Computer Graphics:
An Enabler of Visual Modeling, Drawing, and Communication

劉興民
國立中正大學資訊工程學系

damon@computer.org
http://www.cs.ccu.edu.tw/~damon

國立中興大學資訊科學與工程學系
二〇〇八年三月二十八日

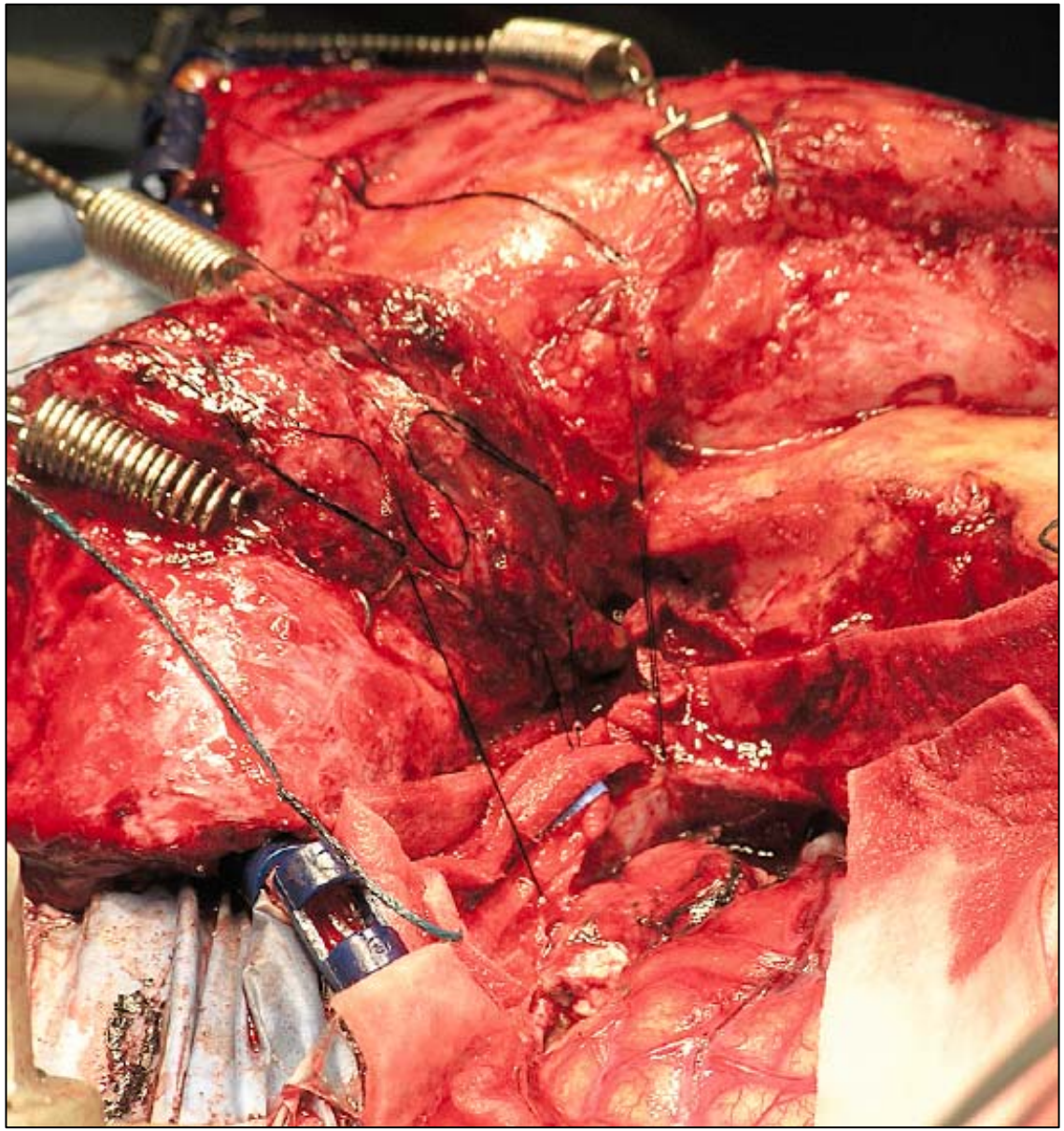
Purpose of Using Computers

- Database – a shift from computation to information: Web, search engines, photo & video libraries, e-Commence.

- As computers become *faster*, *smaller*, *ubiquitous*無所不在, and *interconnected*, their primary function is shifting from computation to communication.

Photo-realistic Images

- Much of the research in computer graphics.
- *Efficiency* and *quality*.
- Not always the best option for representing information.



Scientific Illustrations

- Might or might not be *visually realistic*.
- Conveys傳達 complex structures or procedures in an *easily understandable* way.
- Uses abstraction抽象概念 to prevent visual overload – allows to focus on the *essential* parts.
- Main purpose: communicate information and not necessarily to look "*real*"
  - this makes scientific illustrations differ from photo-realism and other representational genres類型.

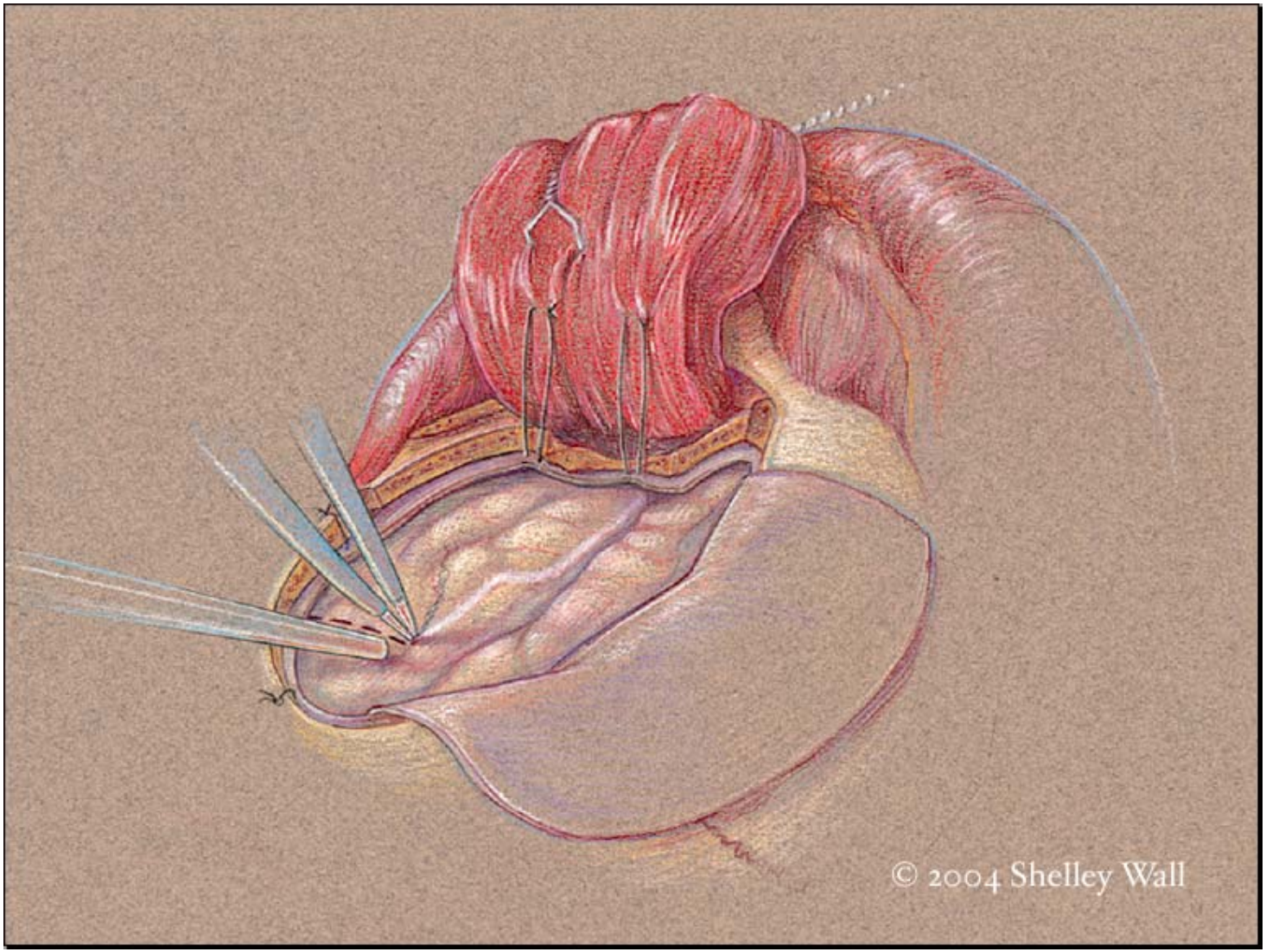
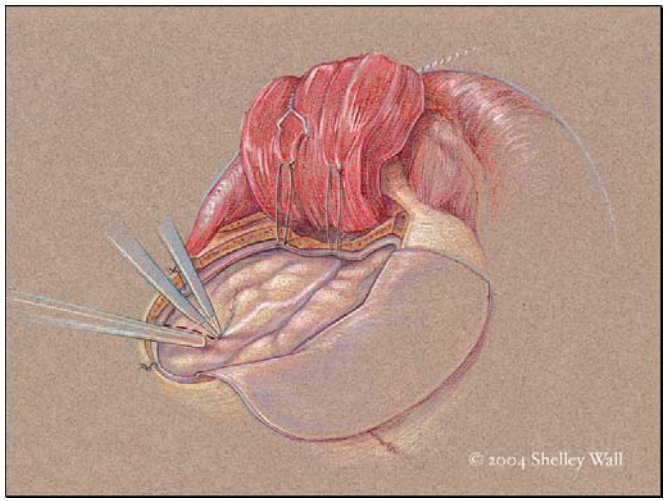
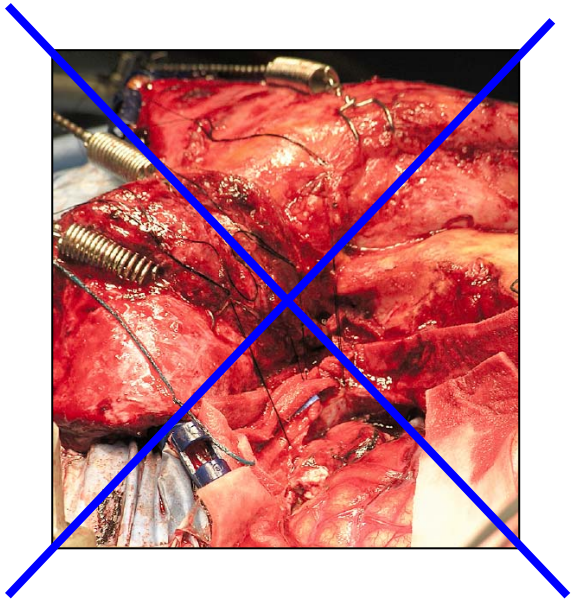

© 2004 Shelley Wall

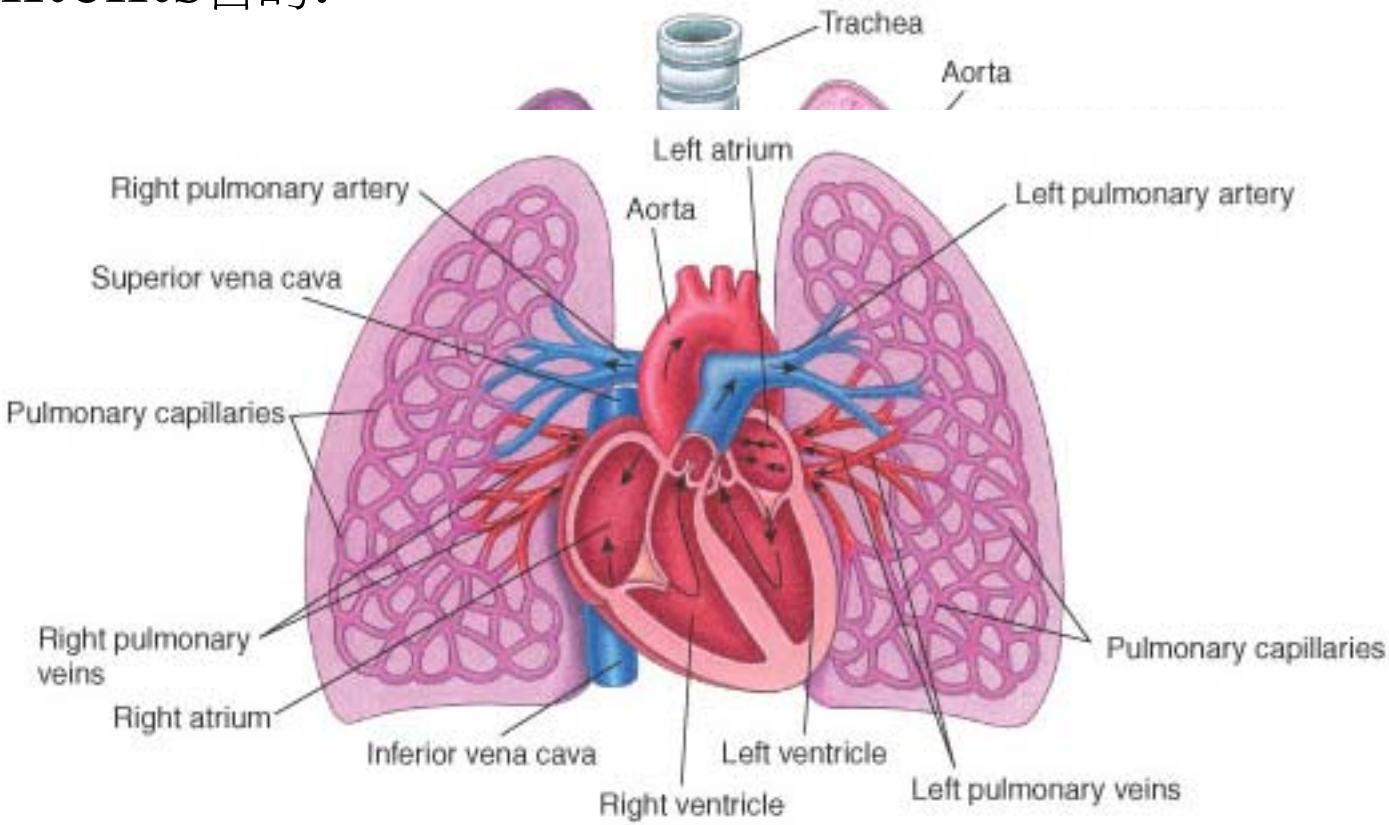
Illustration or Photo … Which?





Abstraction --有點類似情境繪圖

- Different degrees of abstraction for different intents目的.



schematic view of anatomy 構造

Illustrative Graphics Pipeline

Most *illustrative graphics systems* **are either:**

- **Fully interactive**, expecting the user to produce traditional images from scratch (drawing/painting systems) or…
- **Fully automatic**, producing images using automatic techniques (renderers, post-processors).

Illustrative Graphics Pipeline (2)

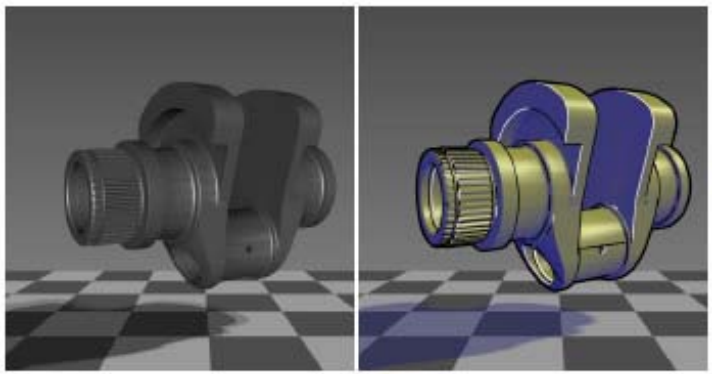
- Trends involve the investigation of hybrid solutions,

  where traditional renderings are produced partly by the system and partly by the user.

Illustrative Graphics Pipeline (3)

- We broke the illustrator's tasks into *six* distinct components:
  - **Modeling** (object construction and representation)
  - **Analysis** (feature extraction, shape measures)
  - **Materials** (media + tools)
  - **Rendering** (stroke marks, tone, texture)
  - **Steps** (rendering progression)
  - **Composition** (principles, effects)

- Non-Photorealistic Rendering (NPR)非擬真繪製 takes inspiration from a long tradition of artistic style and illustrative depiction描繪, such as *sketch*素描, *oil painting*油畫, or *watercolor*水彩畫.



Technical illustration

Game

計算機圖學領域規劃重點議題

一. 科學視算 **(Scientific and Mathematical Visualization)**

二. 計算攝影學 **(Computational Photography)**

三. **3D**模型技術

四. 成像 **(Rendering)** 技術

五. 電腦動畫 **(Computer Animation)**

科學視算(Scientific and Mathematical Visualization)

- 資訊視覺化 (Information Visualization)
- 三維實體資料視覺化 (Volume Visualization)
- 圖解視覺化 (Illustrative Visualization)

計算攝影學
(Computational Photography)

- 紋理生成及變更(Texture Synthesis and Manipulation)
- 影像變更(Image Manipulation)及影像改善(Image Enhancement)
- 高動態範圍影像(High Dynamic Range Images)的擷取(Acquisition)與顯示
- 高解析度影像處理(Efficient Processing for Big Images)
- 大量影像庫之應用(Applications for Large Collection of Images)
- 人類視覺感知(Human Visual Perception)
- 計算相機(Computational Cameras)如Light-Field Camera
- 計算照明(Computational Illumination)
- 三維攝影學(3D Photography)
- 影像操作(Image Manipulation)
- 影像改善(Image Enhancement)
- 高動態範圍影像(High Dynamic Range Imaging)

3D模型技術

- 網格處理與參數化 **(Mesh processing and parameterization)**
- 點基礎建模術 **(Point-based Modeling)**
- 手繪建模術 **(Sketch-based Modeling)**
- **3D** 資訊隱藏
  - 未來發展之三維模型資訊隱藏演算法必須考慮五個研究主題：
  - **(1)** 資訊安全性**(Security)**
  - **(2)** 不可視**(Imperceptibility)**
  - **(3)** 強韌性**(Robustness)**
  - **(4)** 資訊容量**(Capacity)**
  - **(5)** 可逆性**(Reversibility)**

成像 (Rendering) 技術

- 照明模式 (Illumination Models)
- 基於視訊與影像之成像技術 (Video and Image-Based Rendering)
- 圖形處理單元成像與計算(GPU Rendering and Computing)
- 非擬真成像 (Non-Photorealistic Rendering)
- 即時顯像 (Real-Time Rendering)
- 取樣與光機追蹤 (Sampling and Ray Tracing)

電腦動畫(Computer Animation)

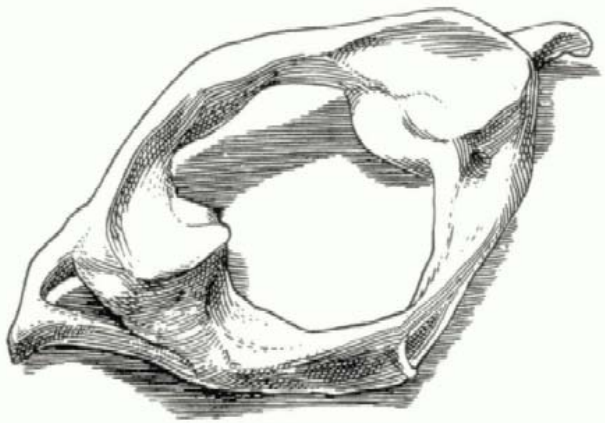
- 三維遊戲設計（**3D Game Design**）
- 虛擬人體動畫
- 基於物理學的電腦動畫
- 動作及表情的擷取、分析、與合成

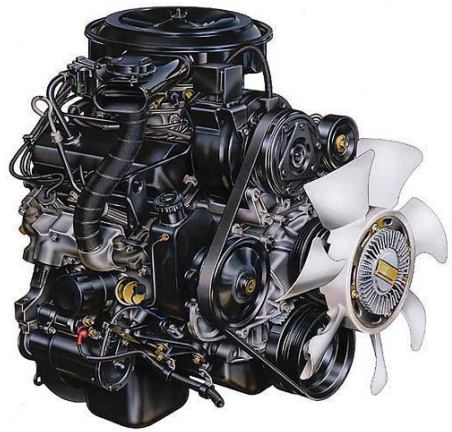
Visual Cues暗示

- **Two fundamental visual cues:**
  - **Silhouette – the visible edges (lines) of a surface.**
  - **Hatching加上影線以爲陰影 – the use of texture to indicate the local *orientation* (shading) of a surface.**

Low-Level Abstraction Techniques

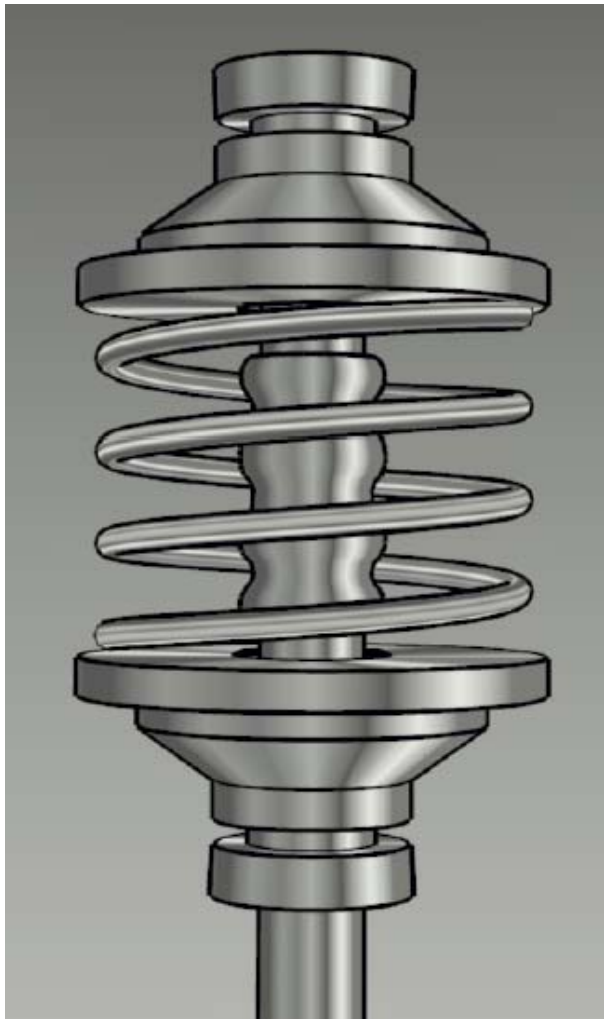
- Concerned with how different objects are presented.
- Stylized depiction
  - Silhouettes and contours, pen and ink, stippling點刻, hatching, ...

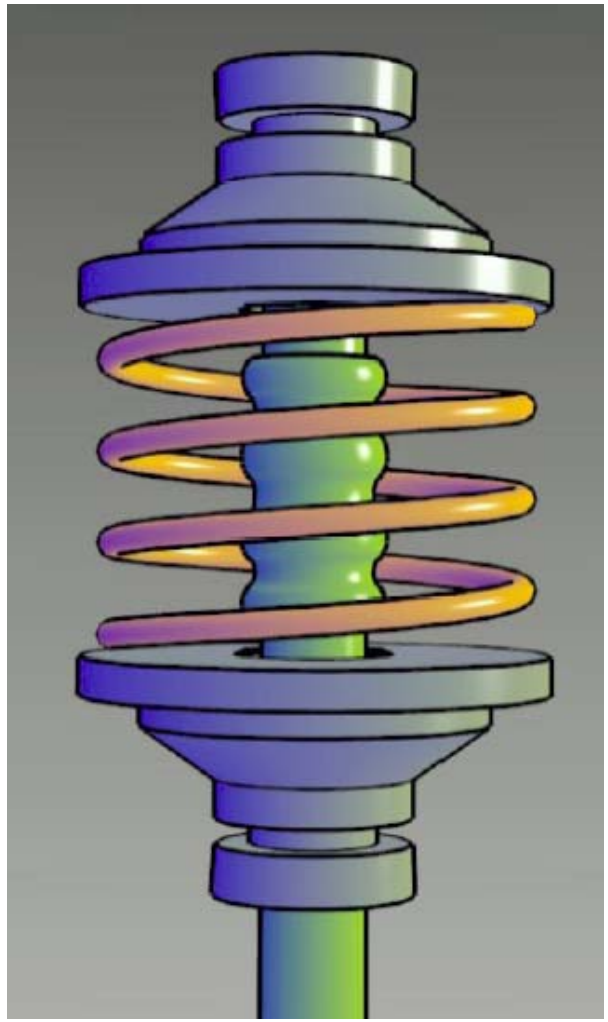





(2)

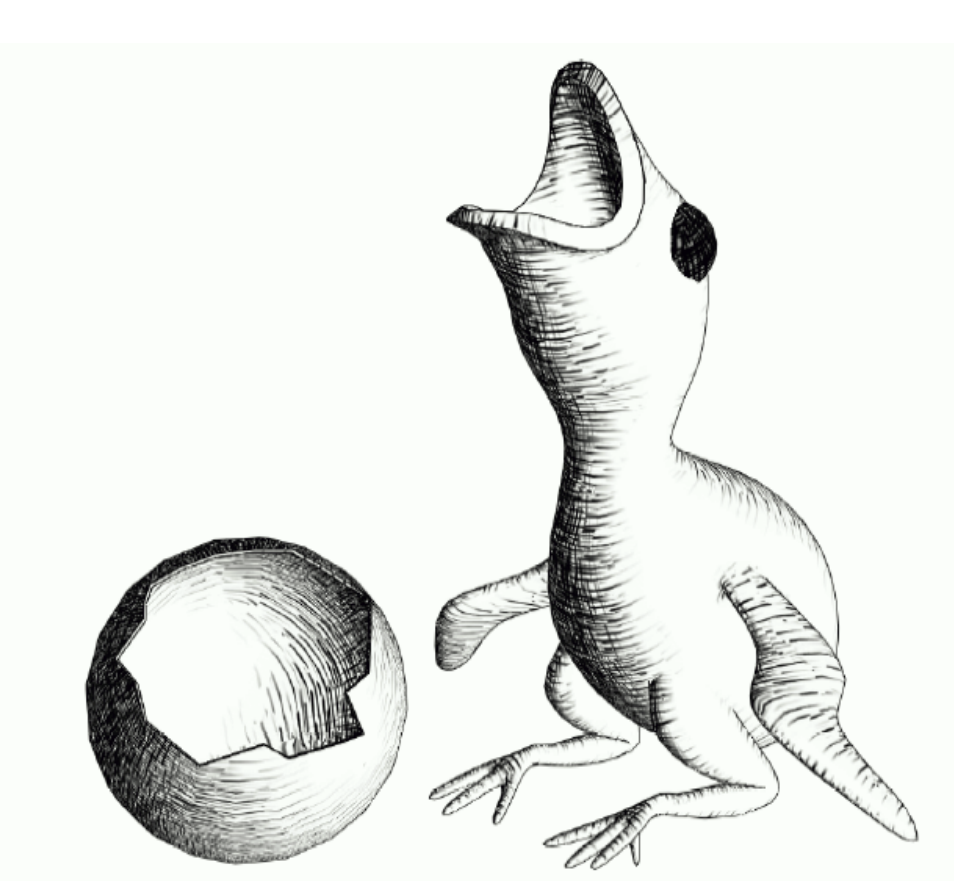
- Metal and tone色調 shading [Gooch et al. 1998]

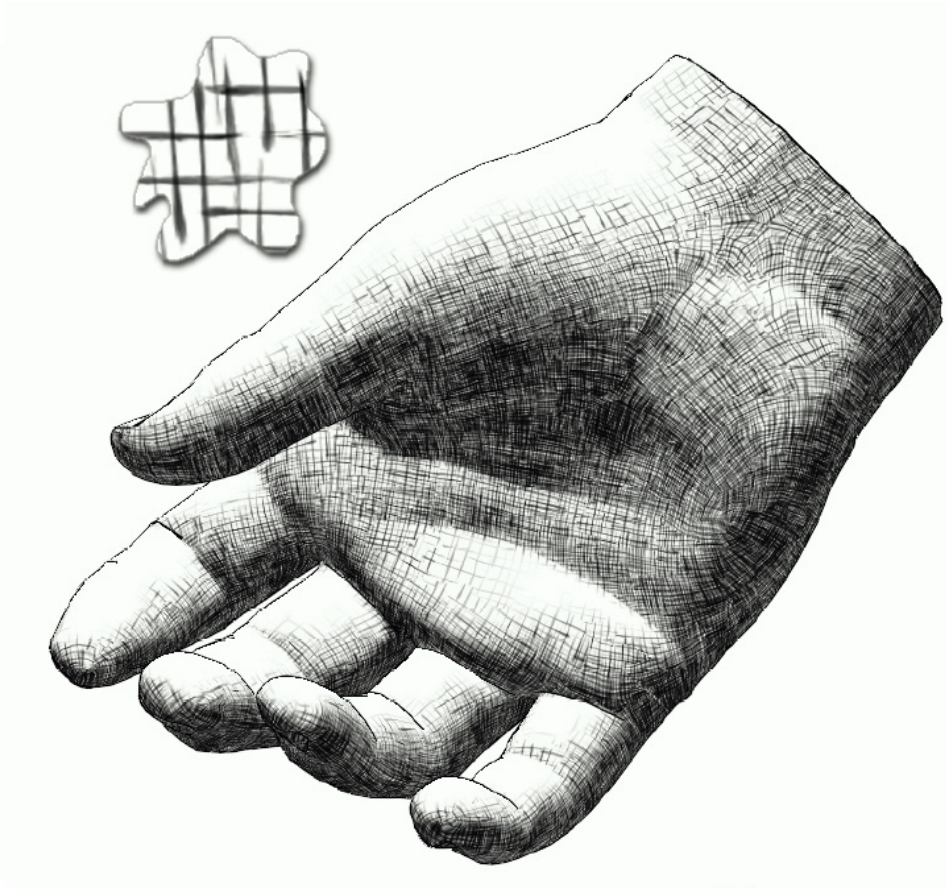




(3)

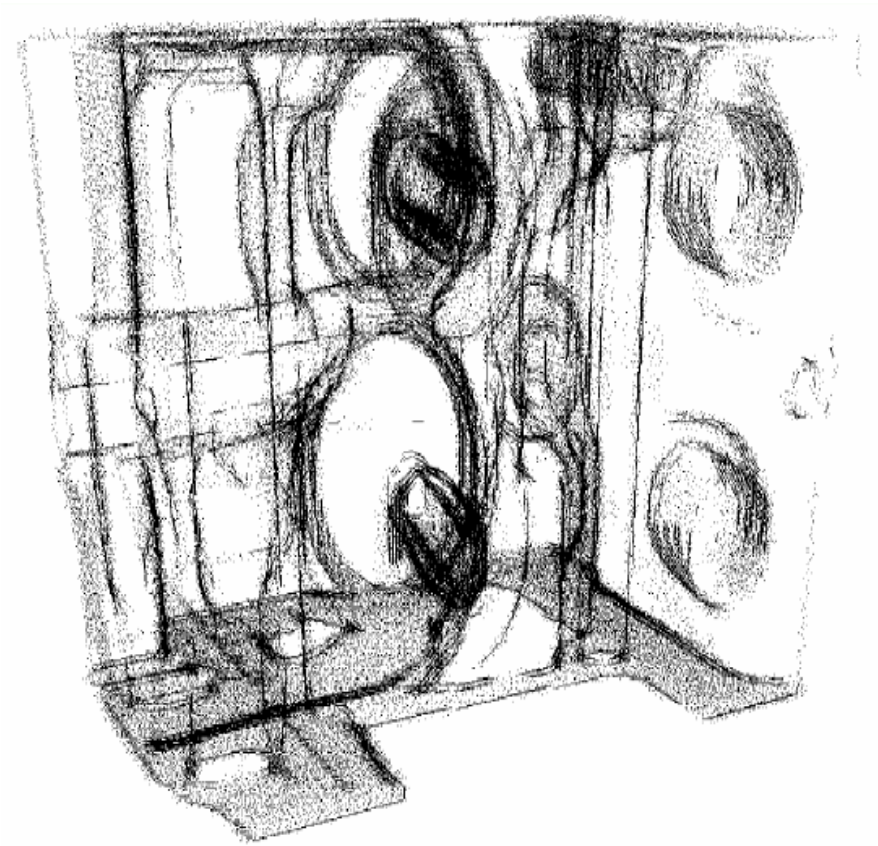
- Real-time hatching [Hoppe et al. 2001]

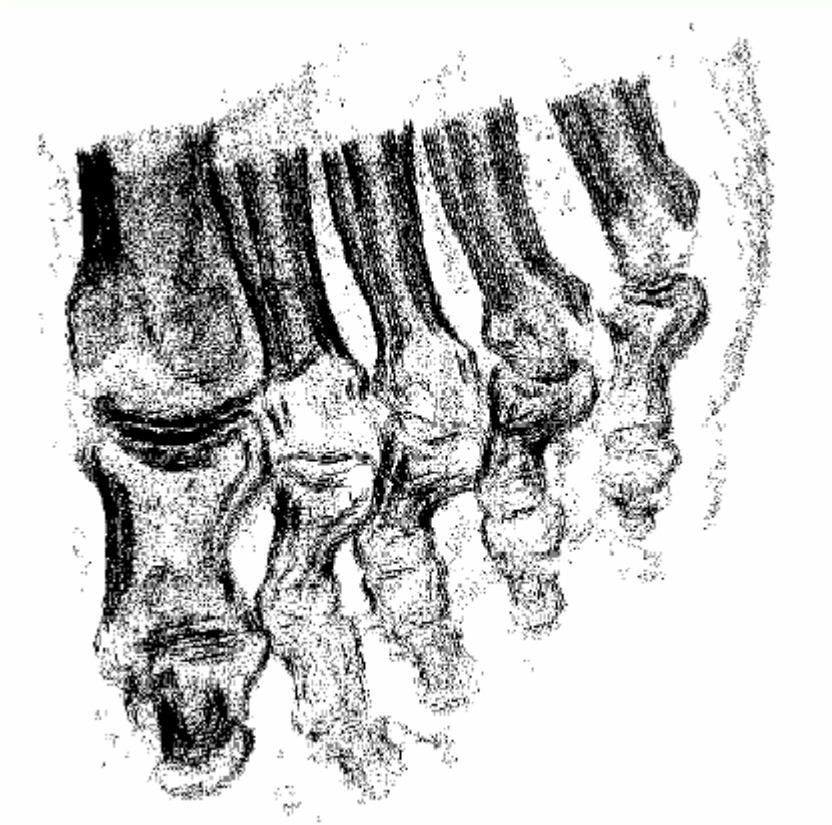




(4)

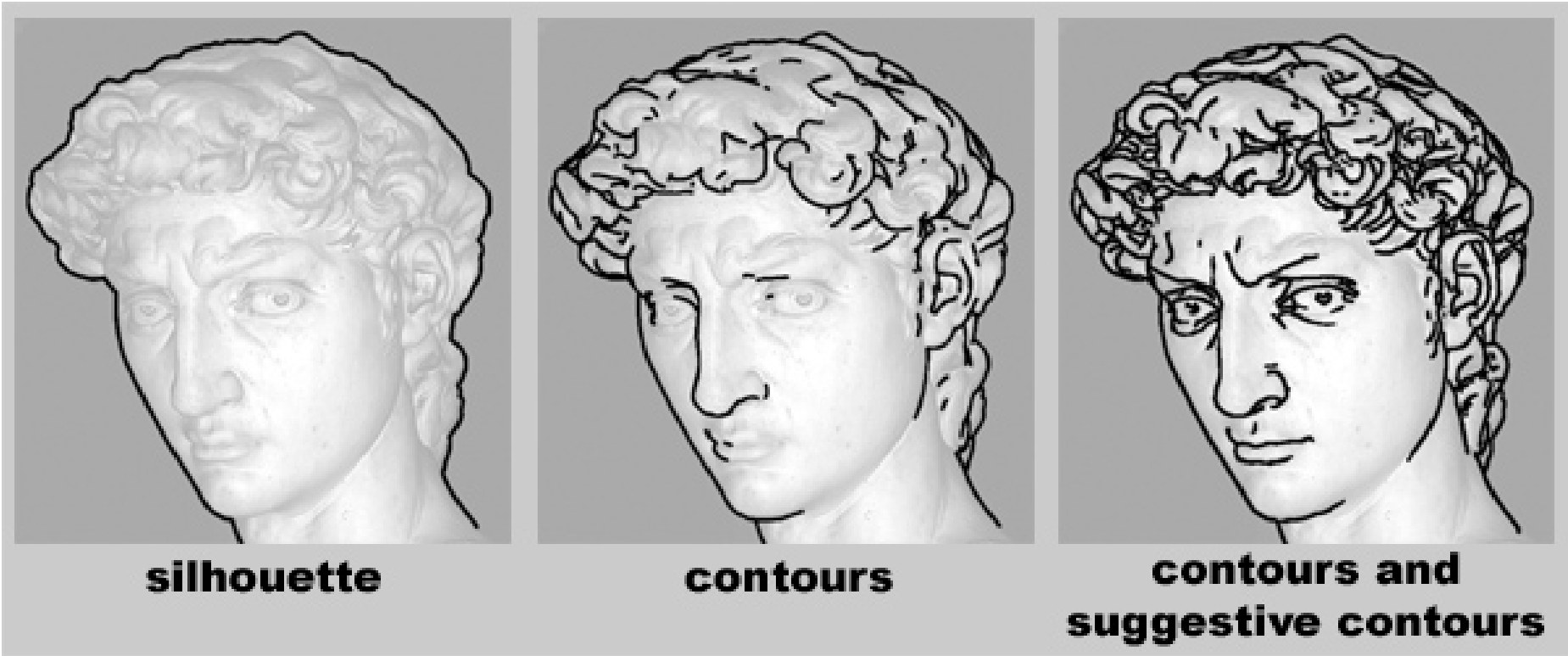
- Volume stippling [Lu et al. 2002]





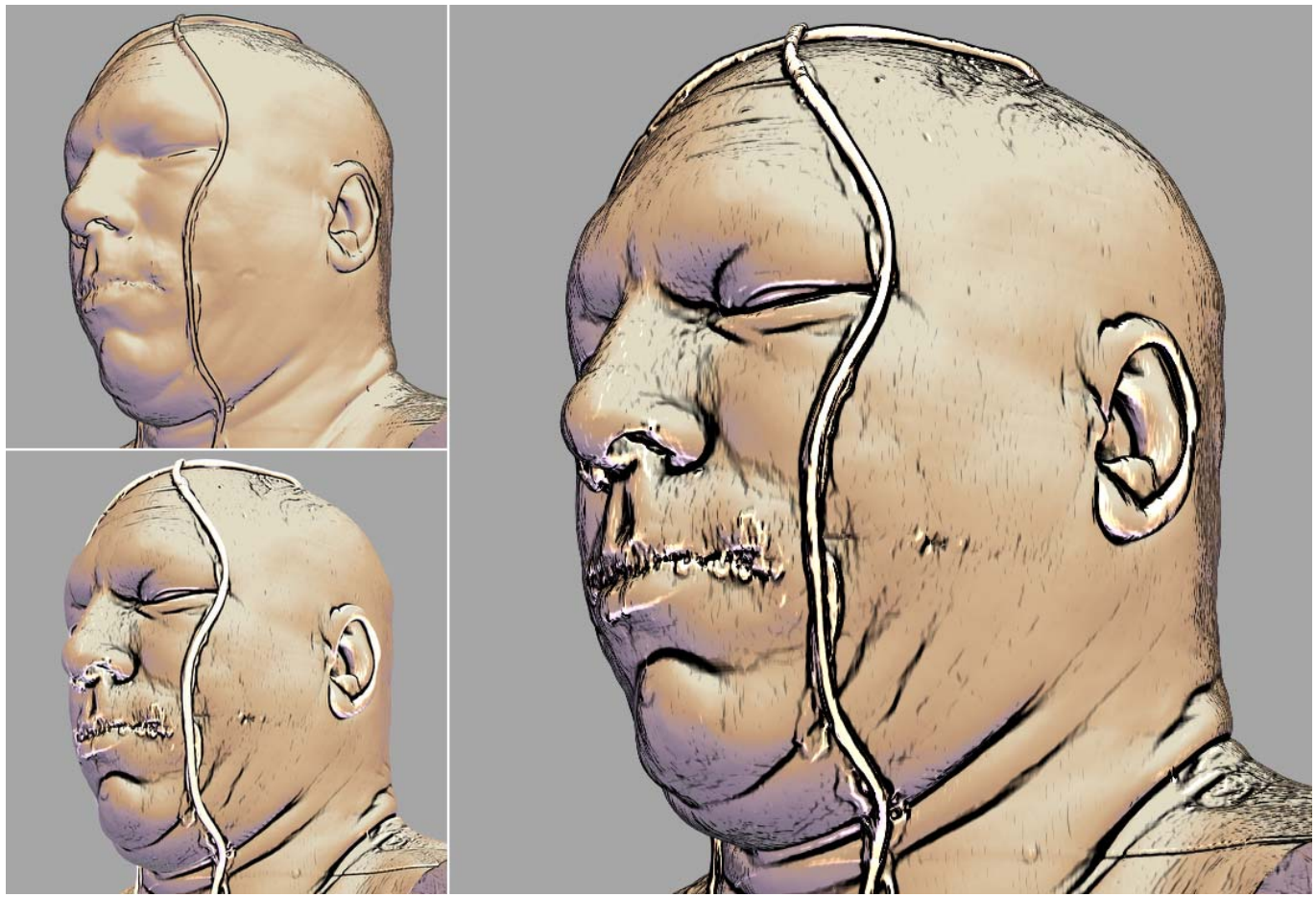
(5)

- Suggestive contours [DeCarlo et al. 2003]



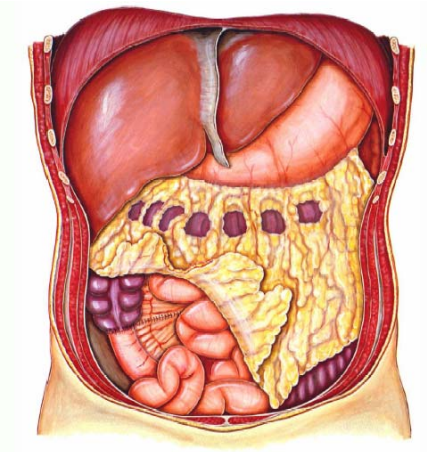
(6)

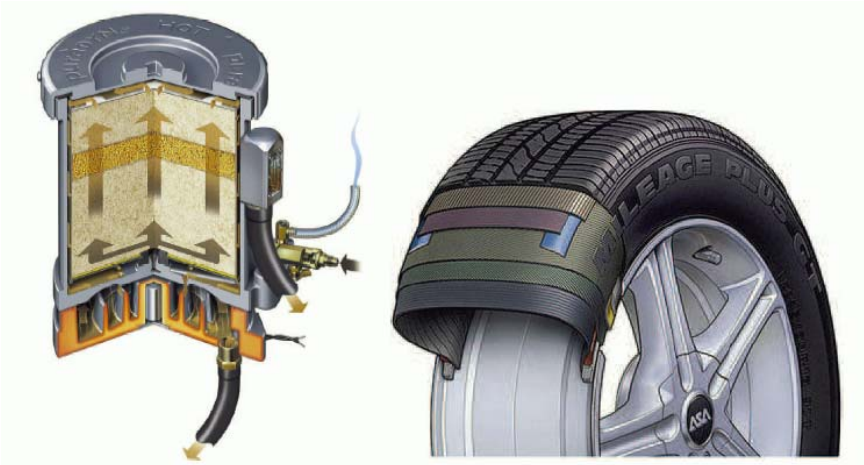
- *Curvature*-based ridge山脊 and valley enhancement [Kindlmann et al. 2003]

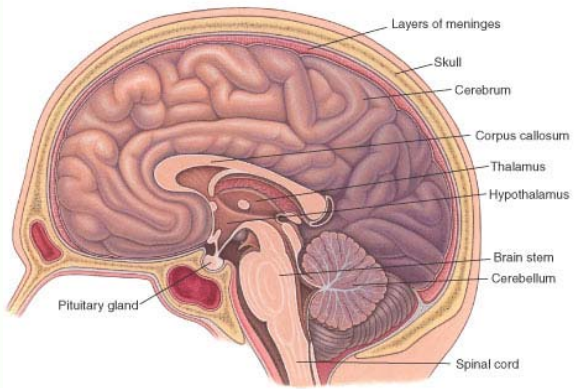


High-Level Abstraction Techniques

- Deal with what should be *visible* and *recognizeable*.
- Smart visibility
  - Cutaways, ghosting, exploded爆炸 views, ...

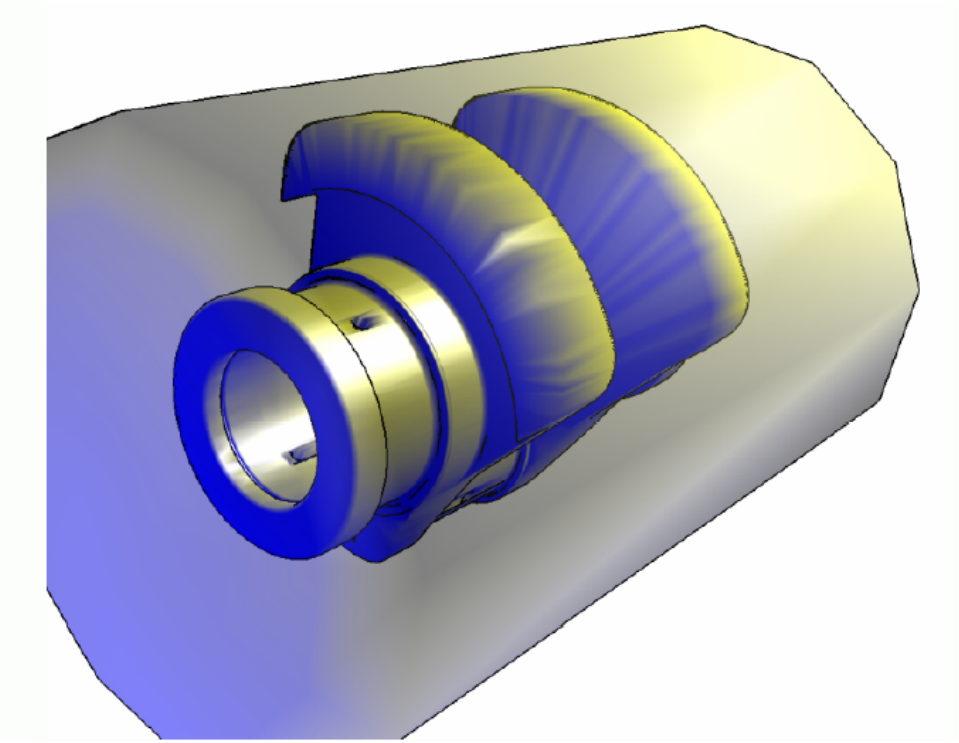


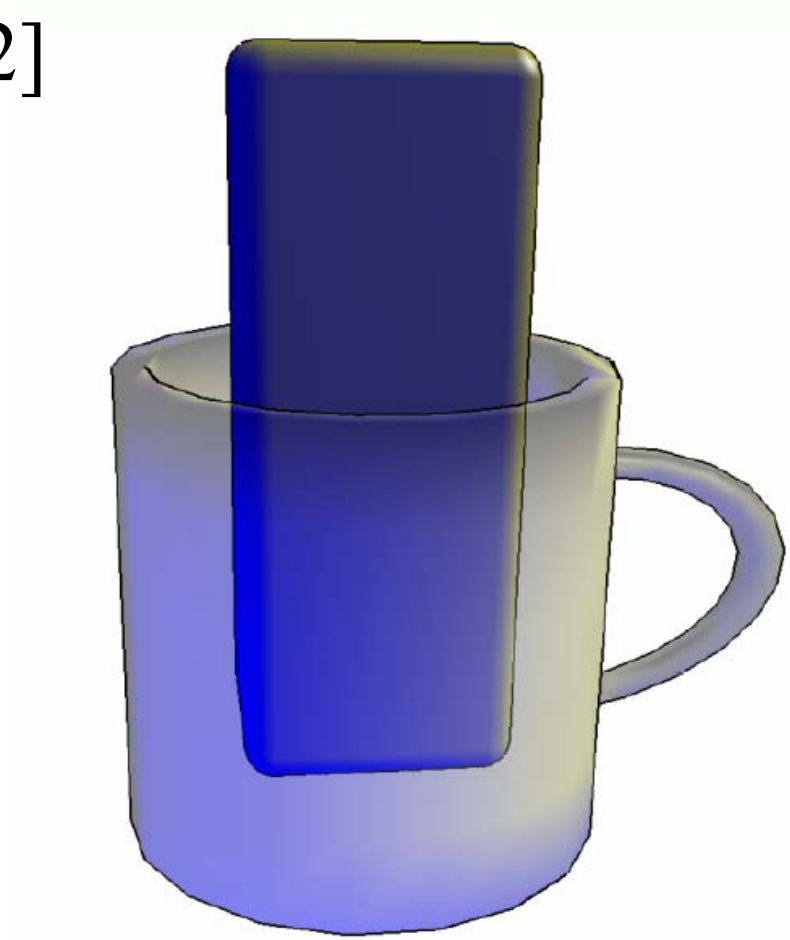




(2)

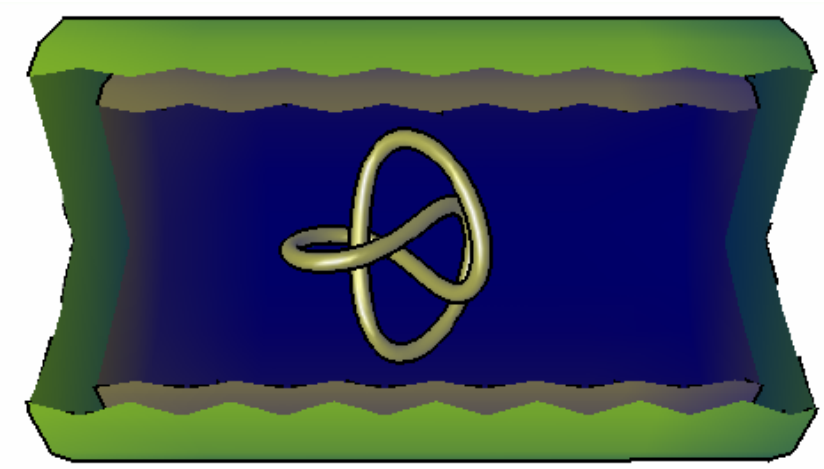
- View-dependent transparency
  [Diepstraten et al. 2002]

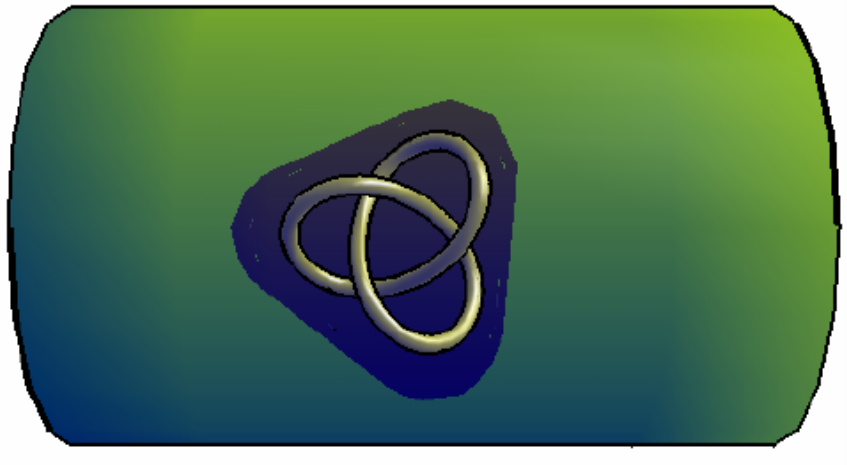




(3)

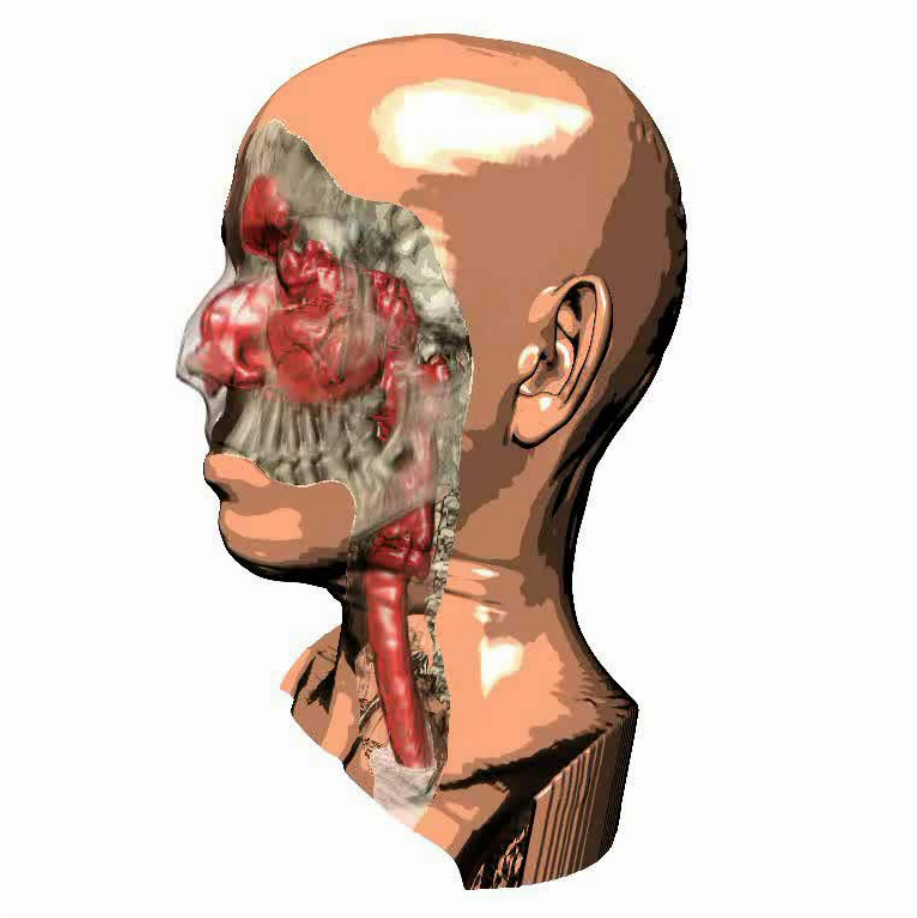
- Cutaways [Diepstraten et al. 2003]





(4)

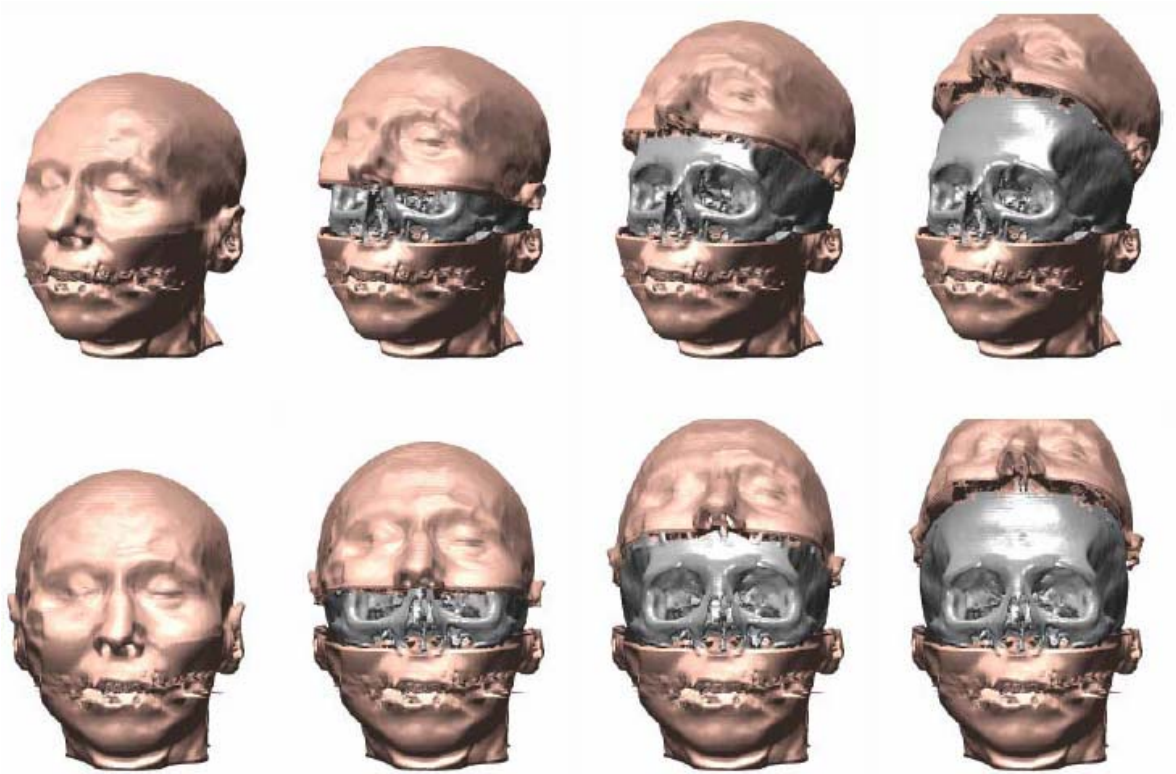
- *Importance*-driven volume rendering [Viola et al. 2004]

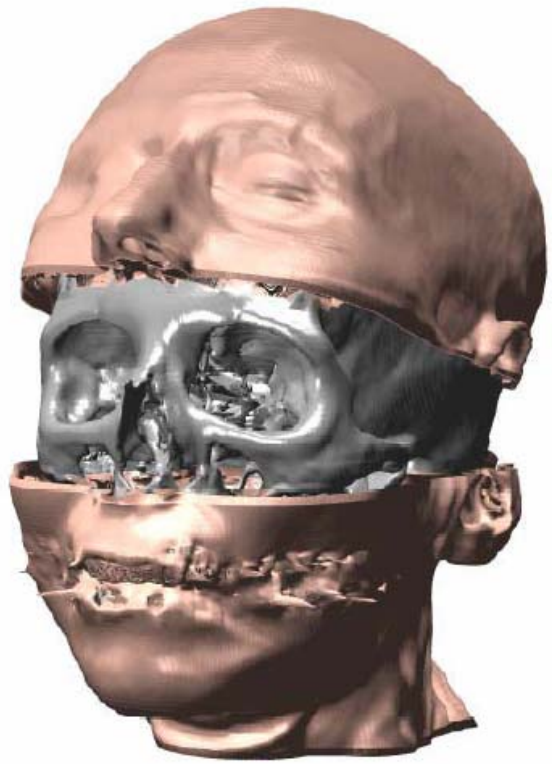


Click for animation

(5)

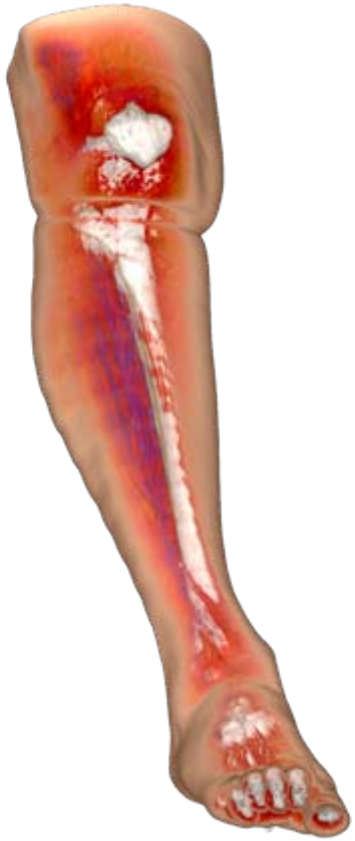
- Volume splitting劈開 [Islam et al. 2004]

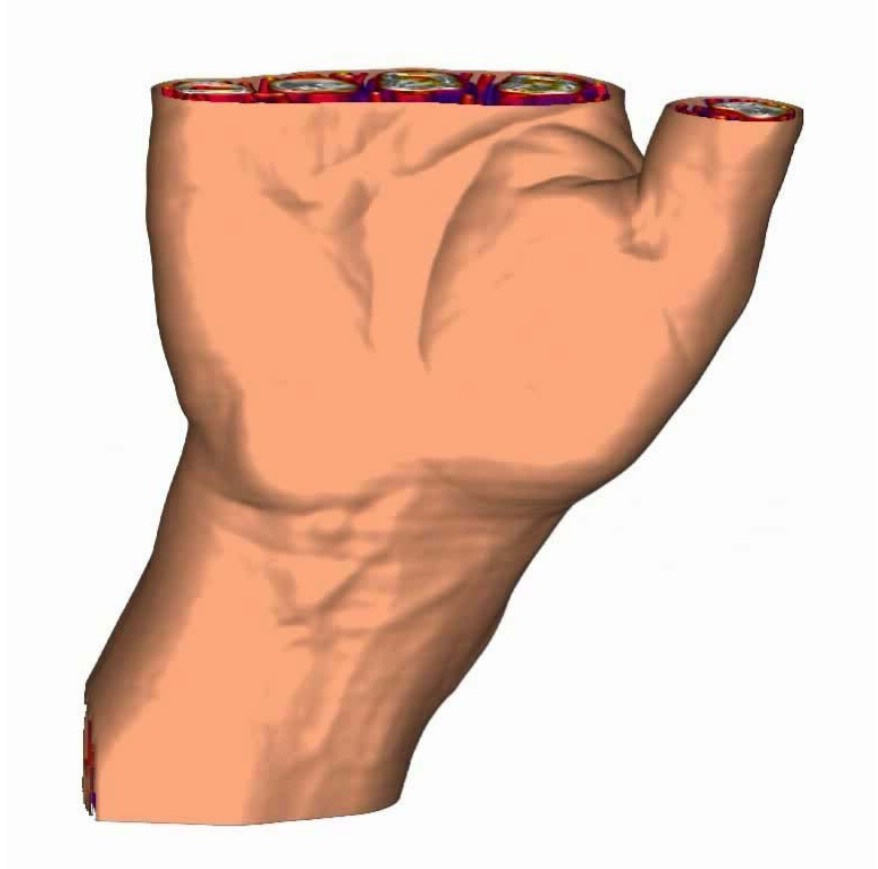




(6)

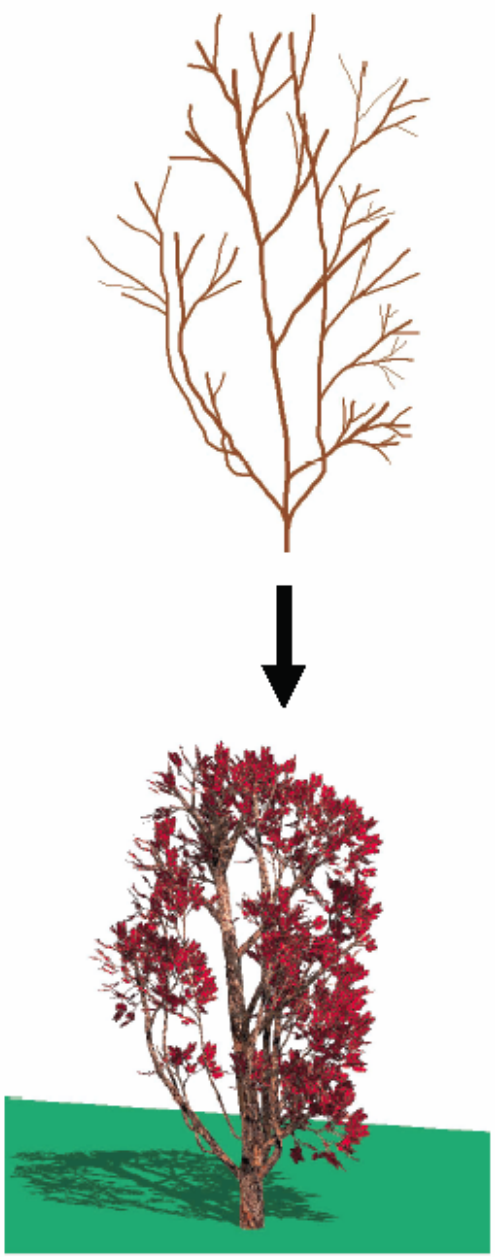
- *Context*-preserving volume rendering [Bruckner et al. 2005]



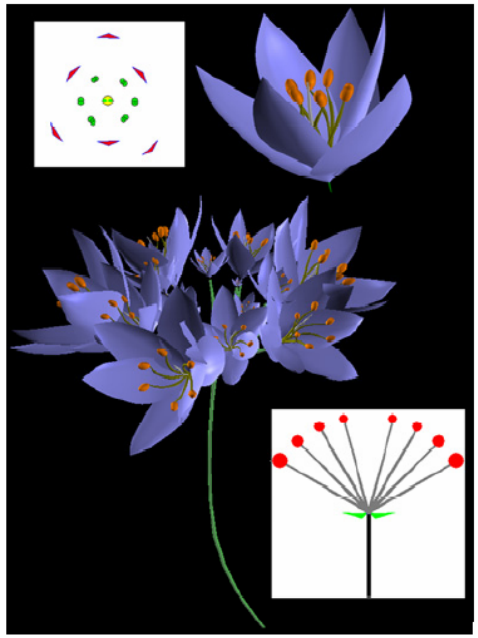


Click to animate

Botanical 植物的 Illustration

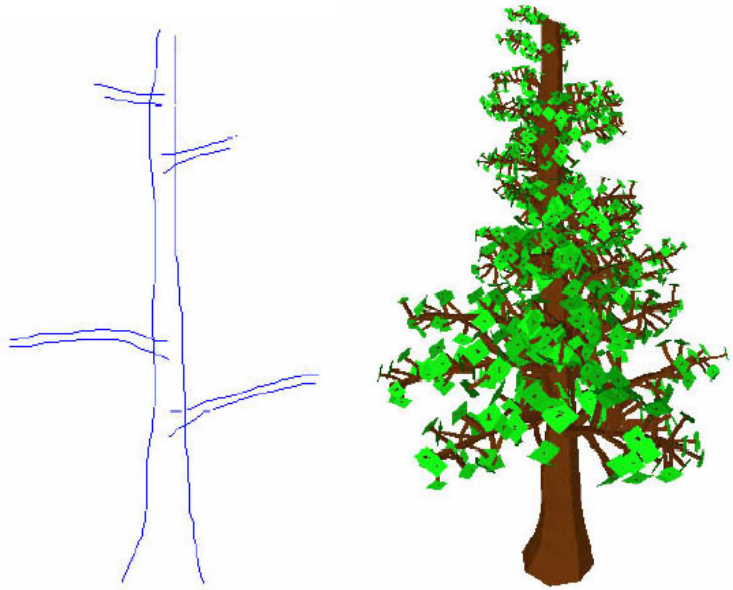


[Okabe et al 2005]



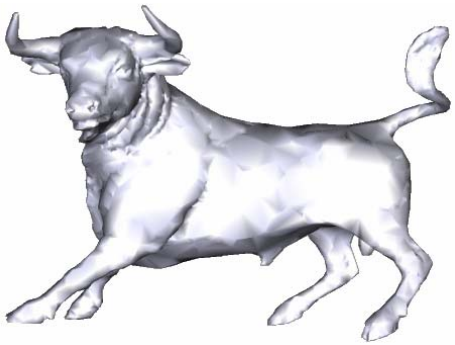
[Ijiri et al 2005]

[Ijiri et al 2004]

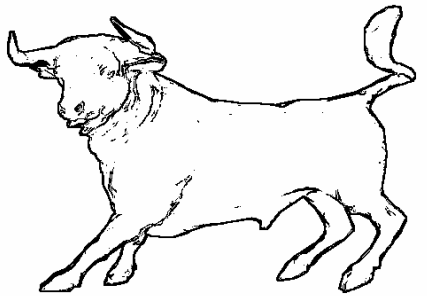


[Okabe and Igarashi 2003]

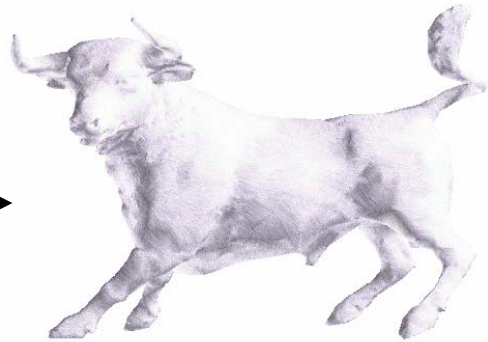
Technique Overview



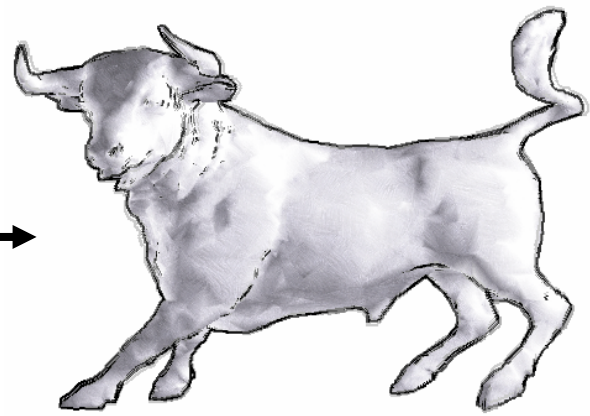
Input model



Contour drawing

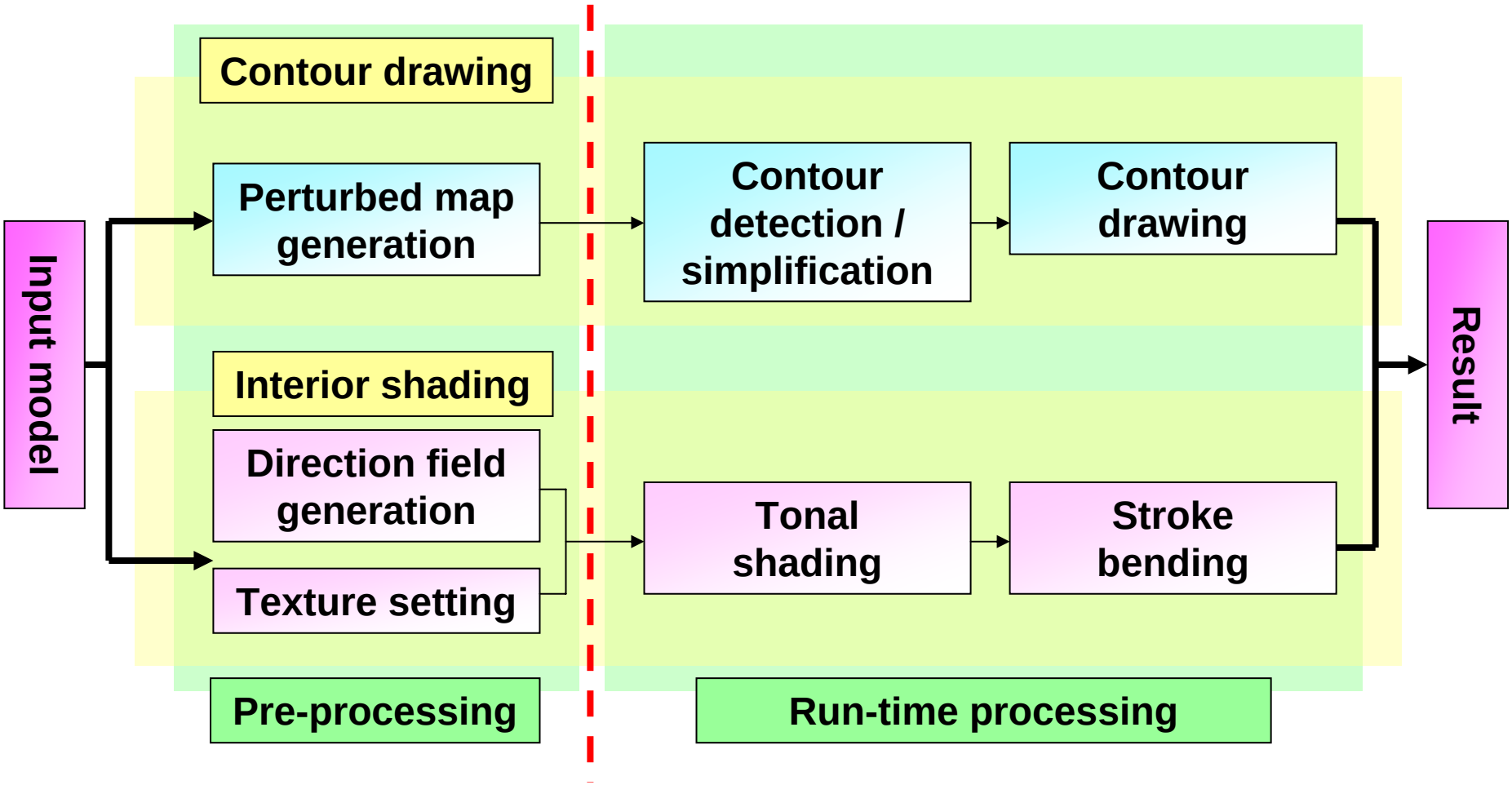


Interior shading

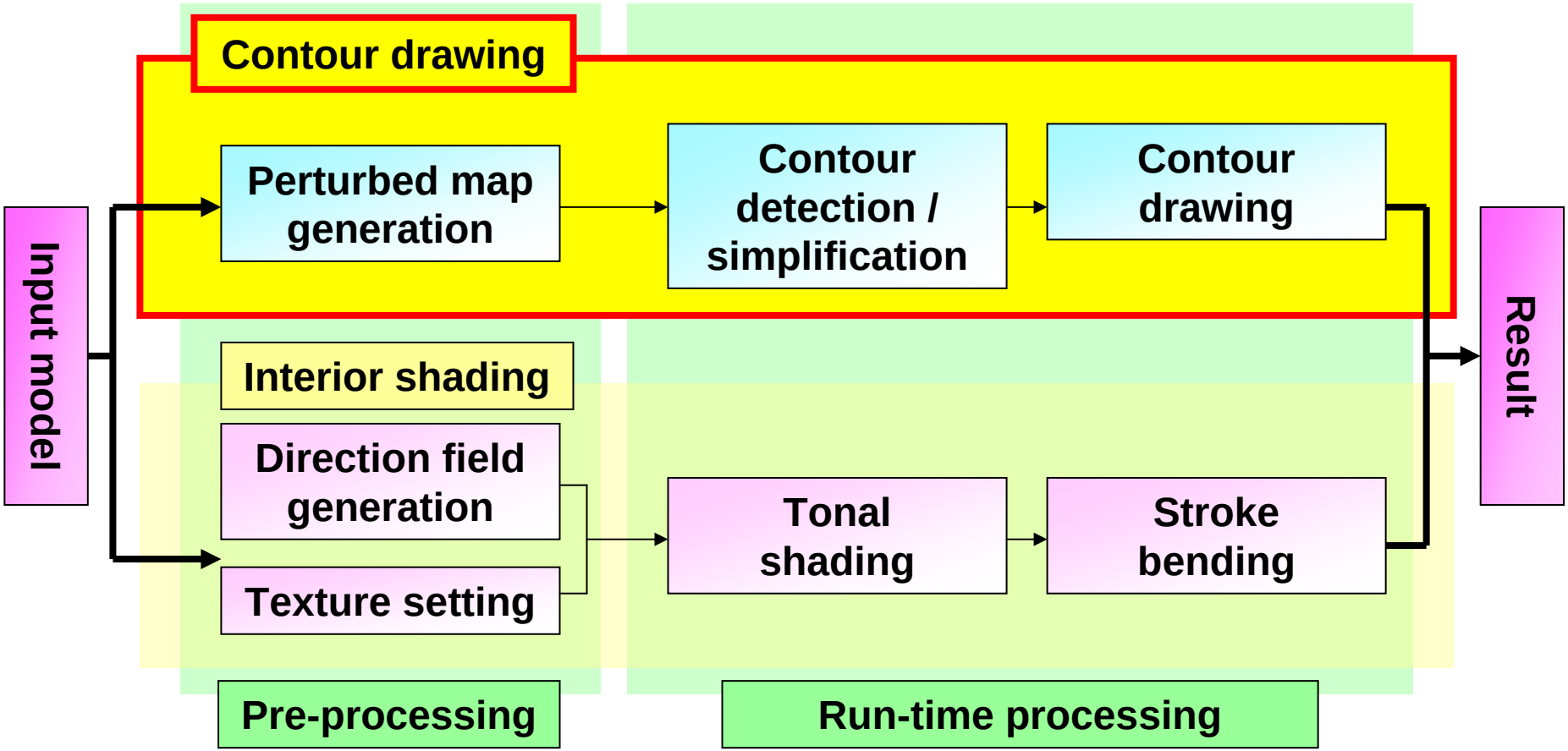


Final result

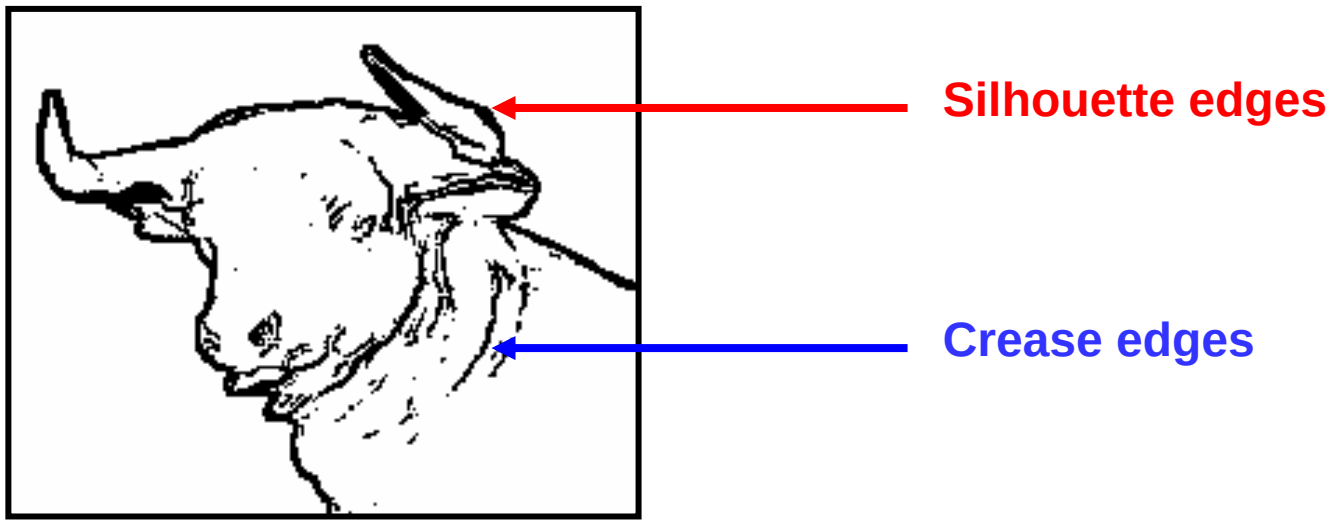
Work Flow


Contour drawing
Perturbed map generation
Contour detection / simplification
Contour drawing
Input model
Result
Interior shading
Direction field generation
Tonal shading
Stroke bending
Texture setting
Pre-processing
Run-time processing

Contour Drawing

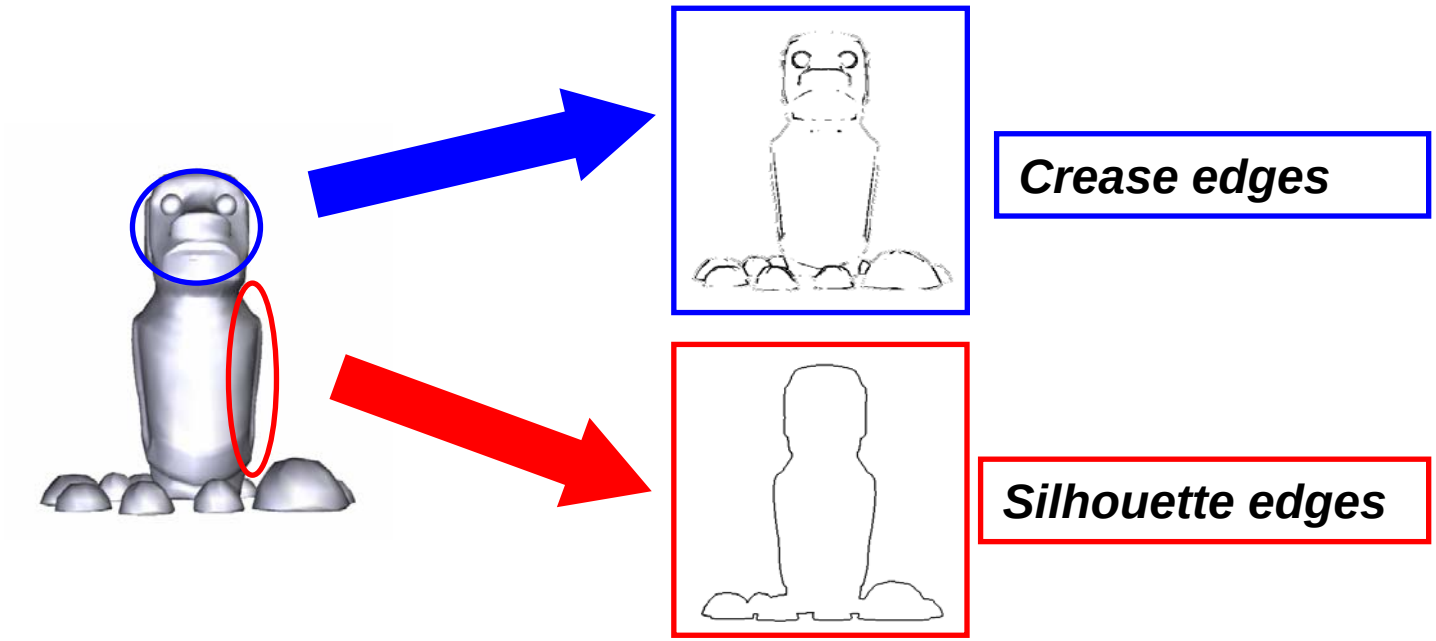


Edges in a Model



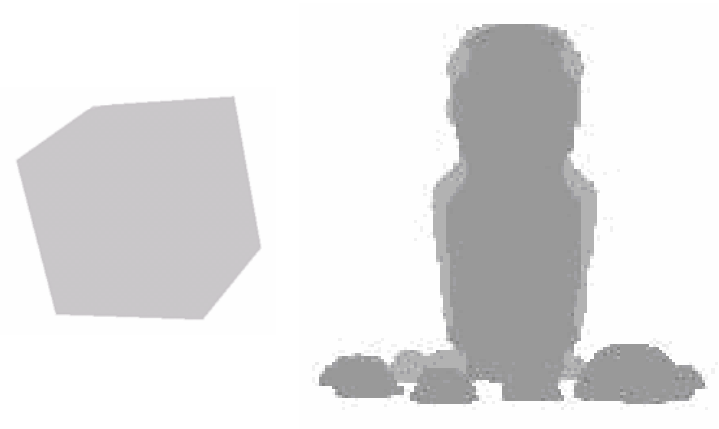
- **Silhouette edge:**
  - **An edge adjacent to a polygon facing towards the camera (*front*-facing) and one polygon facing in the opposite direction (*back*-facing).**
- **Crease edge:**
  - **An edge between two front-facing (or back-facing, respectively) polygons whose *dihedral angle* is above some threshold.**

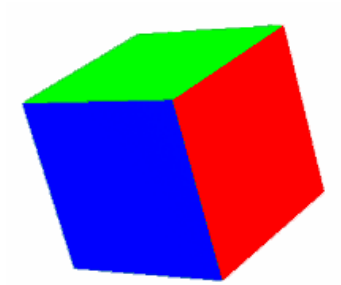
    The dihedral angle defines the *intensity* of a crease edge.

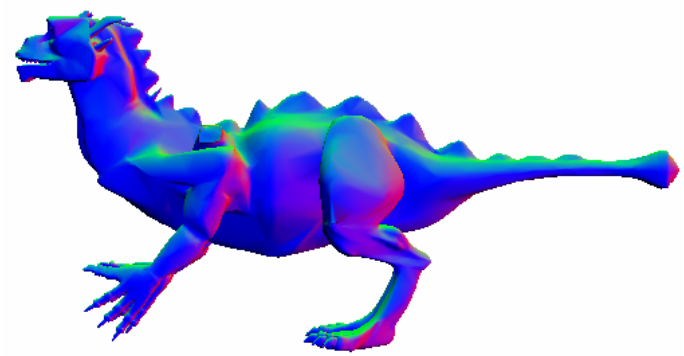


Edge Detection in *Image-Space*

- **Depth map**
  - Represent depth variation using *grayscale* color.
- **Normal map**
  - Represent curvature variation of surface using *RGB* color.
- Use a simple *Sobel* mask to filter the edges.

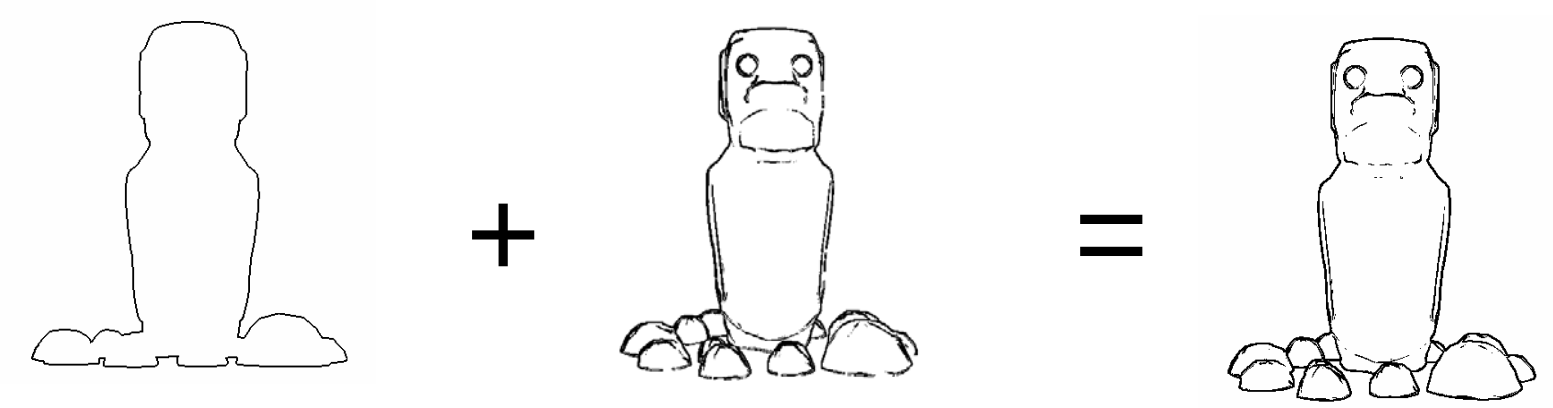






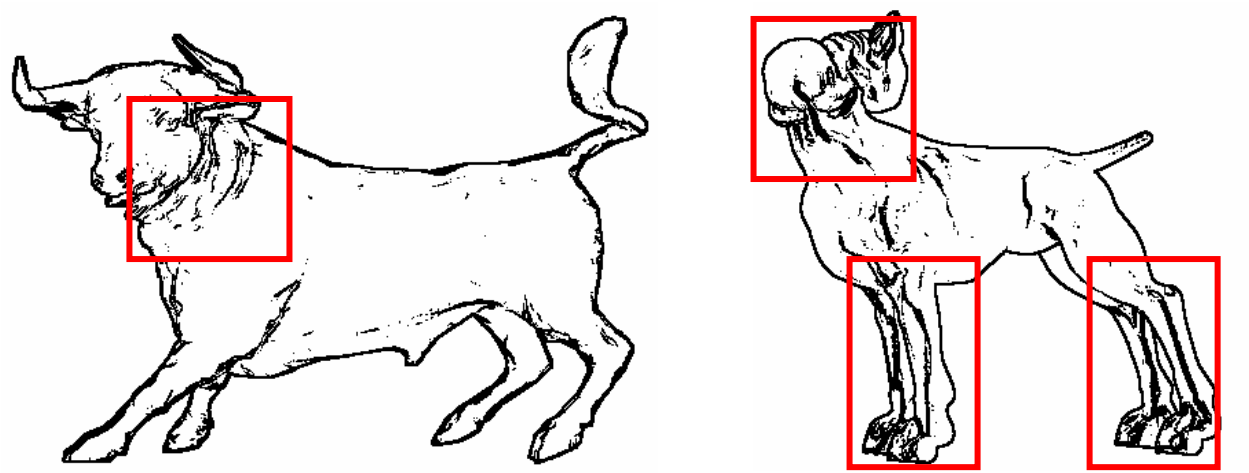
- Combining both crease edges and silhouette edges as a complete edge (contour) map.

Silhouette edges **Crease edges** **Complete edges**



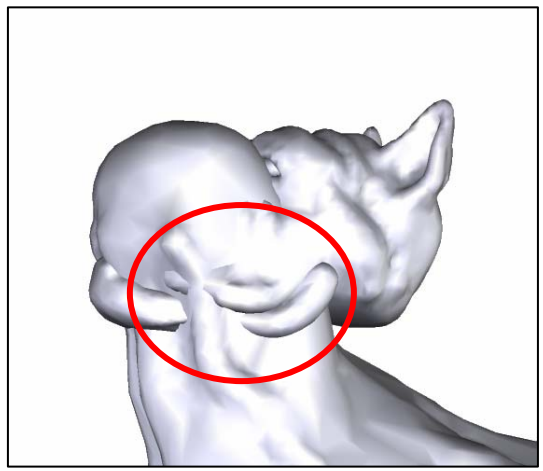
Results of Edge Detection

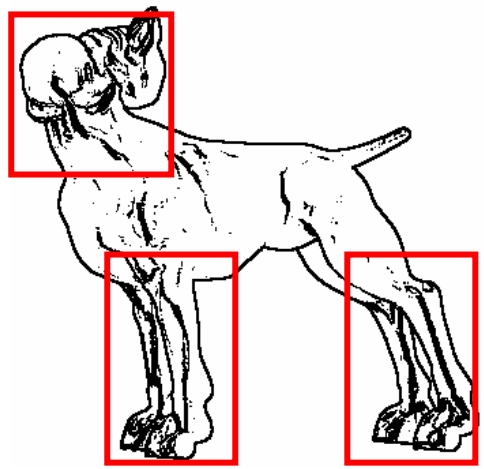
- Depth and normal map detect the edges expeditiously迅速的.
- However, complete edge map may include *too many unnecessary* redundant edges.



Edge Simplification

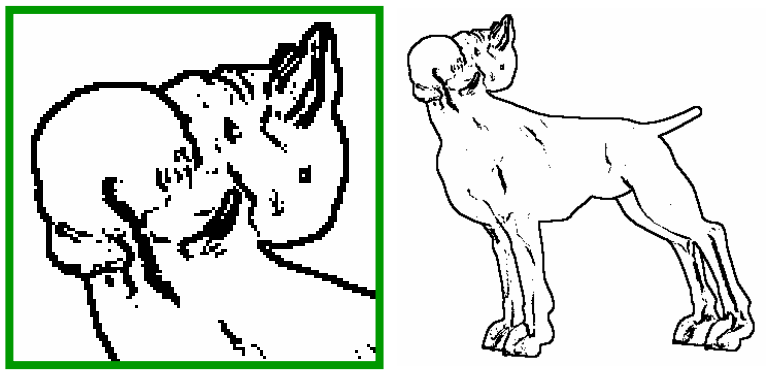
- **Redundant edges usually occur at *abrupt* changes in a small region.**
- **Most of them are crease edges, since they often occur in arc surfaces.**



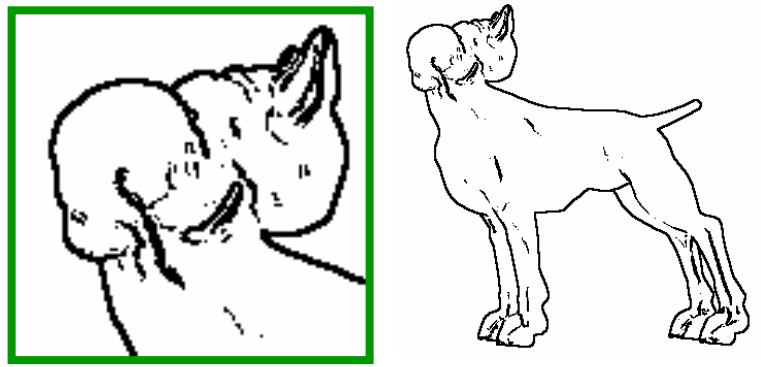


Normal Map Blurring

- We used a filter to blur the normal map.
- This additional step reduces the unnecessary *tiny* segments in surface.

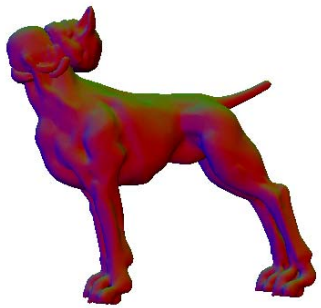


Before filtering



After filtering

Overview of Edge Simplification

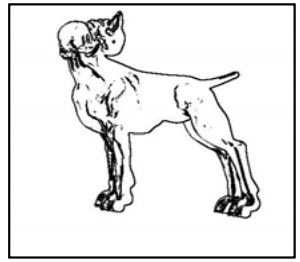


Normal map

Mean filter

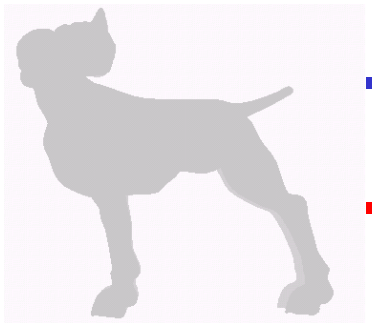
$$M = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Blurred normal map

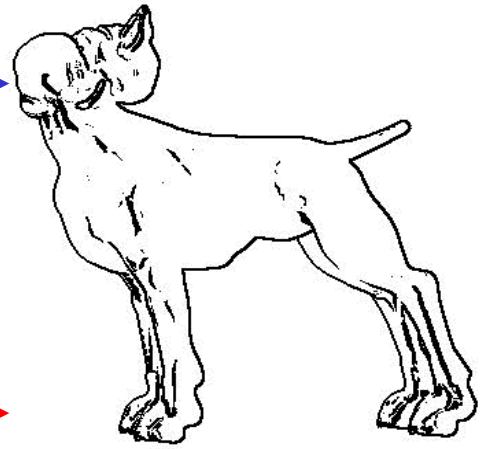


Without simplification

Filter Function



Depth map



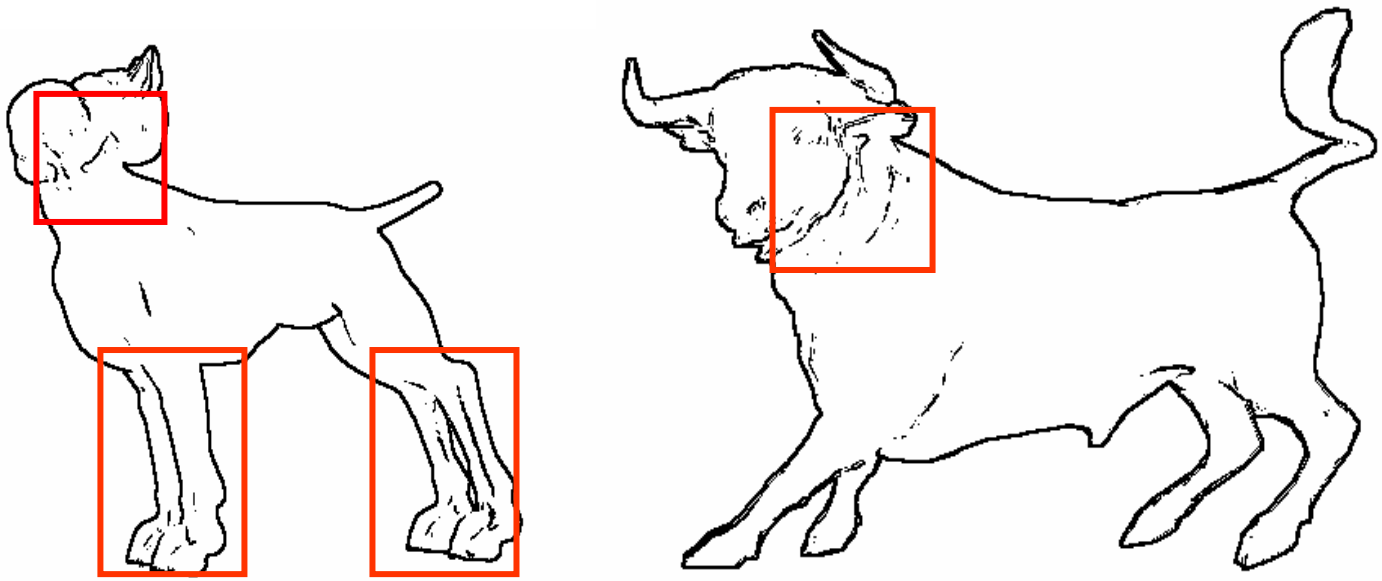
Contour map

Crease edge

Silhouette edge

Result of Edge Simplification

- Edge simplification can help users to focus on the actually important contours.

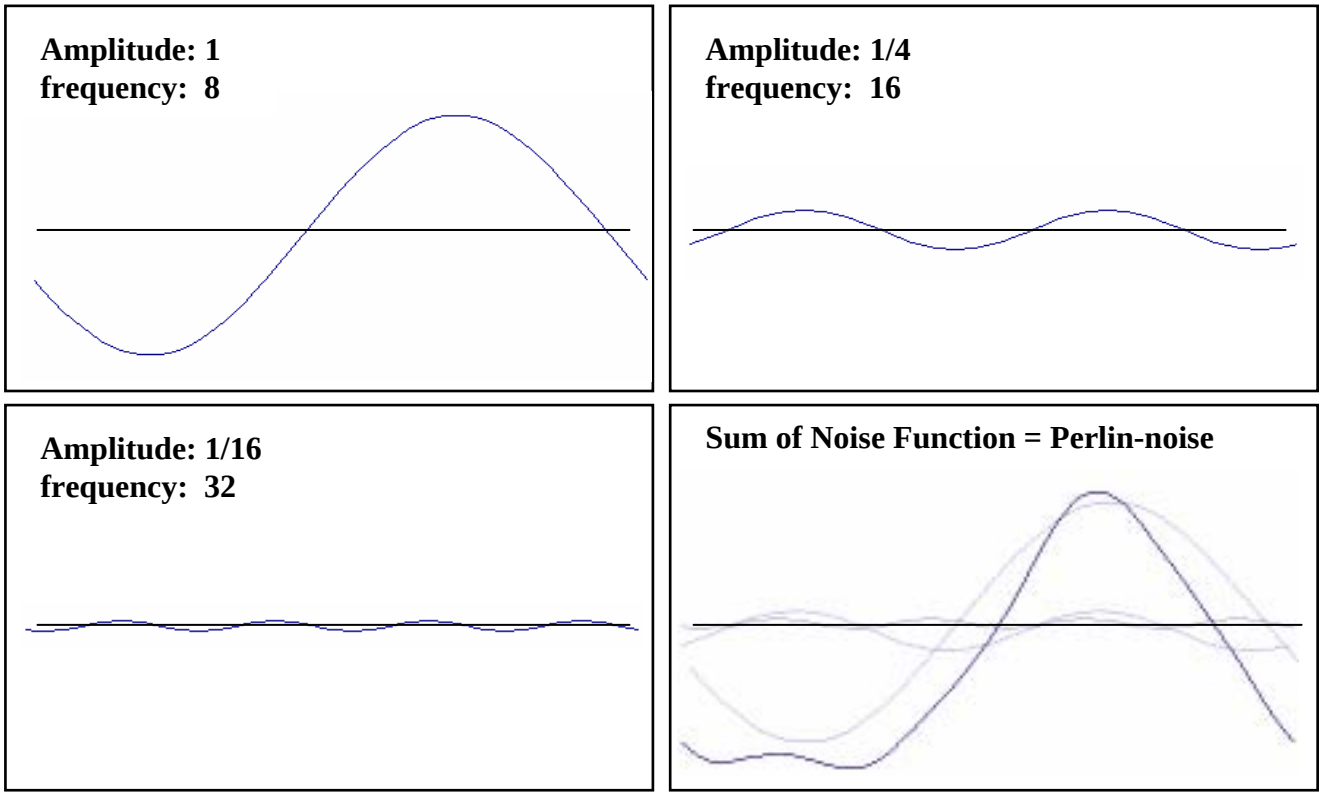


Stylized Contour

- **We utilized a noise function and redrew the contour several times to simulate *wiggliness*扭動 of contour drawing.**
- **Random number is certainly used in error noise, but the randomness may be too harsh to appear natural at times.**
- **In our system, we chose Perlin-noise to imitate the randomness of human drawing.**

Perlin-Noise

- Perlin-noise is approximated by mixing a simple periodic function with distinct *amplitudes* and *frequencies*.



Perturbed擾動的 Mapping

- Perlin-noise exploits *perturbed map* to shake contours *indirectly*.
- There are reasons that we have to use indirect method to shake contours
  - Contour map is a pure image, all edges are composed of independent pixels.
  - In GPU programming, a pixel shader cannot write a value to another except the current one.
- We generated perturbed maps in pre-processing stage.

Example Perturbed Mapping

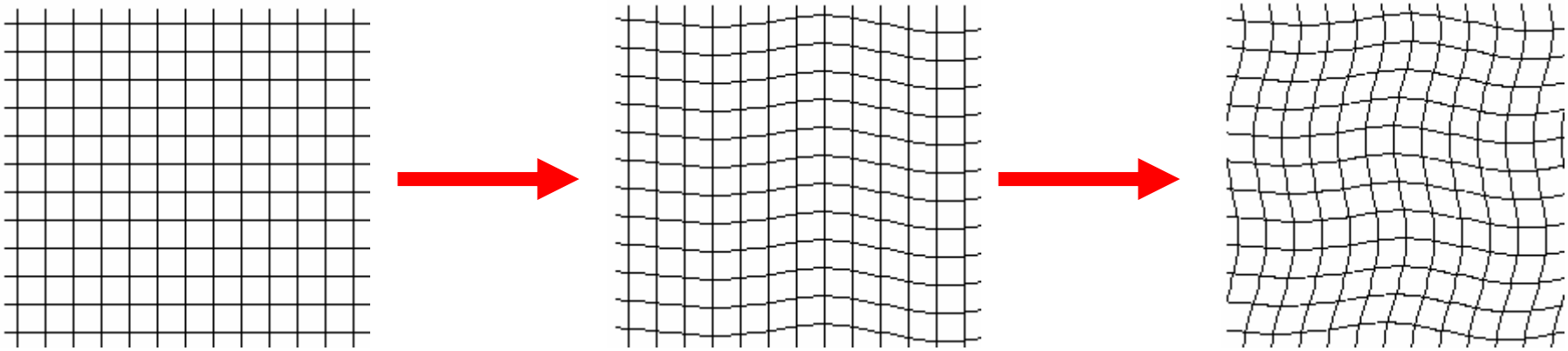
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

5	-1	2	8	4
10	1	7	13	9
15	6	12	18	14
20	11	17	23	19
-1	16	22	-1	24

5	-1	2	8	4
-1	10	1	7	13
-1	-1	15	6	12
11	17	23	19	-1
-1	16	22	-1	24

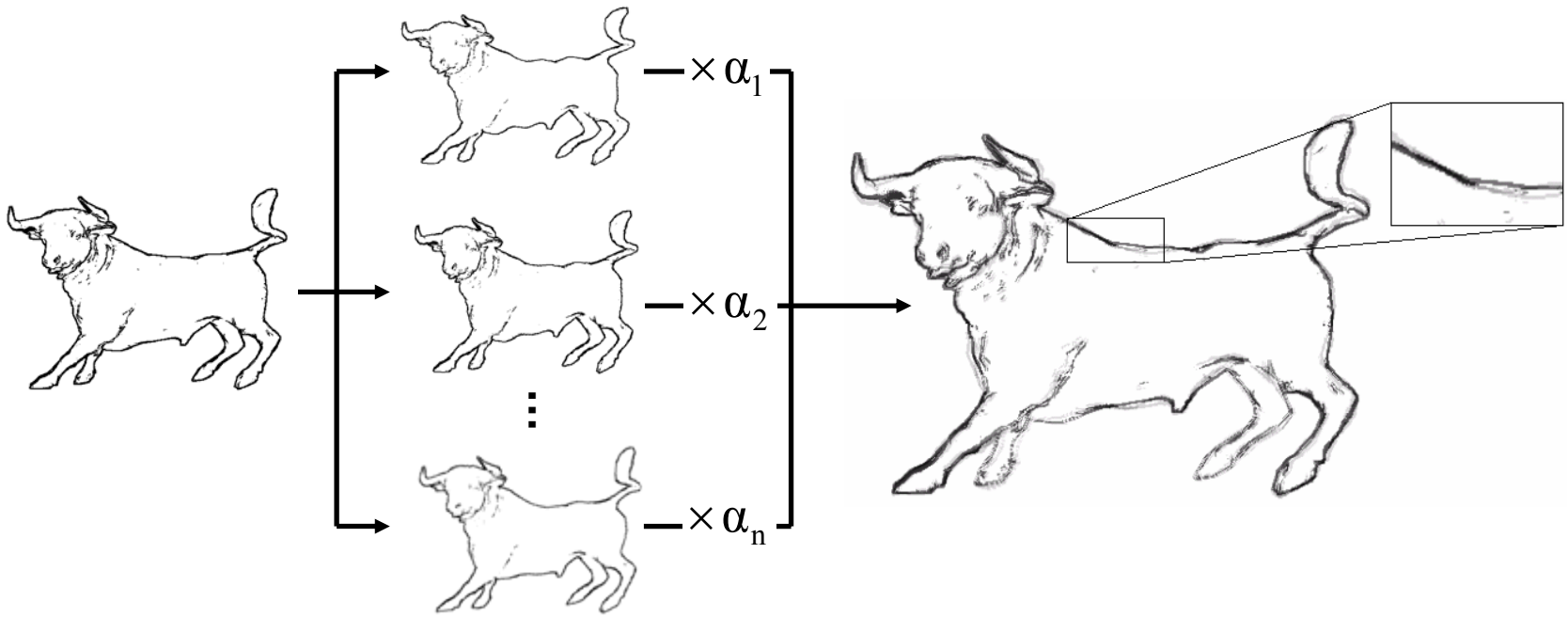
Perturbing Y-axis

Perturbing X-axis

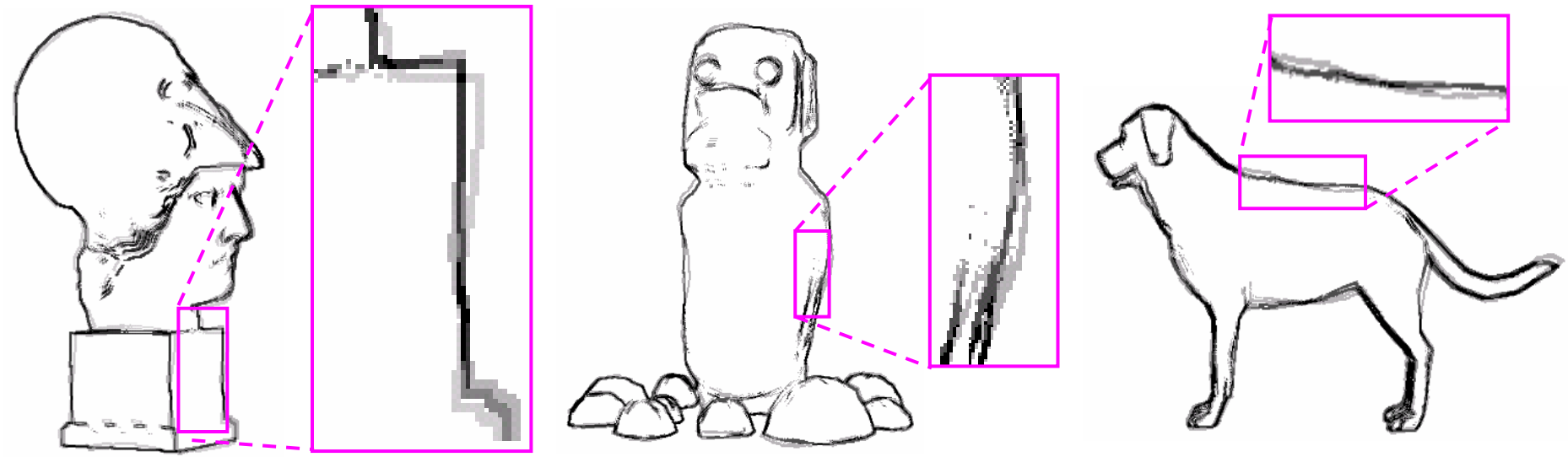


Multiple Contours

- A line drawn by human is usually composed of overlapped similar but different lines.
- Our system composed several contours with distinct Perlin-noise to achieve human drawing effect.



Result of Multiple Contours



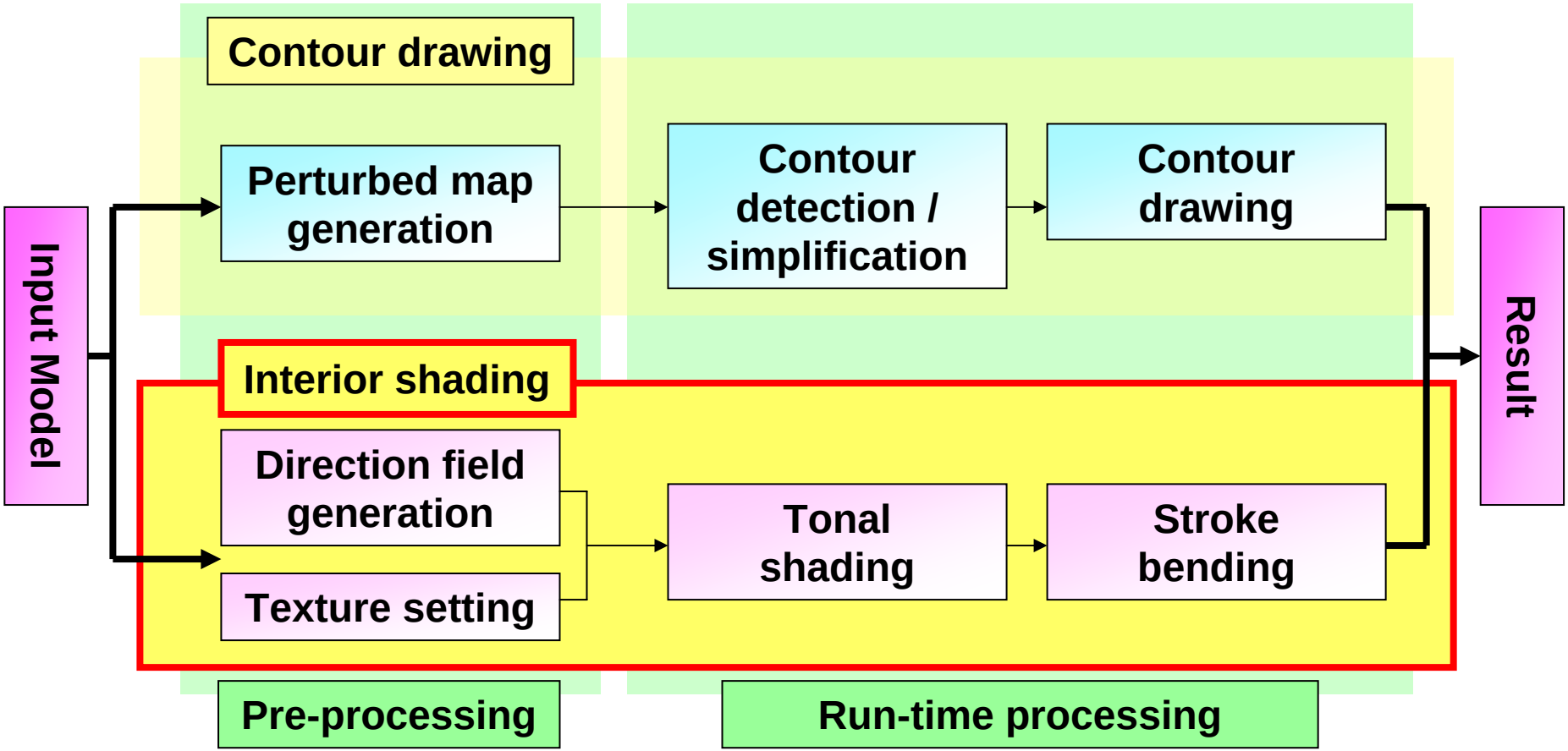
Compared with Other Works

- Comparing the perturbed result to other similar recent works.

J.P. Lewis et al. [More optimal strokes for NPR sketching.]

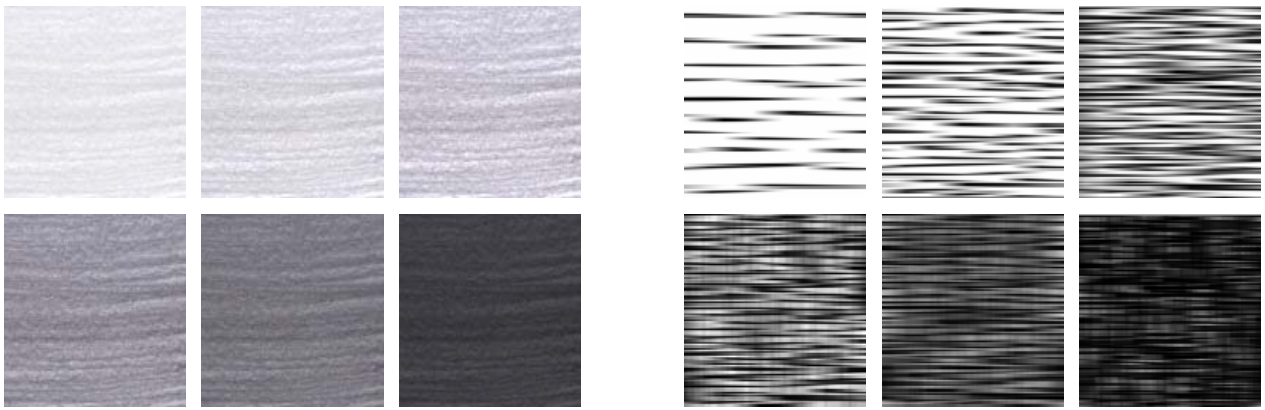
Our work

Interior Shading



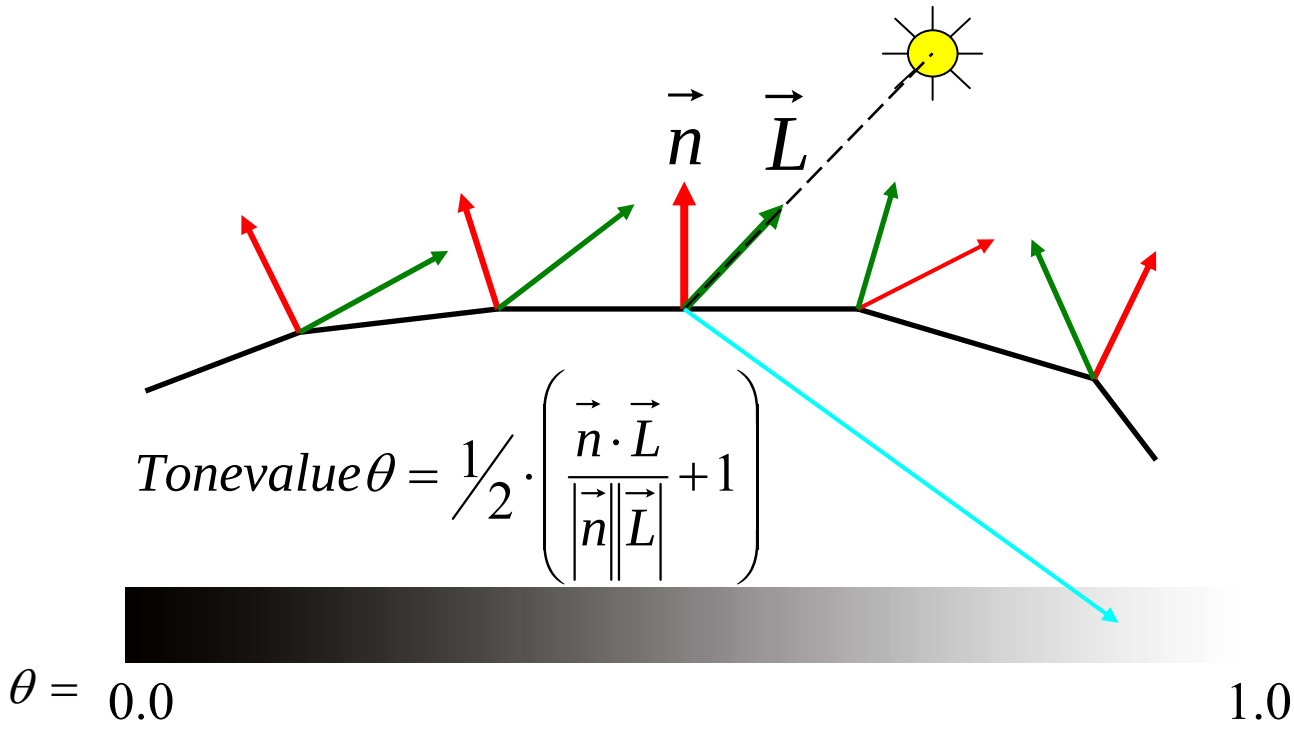
Stroke筆劃 Shading

- Drawing strokes as individual primitives is very *expensive*; it is difficult to render each stroke individually.
- We rendered the complex detail using texture map which includes a group of strokes in one texture.



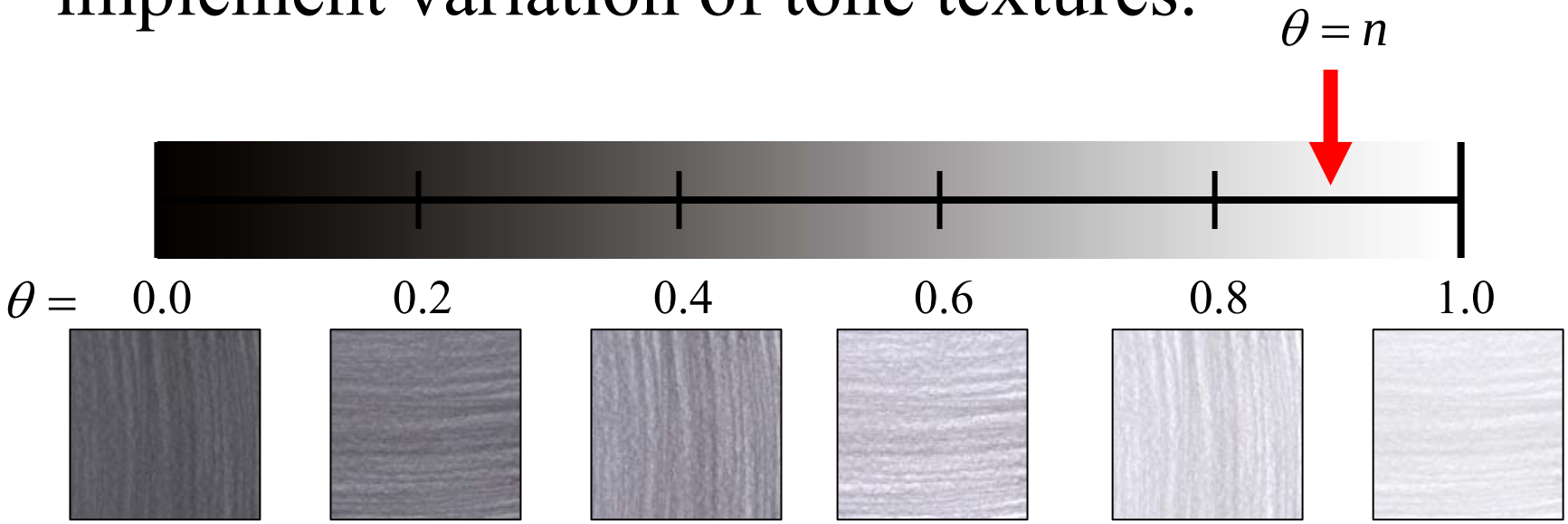
Calculating Tone Value

- Utilized the angel between *vertex normal* and *light source* to determine intensity of brightness.
- Tone value is a floating point number in [0, 1].

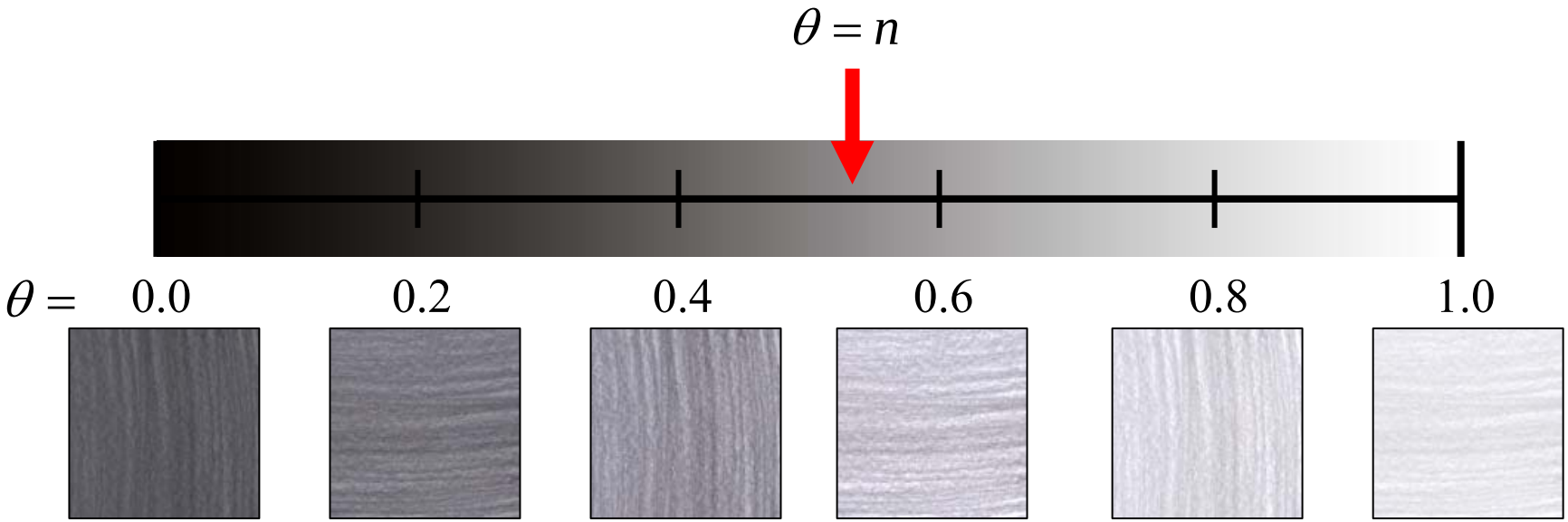


Tonal Texture

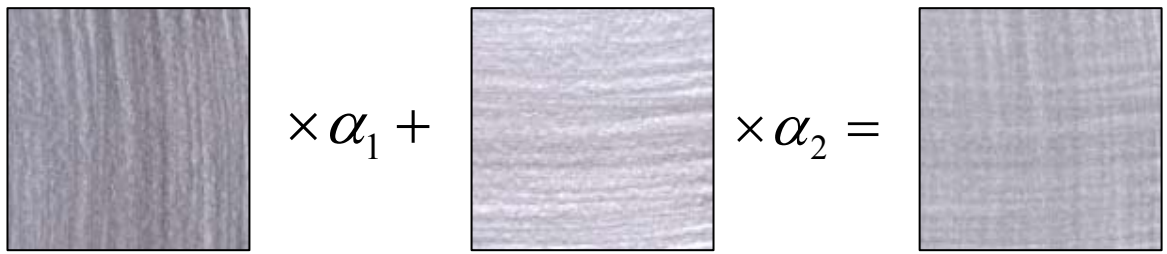
- It is impossible to prepare *infinite* number of stroke textures to adapt tone value we calculated.
- System exploits *color-buffer blending* to implement variation of tone textures.



Texture Multi-Rendering

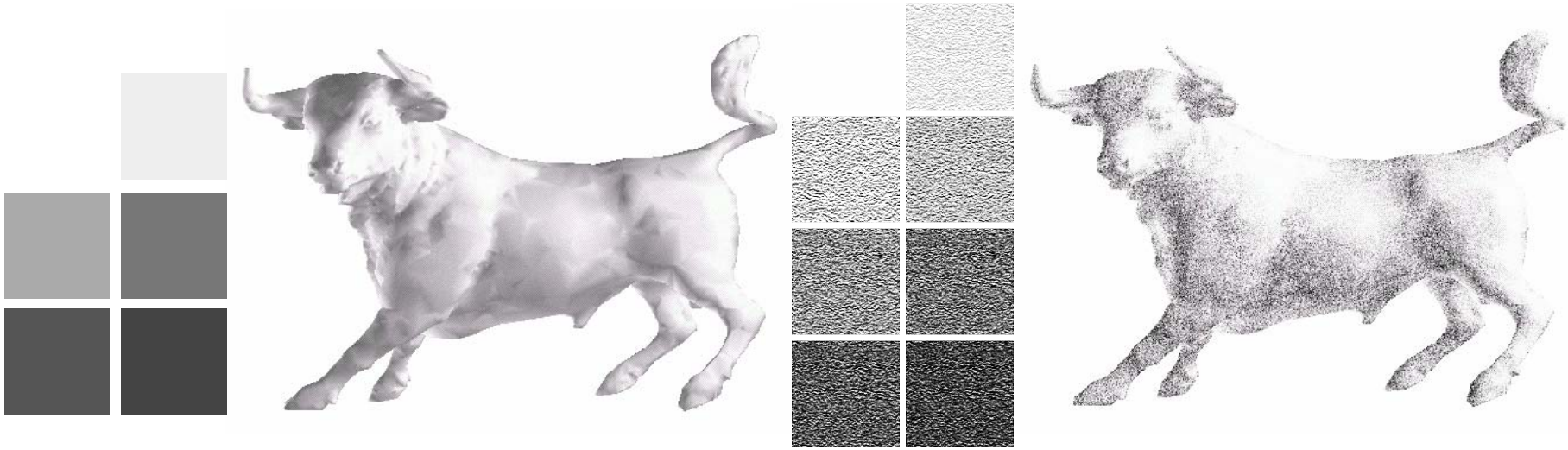


$$contribution\,\alpha_1 = \frac{n - 0.4}{0.6 - 0.4} \qquad contribution\,\alpha_2 = \frac{0.6 - n}{0.6 - 0.4}$$



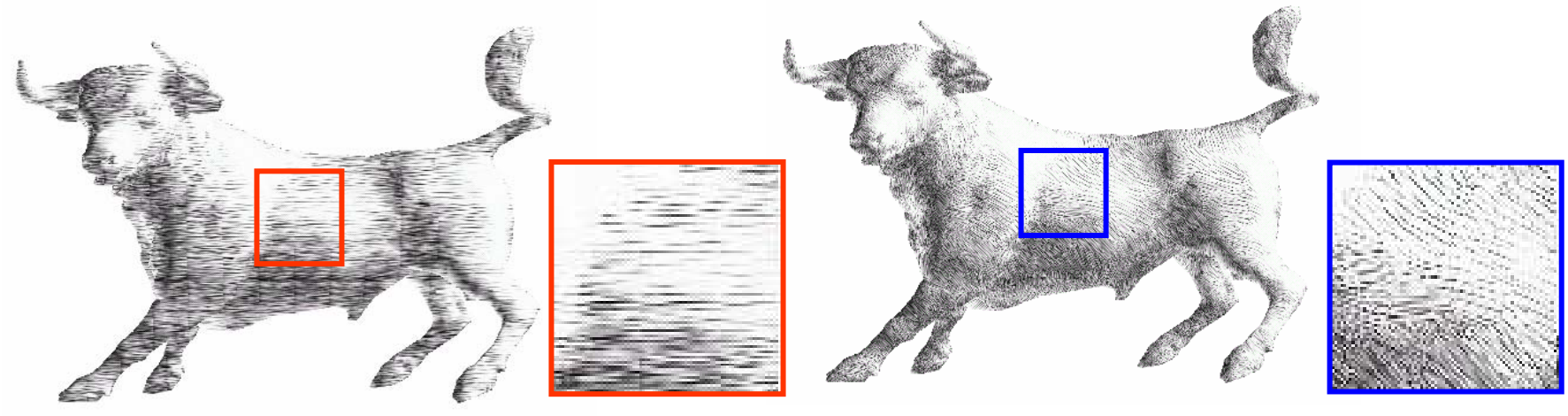
Result of Multi-Rendering

- Multi-rendering smoothly renders objects using a small number of textures.



Stroke Bending彎曲

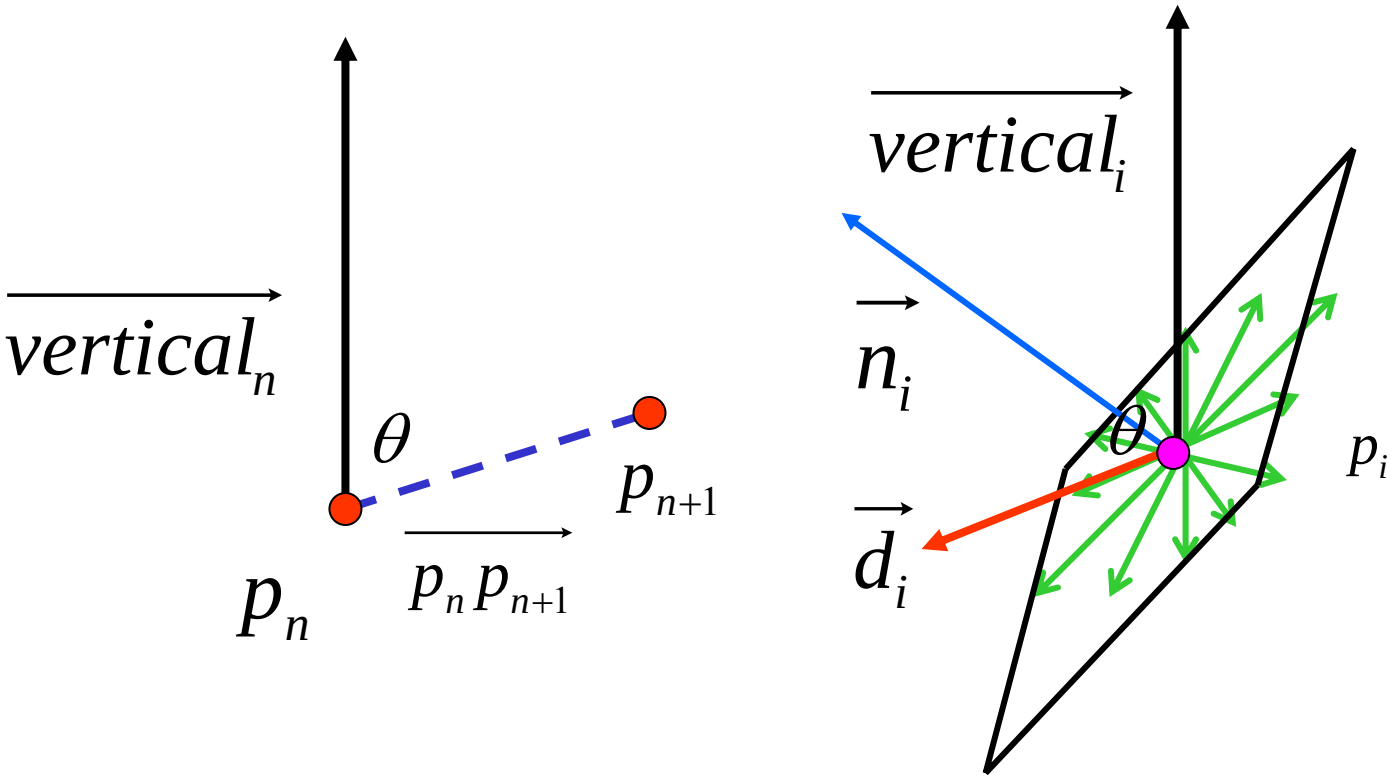
- **Variation of *curvature direction* is usually used to describe the modeling of object.**
- **Strokes of interior shading are enhanced by *direction field*方向場; we *rotated* textures to simulate the changes of curvature.**



Without direction field

With direction field

Calculating Direction



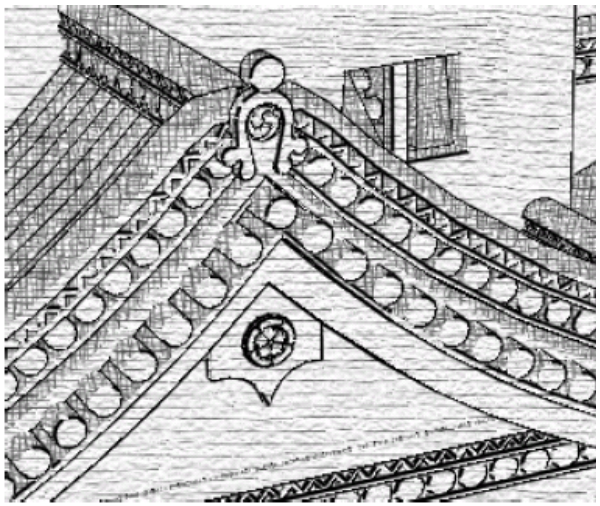
Direction Field

- Calculate stroke directions in *pre-processing* stage; store them in the direction field.
- In *run-time*, stroke texture rotates using the information in direction field.

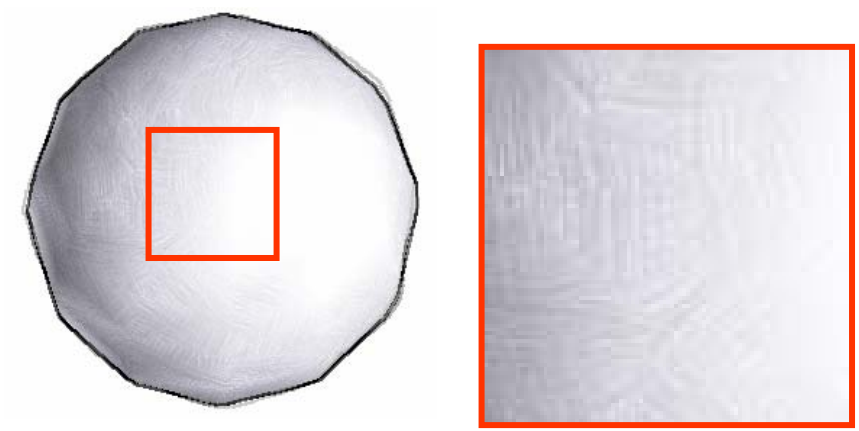
- There are two types of texture rotation:
  - **triangle**-based
  - **vertex**-based

Contrasted with Other Works

- Comparing our work to other similar recent works.



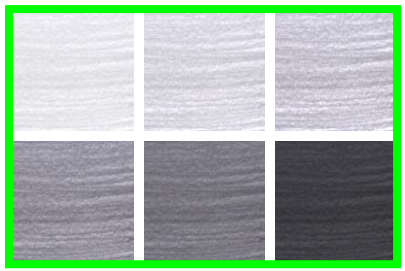
Adam Lake et al. [Stylized rendering techniques for scalable real-time 3D animation.]



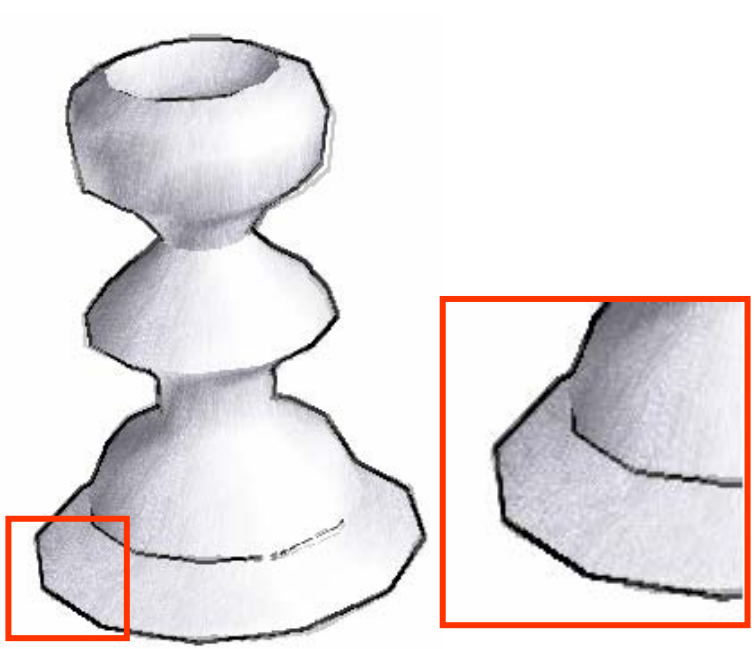
Our work

Demonstrating Results

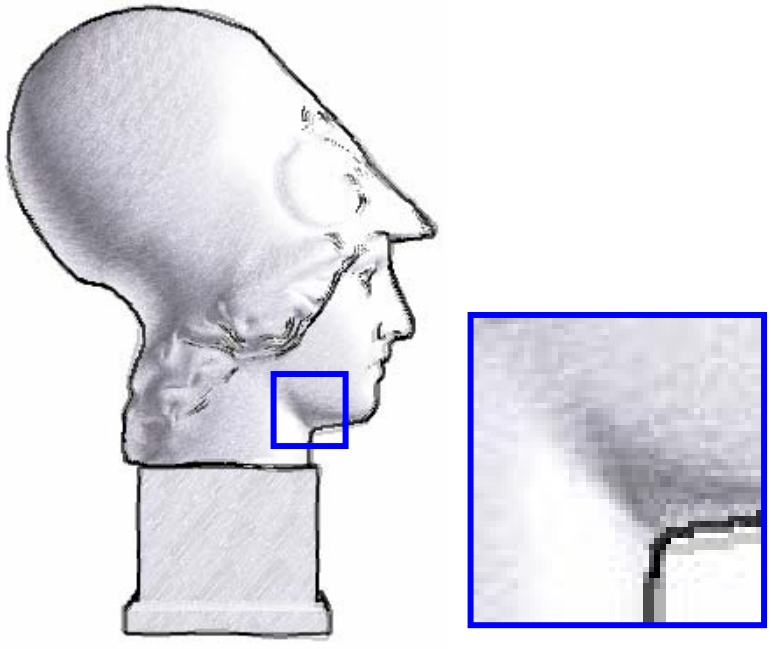
Sketchy Style



- Utilized *pencil* stroke texture and *triangle*-based direction field to simulate sketchy drawing.

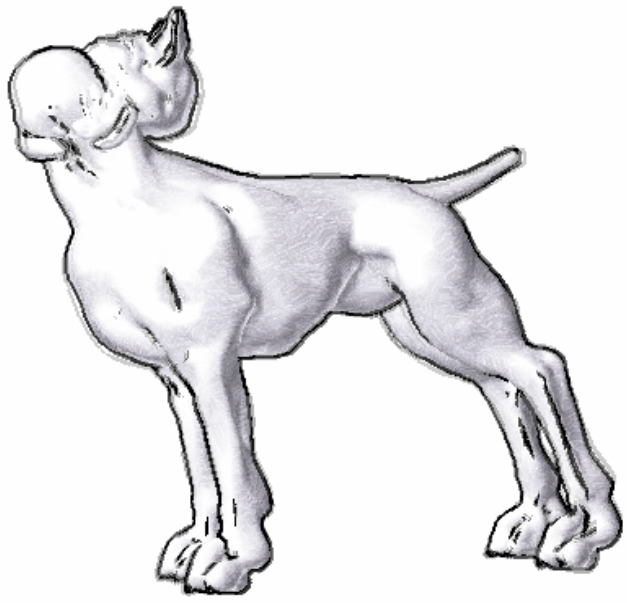


Chess model

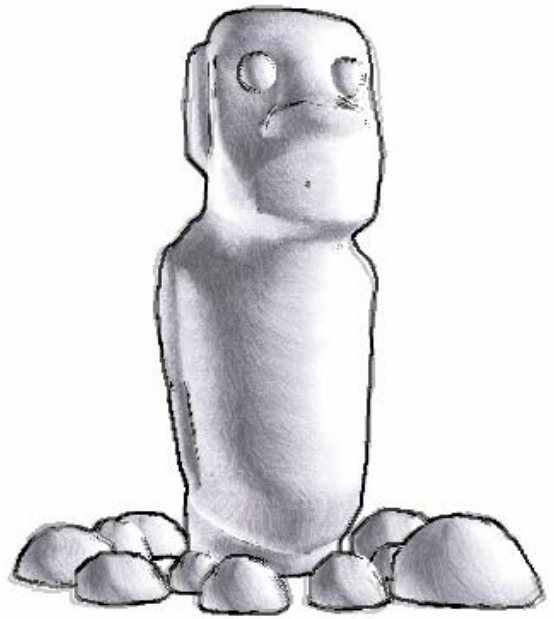


Athena希臘女神 model

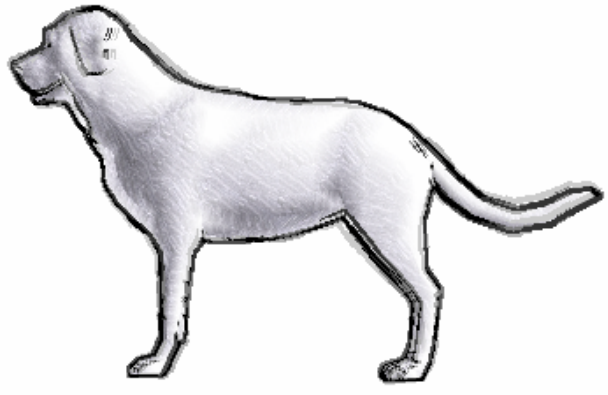
(2)



Pitbull鬥牛犬 model



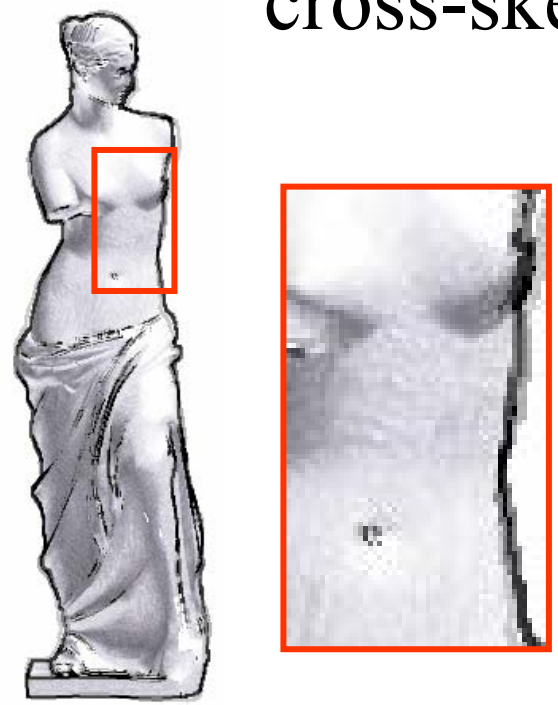
Easter model



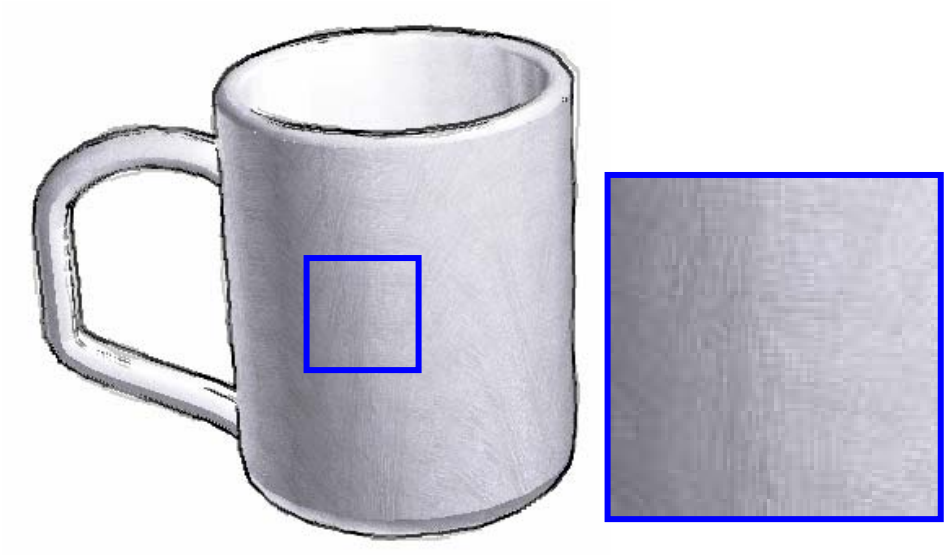
Dog model

Cross-Sketchy Style

- Utilized *pencil* stroke texture and *vertex*-based direction field to simulate cross-sketchy drawing.



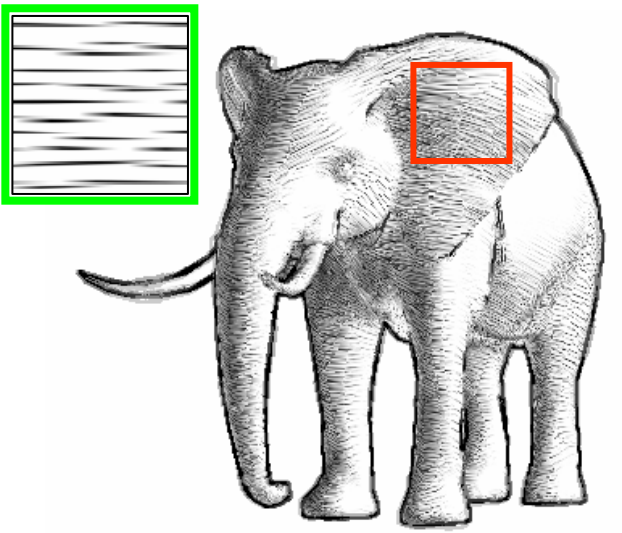
Venus model



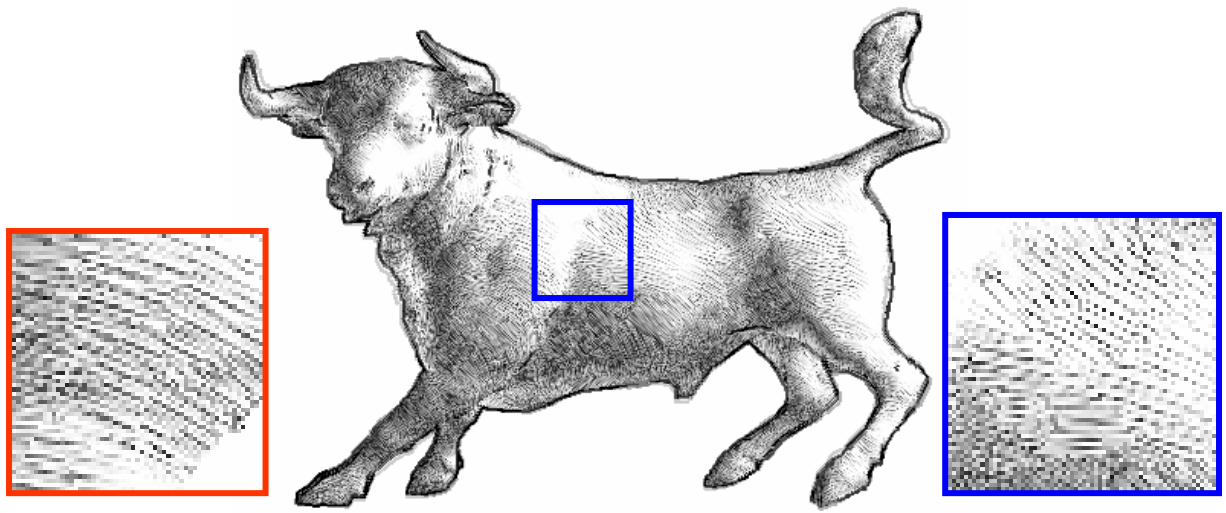
Coffee-cup model

Hatching Style

- Utilized *pen-and-ink* stroke texture and *triangle*-based direction field to simulate hatching.

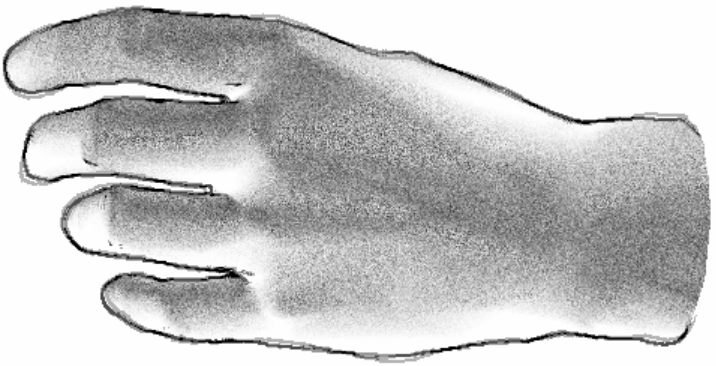


Elephant model

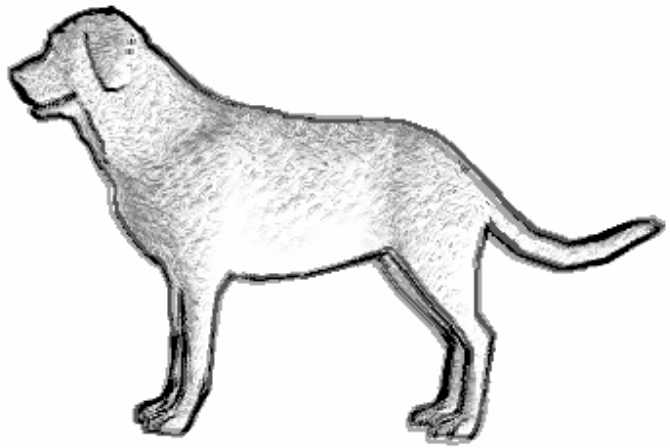


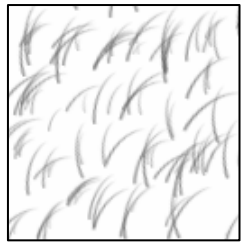
Bull model

Other Styles

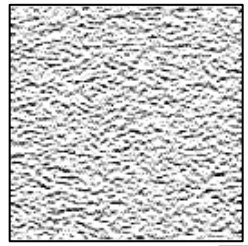


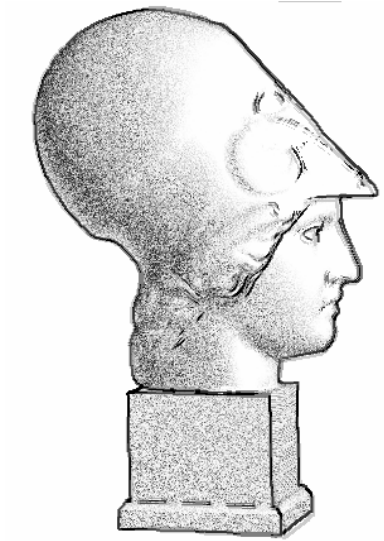
Fog style





Fur style





Stone style

Performance Diagnosis

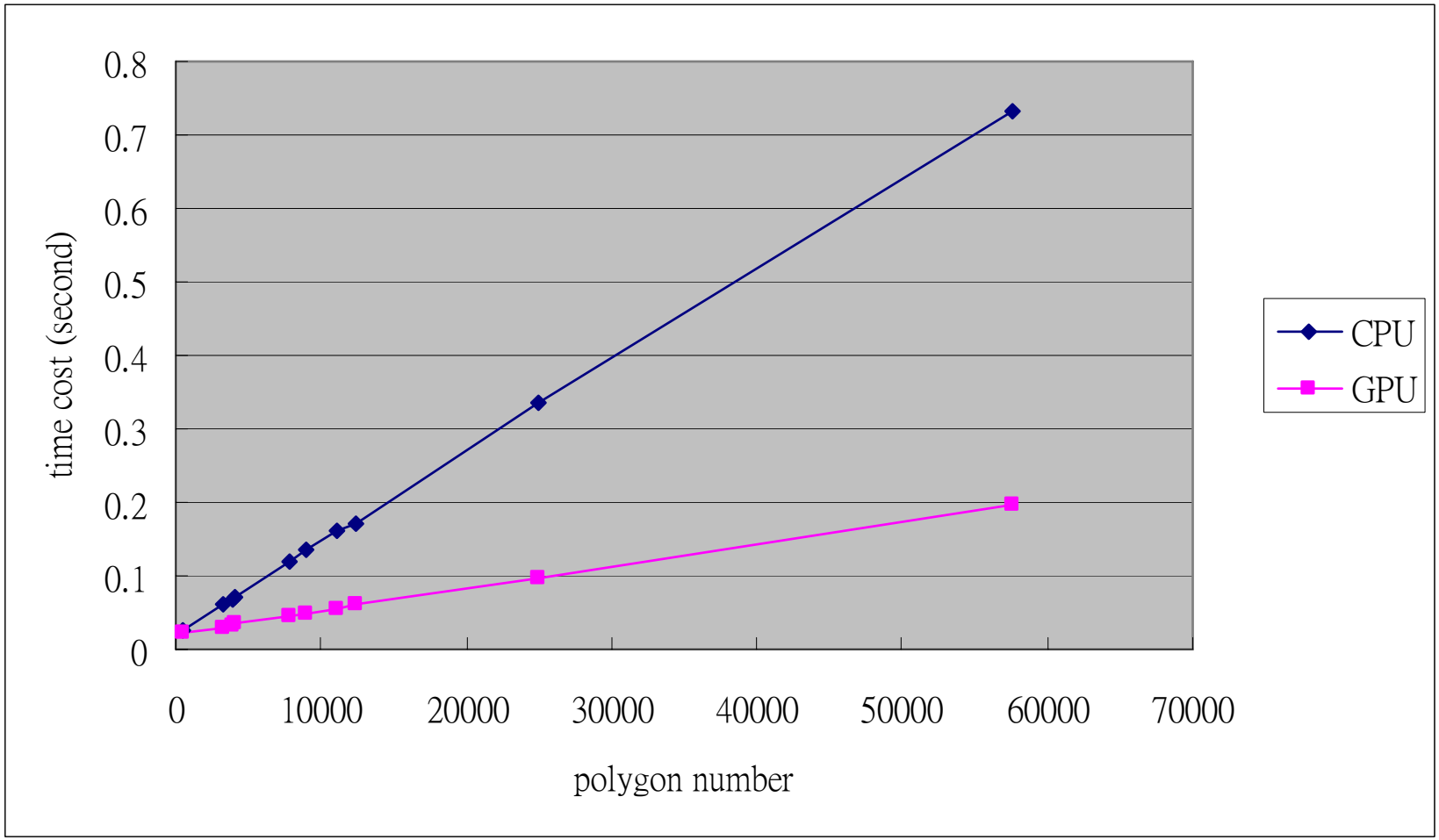
Programming component	Time measured (sec)	% of overall time cost
Reading normal map	0.002	1.7
Reading depth map	0.002	1.7
Shading with multi-texture blending	0.092	81.5
Sobel mask	0.004	3
Redundant edges removal	0.01	8.7
Combining shading & contouring	0.004	3.4
Total	**0.114**	**100**

Performance of each programming component in the system. (Input: *Moai* model)

Timing Data for Using GPU

Model name	# of polygon	Time cost of a frame (second)	# of frames per second
Human head	426	0.022	45.5
Dog	3220	0.031	32.3
Rubber duck	3864	0.033	30.3
Human hand	4057	0.035	28.6
Elephant	7897	0.045	22.2
Moai	8956	0.048	20.9
Dragon	11102	0.055	18.2
Bull	12398	0.059	16.9
Pitbull	25030	0.098	10.2
Dinosaur	57532	0.198	5.1

Hardware Acceleration



Conclusions

- NPR = Computer Graphics as an Interpretive and Expressive Medium
  - Communicate shape and structure
  - Emphasize or de-emphasize
  - Prevent visual overload
  - Suggest artificiality
  - Ensure visibility of important structures
  - Provide spatial context
- **As detailed as necessary – as simple as possible.**