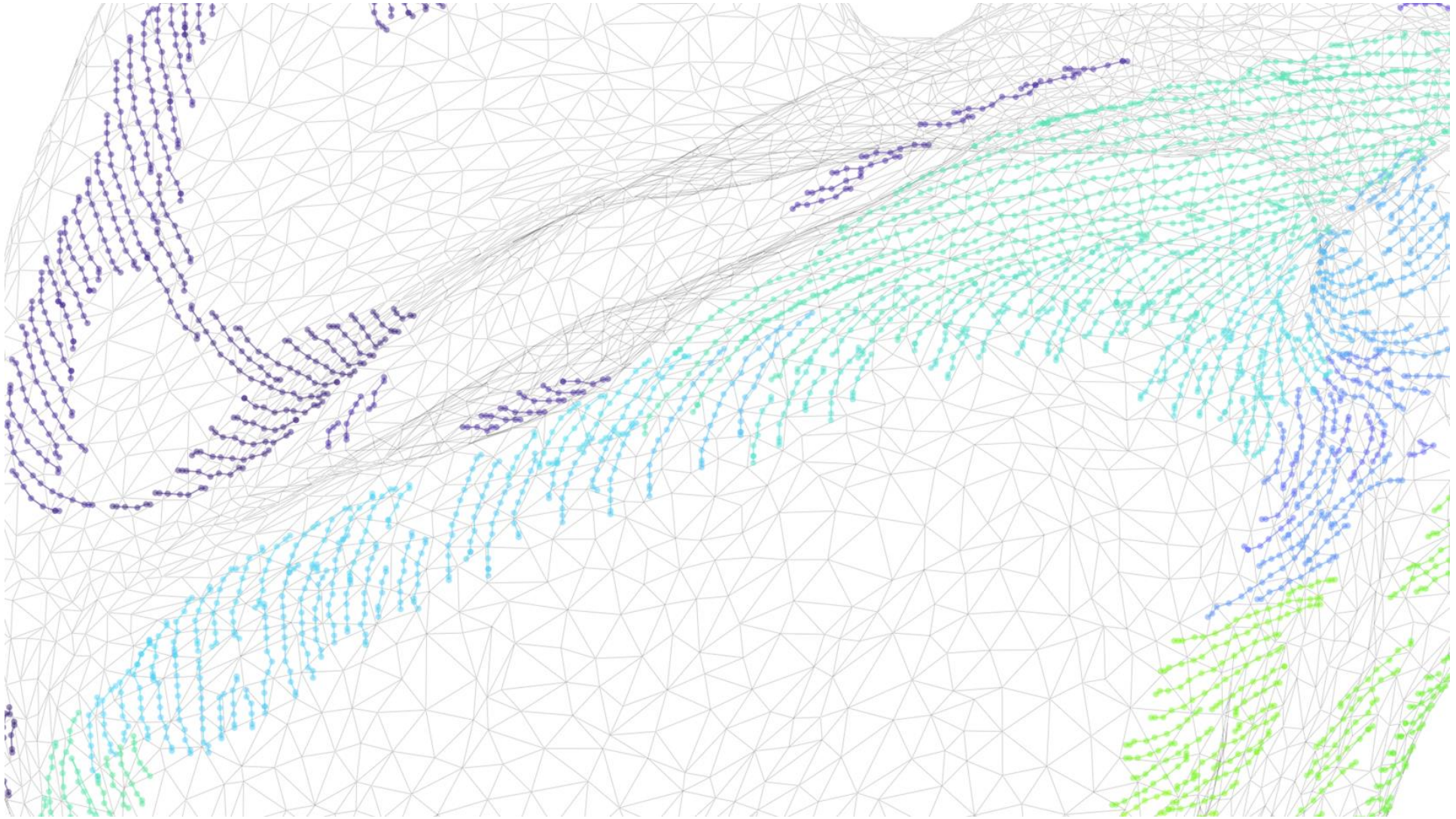
Overview


Learning
Synthesis
example illustration
texture
segmentation image
3D model
detected strokes
patch classifier
direction field predictors
distance functions
target 3D model
surface patches
direction field
stroke trajectories
interaction
example-based illustration

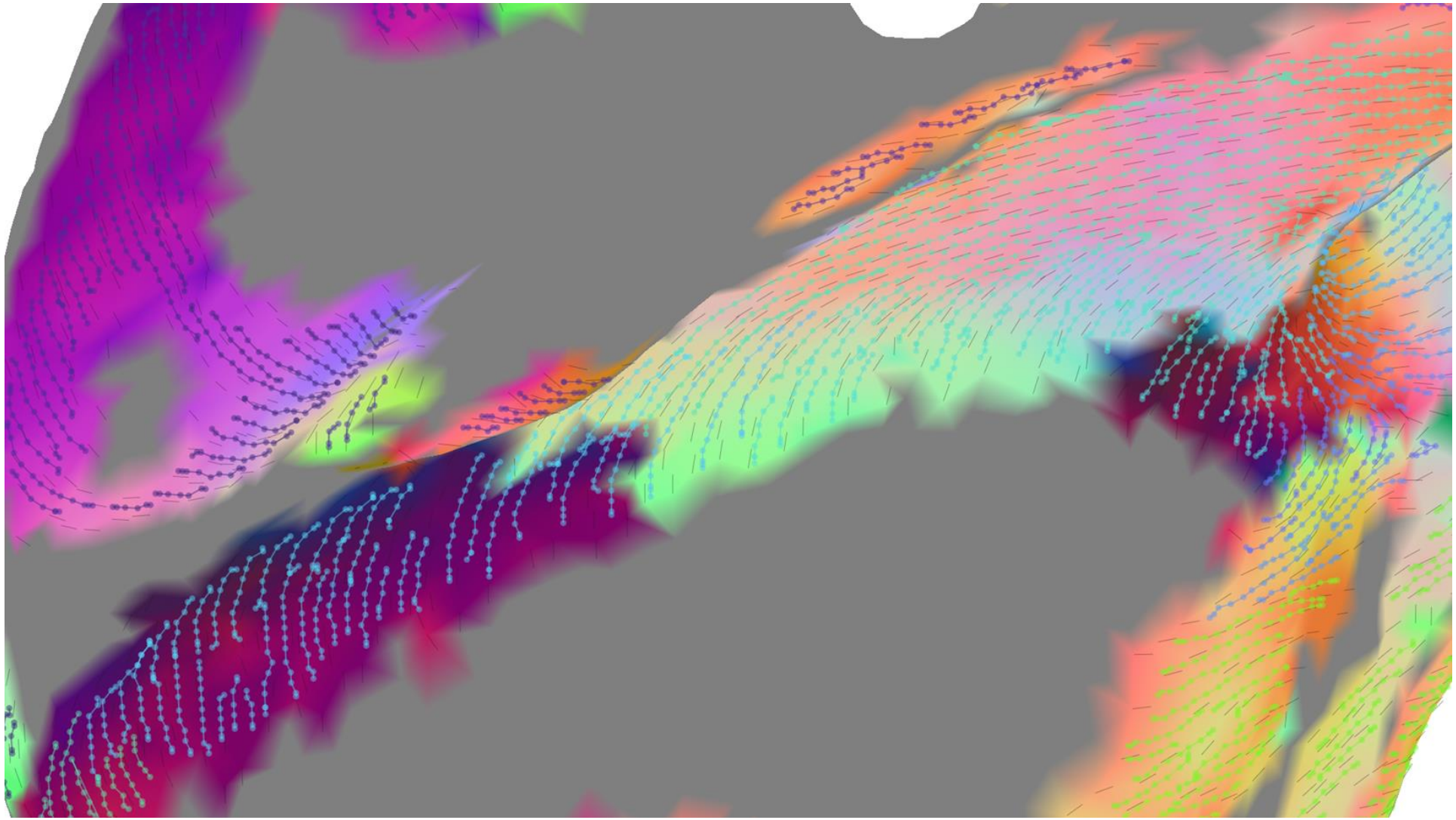
Learning regions/patches



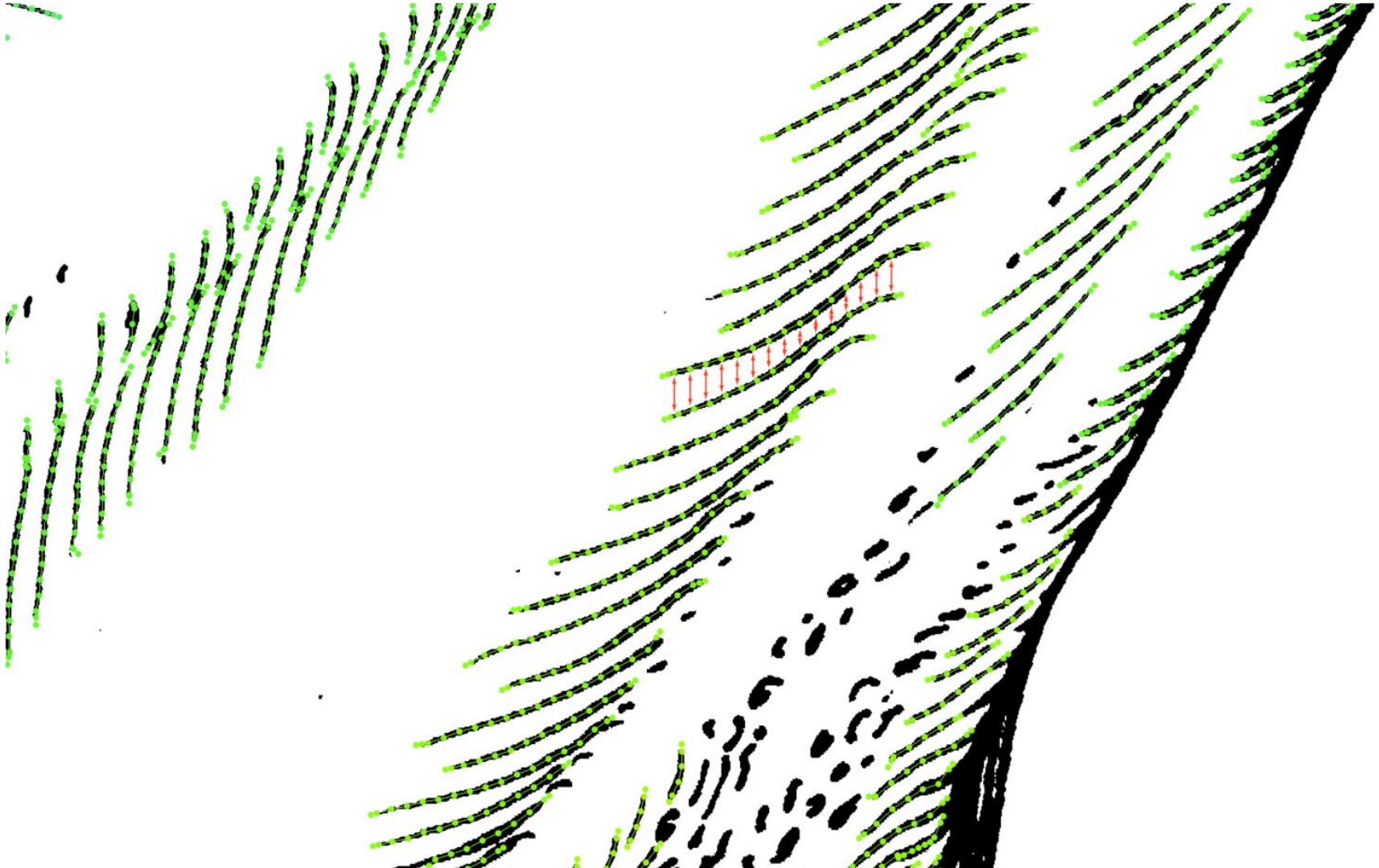
Learning directions: Strokes on surface



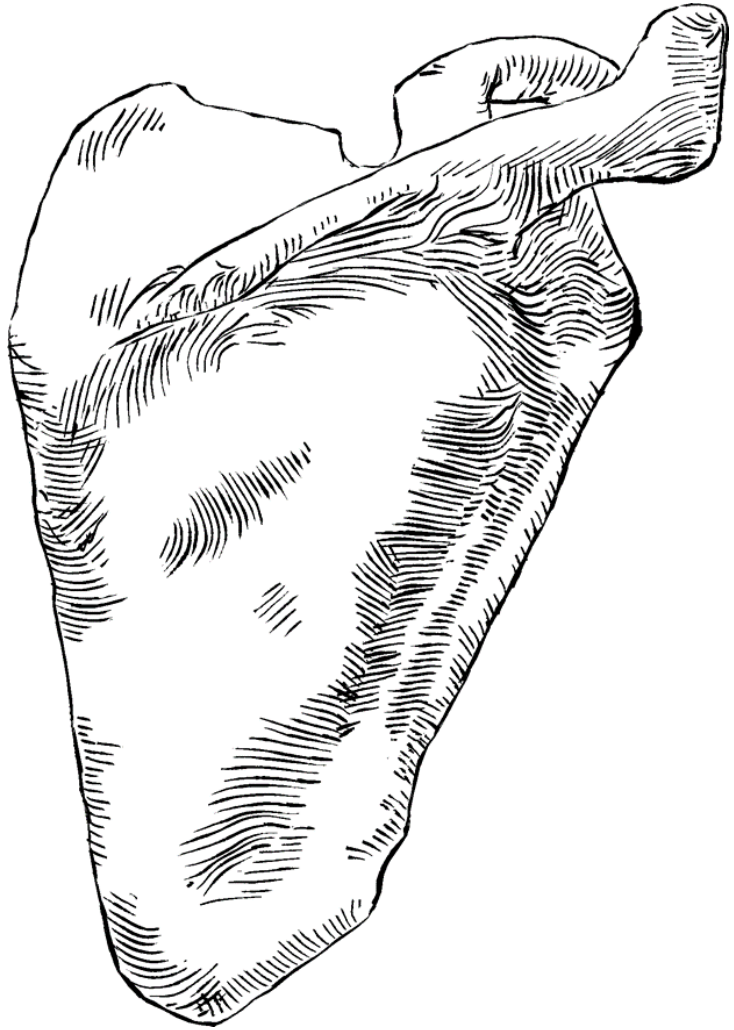
Learning directions: Derive direction field



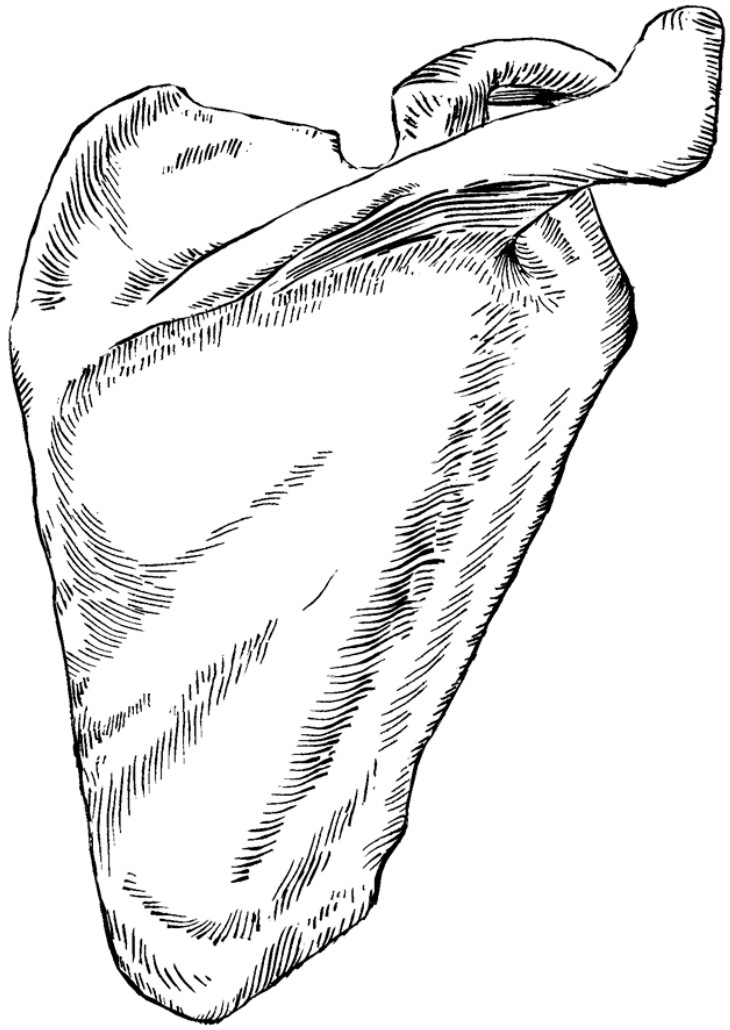
Learning distances



Automatic & semi-automatic synthesis

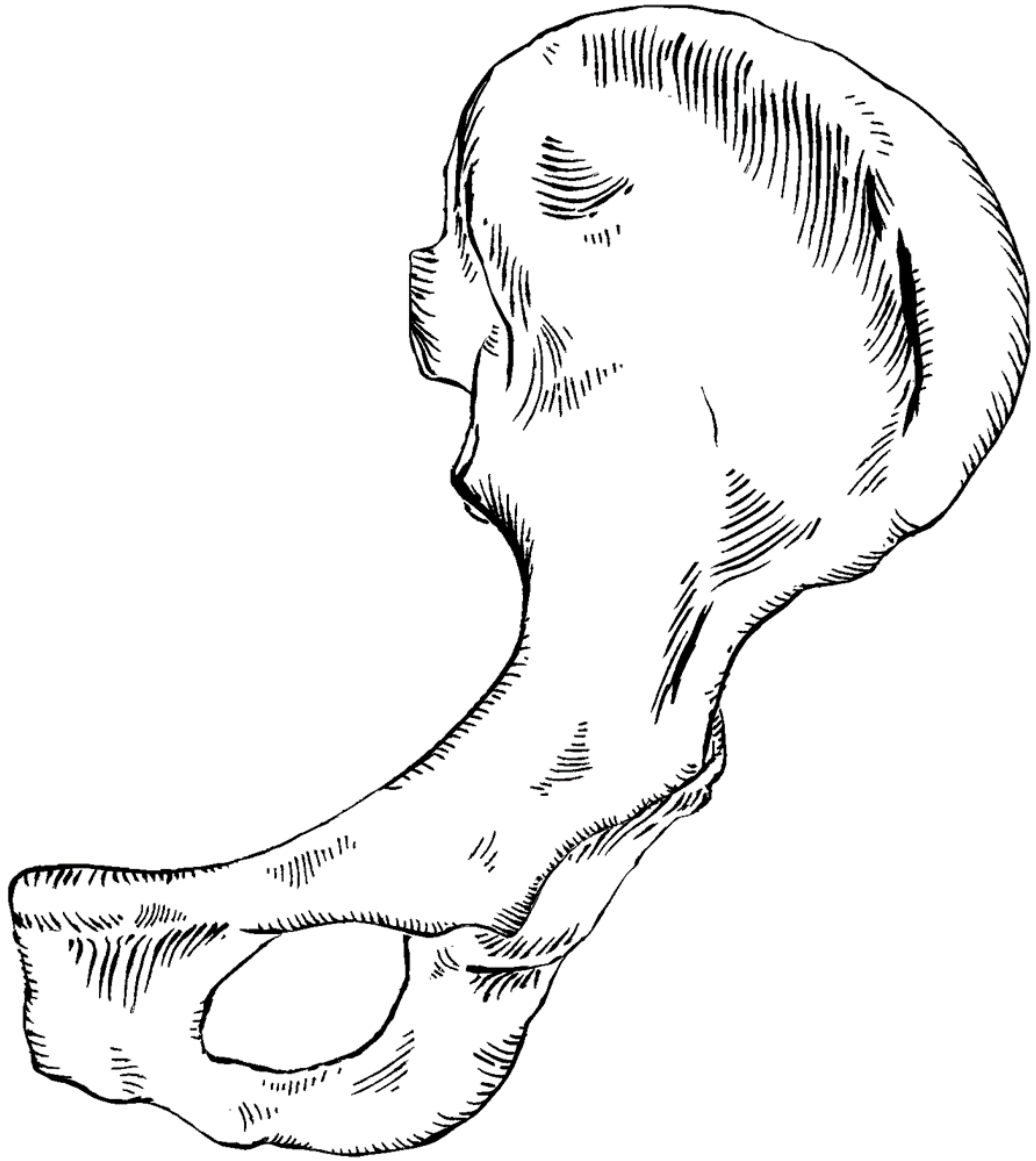


automatic

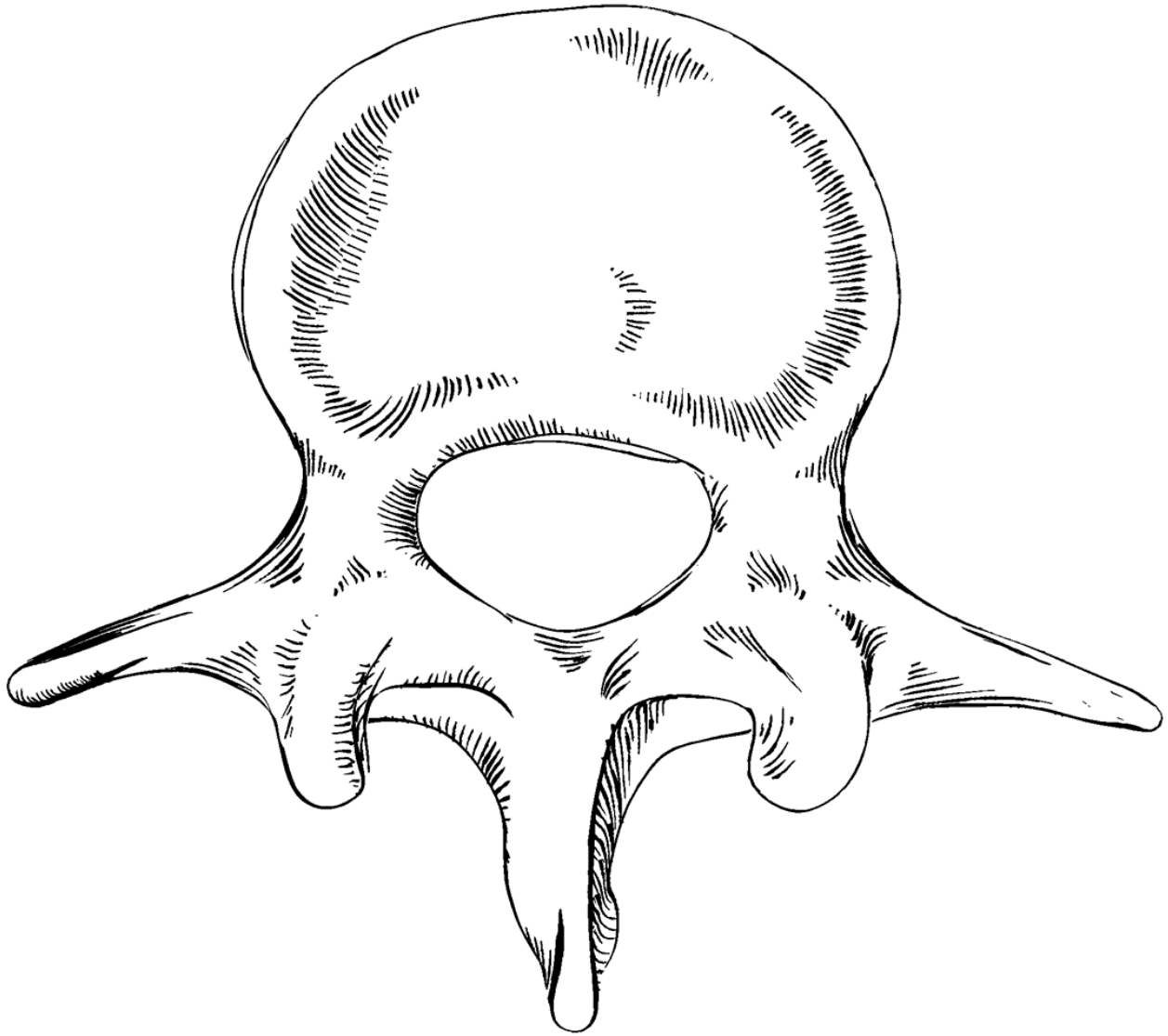


semi-automatic

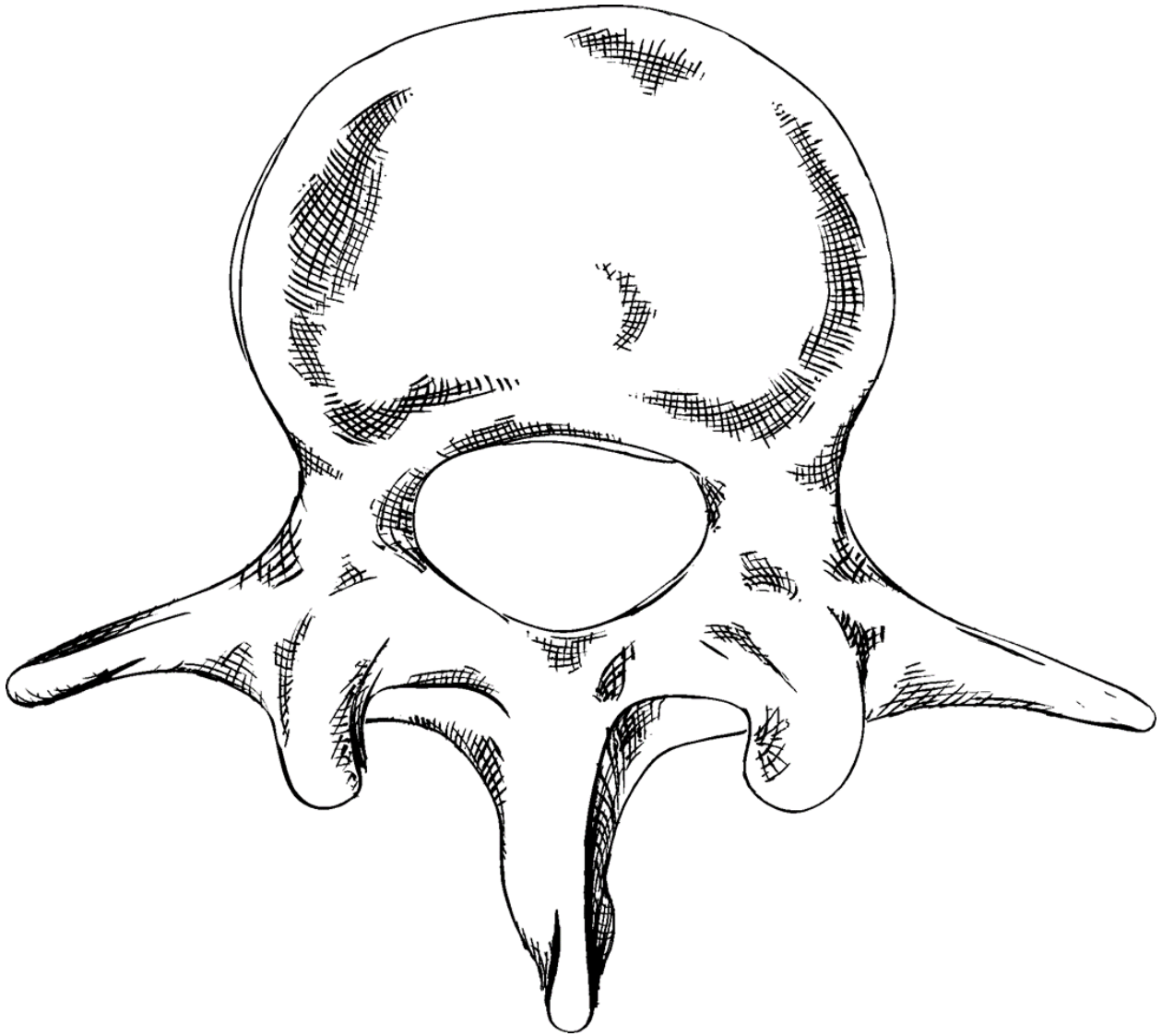
Results



Results



Results



Deep Learning

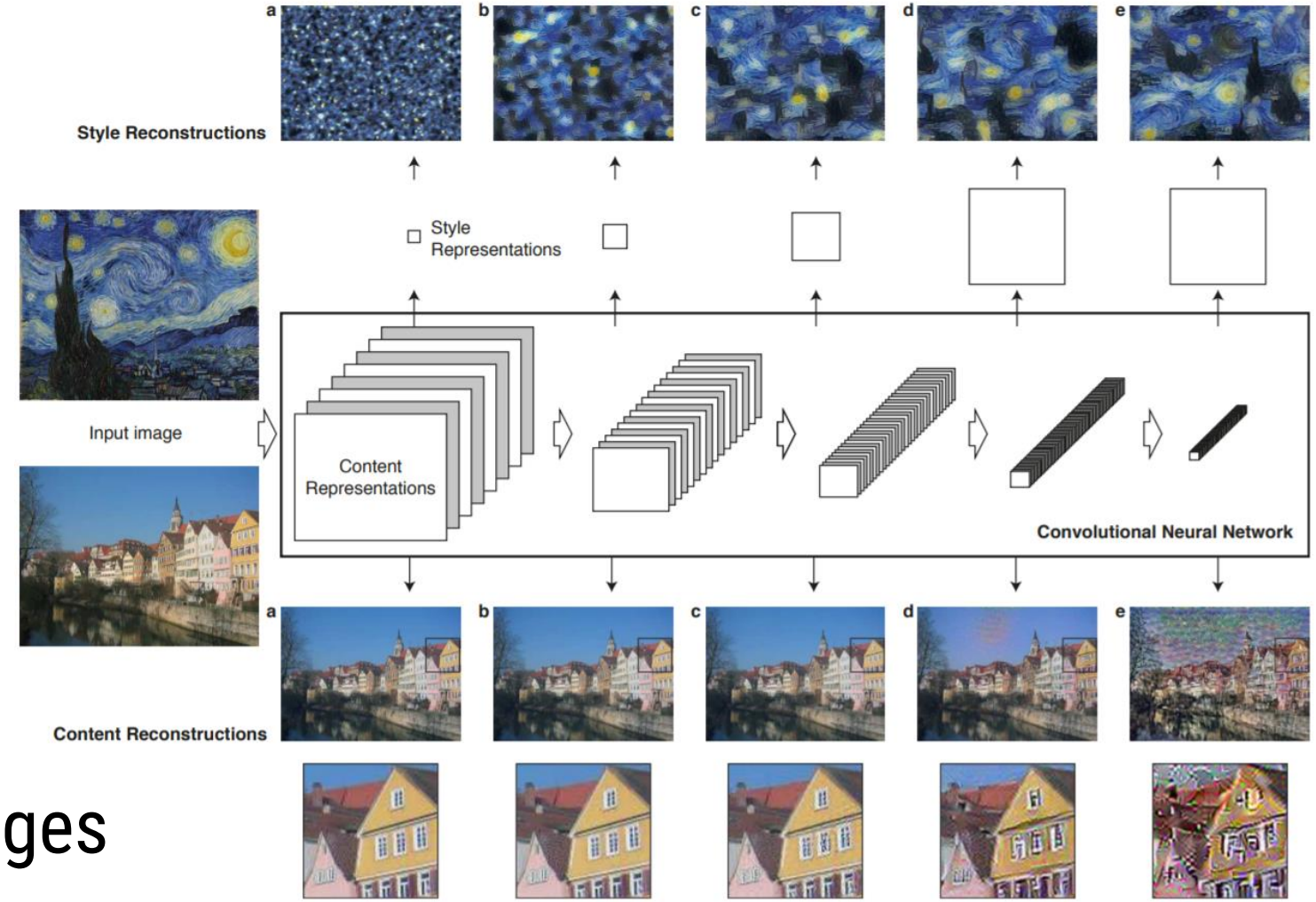
Deep learning for NPR

- neural style transfer: use of neural networks for stylization

- early approaches tried to automatically capture style
- e.g., image analogies (needs matching pairs of input and style images)

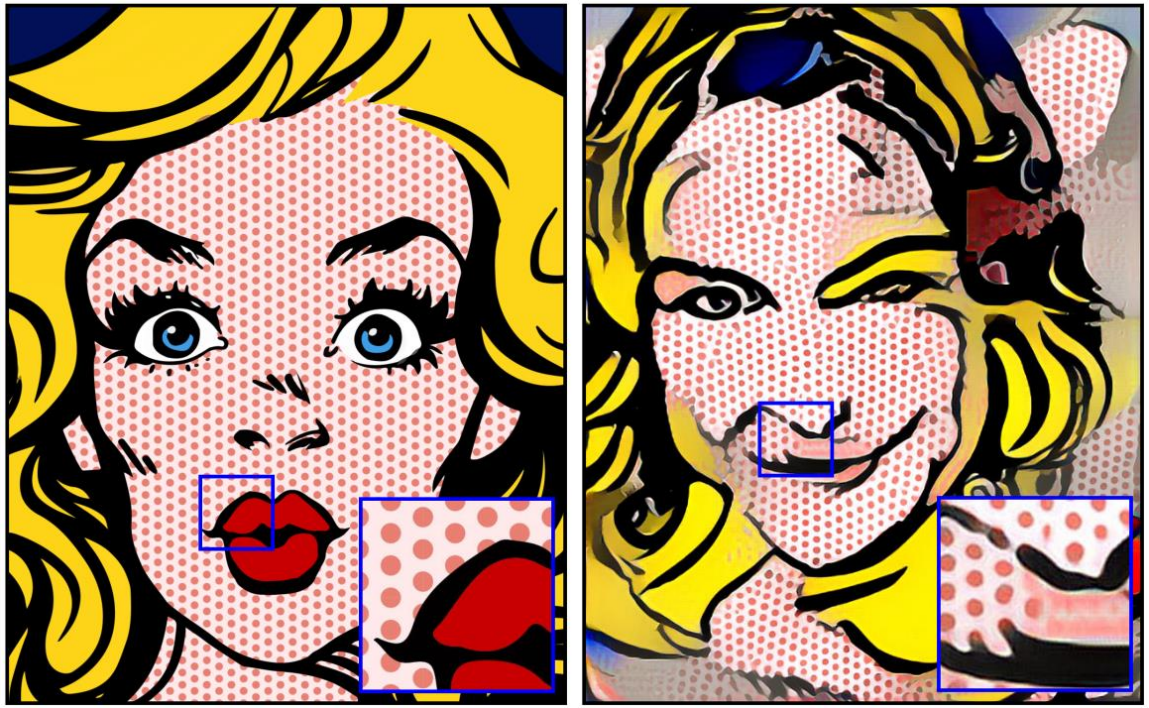
Convolutional neural networks (CNN)

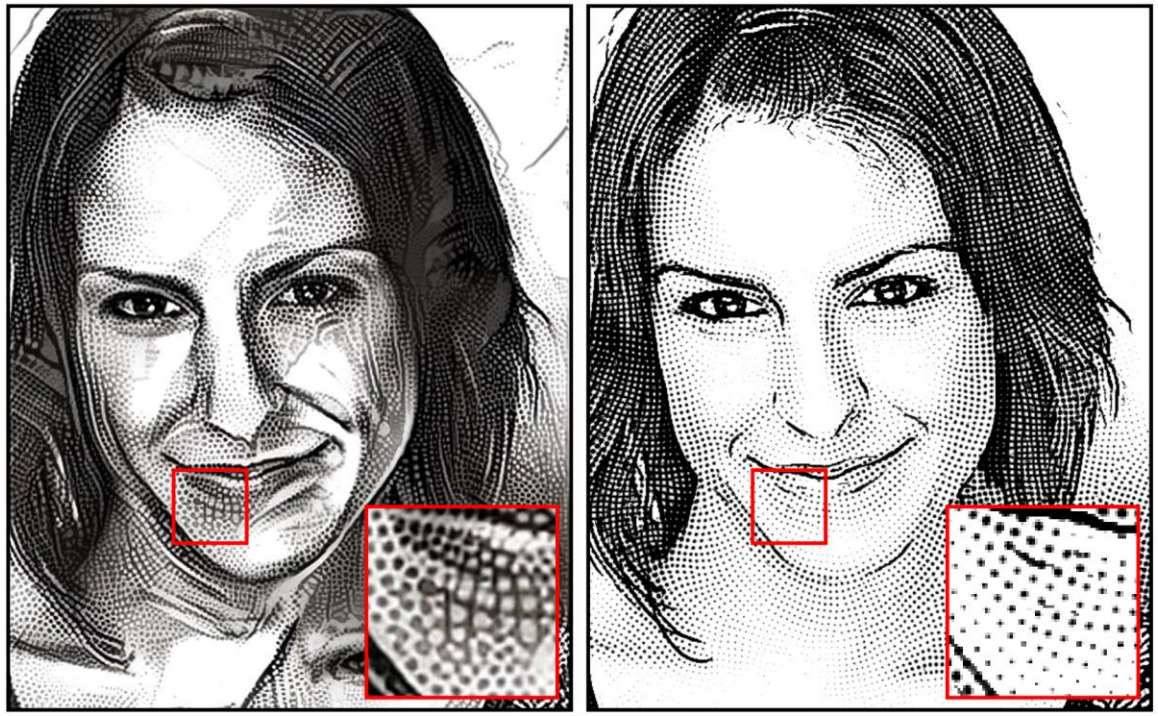
- deep convolutional neural networks can accurately classify high-level image contents
- layers of deep CNNs can be activated to match content and style statistics between arbitrary images



Examples

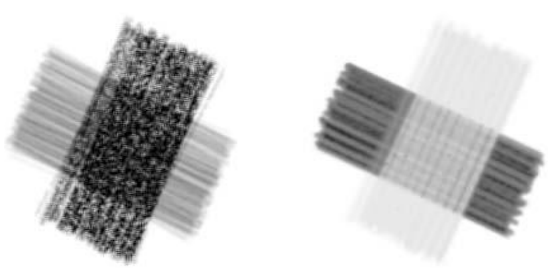
Limitations at meaningful low-level aspects

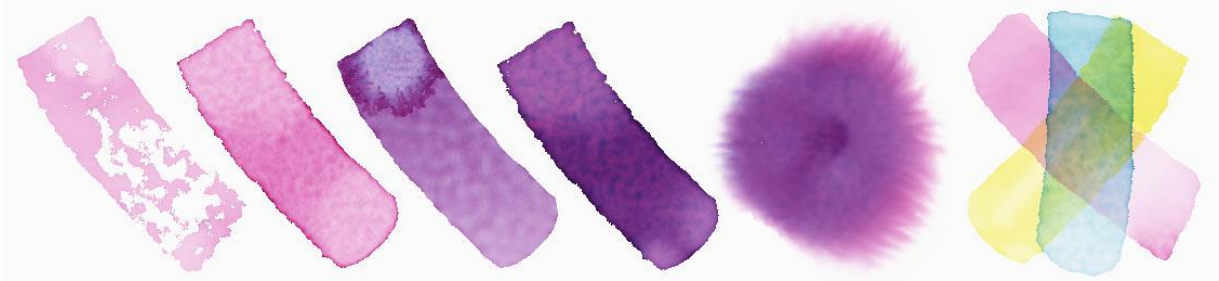




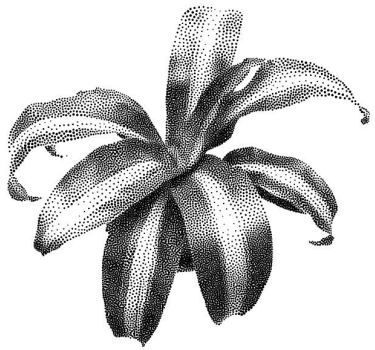
Summary

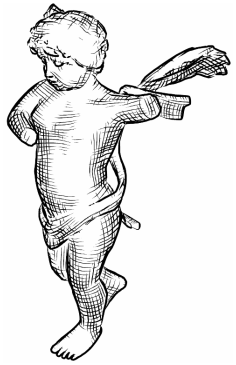
- distinction between media simulation





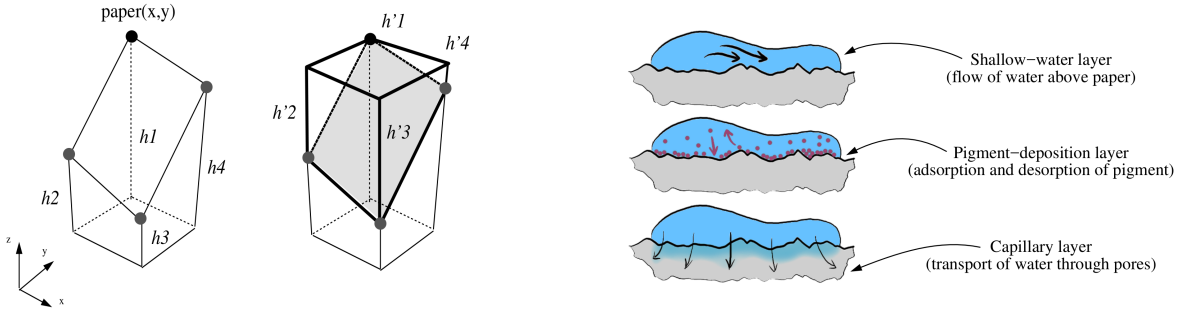
and low-level artistic techniques



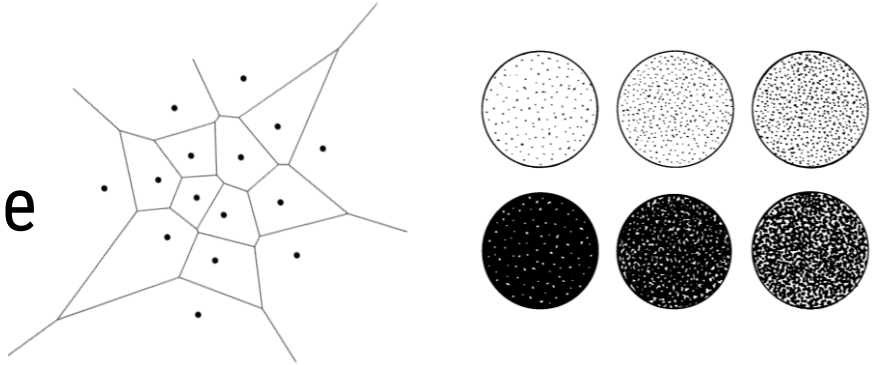


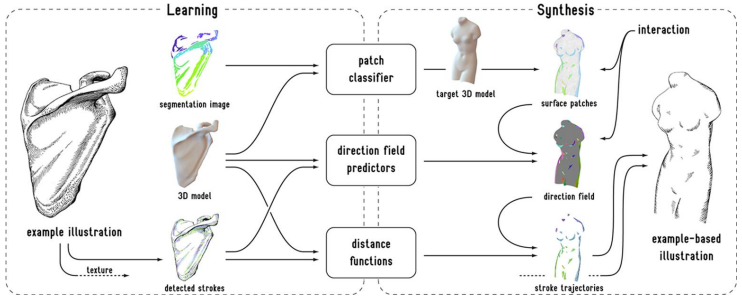
Summary

- media simulation typically via **physically based simulation**



- low-level artistic techniques by
  - (simplified) **modeling** of low-level technique
  - capturing artistic input (**example-based**)

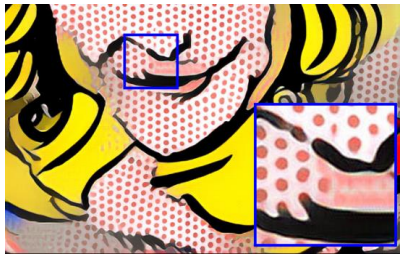




Summary

- deep learning very recent development with a lot of potential

- but also with limitations



Summary

- (parts of) NPR can also be seen as part of the **quest for realism in computer graphics**