Visualization and Imaging

Summer Term 2015

Overview

- Literature:

  [1] C.D. Hansen, C.R. Johnson: The Visualization Handbook, Elsevier, 2005

  [2] I.N. Bankman: Handbook of Medical Imaging, Academic Press, 2000

  [3] R.C. Gonzales, R.E. Woods: Digital Image Processing, Prentice Hall, 2002
- Software:
  Amira, Paraview, VisIt, Matlab, ...
- Topics:
  Histogram Modification, Filtering, Segmentation, Vector Field Visualization, Volume Rendering
- Data:
  Microscopy, CT, MRI, telescope, satellite, ...
  Image files (TIFF, JPG, BMP, ...)
  Simulation results (Matlab, ...)

Notations

- **Image**: Function $f : \Omega \to \mathbb{R}$ (scalar grayscale), $f : \Omega \to \mathbb{R}^3$ (scalar color, vectorial grayscale), ..., for some domain $\Omega = [0,1]^2 \subset \mathbb{R}^2$ (planar image), $\Omega = [0,1]^3 \subset \mathbb{R}^3$ (volume data), ...
- **Planar Greyscale Image, Pixel Representation**:
  $\Omega = \{0, 1, \dots, M-1\} \times \{0, 1, \dots, N-1\}$,

  $$f(m,n) \in \{0, 1, \dots, P-1\}, \qquad m = 0, \dots, M-1,\ n = 0, \dots, N-1$$

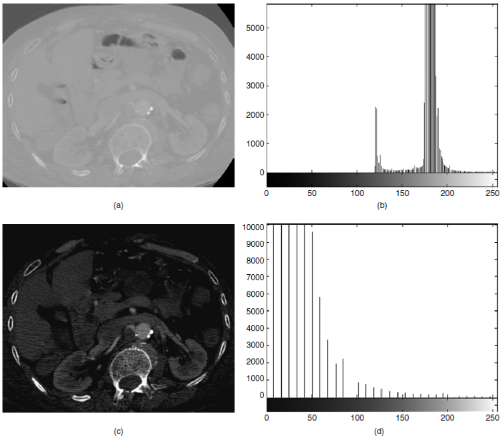
  denotes the intensity at pixel (m,n).
- **Histogram, Pixel Representation**:

  $$h(i) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \delta(f(m,n) - i), \qquad i = 0, \dots, P-1$$

  with $\delta(x) = 1$, if $x = 0$, and $\delta(x) = 0$, else

Intensity Scaling

Image infomration might only be present in small intensity bands



Intensity Scaling

Fix intensity limits f_1, f_2 in which the information of f is contained. The enhanced image g is obtained by

$$c(m,n) = \begin{cases} f(n,m), & \text{if } f_1 \leq f(m,n) \leq f_2 \\ 0, & \text{else} \end{cases}$$

$$g(m,n) = \frac{c(m,n) - f_1}{f_2 - f_1} \cdot (P-1)$$

Disadvantage: Details outside $[f_1, f_2]$ are completely opressed

Histogram Equalization

Distribute intensity information uniformy across the histogram (goal: an approximate pixel count of $\frac{MN}{P}$ per intensity)

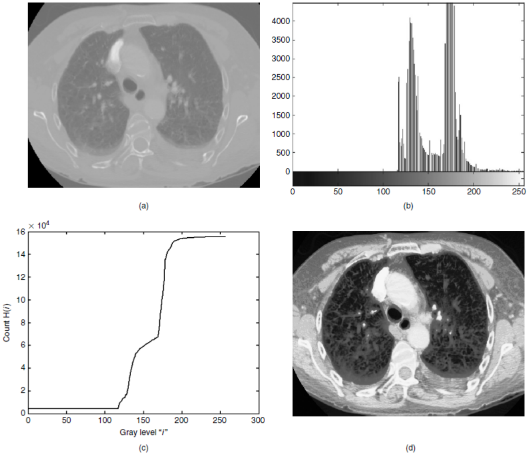
Normalized Cumulative Histogram:

$$H(j) = \frac{1}{MN} \sum_{i=0}^{j} h(i), \qquad j = 0, \ldots, P-1$$

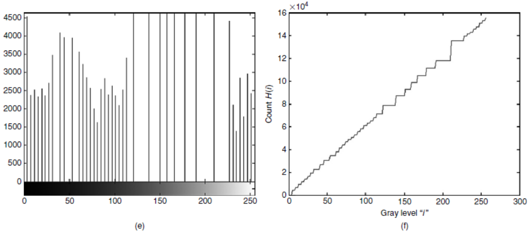
The enhanced image g is obtained by

$$g(m,n) = (P-1) \cdot H(f(m,n))$$

Histogram Equalization



Histogram Equalization



Disadvantage: Contrast level takes only global information into account; noise might be enhanced as well

Local Area Histogram Equalization

Apply Histogram Equalization to a small areas around each pixel. For a pixel (m, n), a local area of size $(2K + 1) \times (2L + 1)$ is defined by

$$LA(m, n) = \{m - K, \dots, m, \dots, m + K\} \times \{n - L, \dots, n, \dots, n + L\}$$

Local Area Histogram:

$$h_{LA(m,n)}(i) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} \delta(f(m - k, n - L) - i), \qquad i = 0, \dots, P - 1$$

Normalized Local Area Cumulative Histogram:

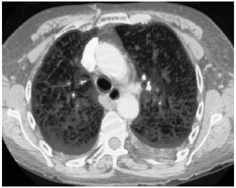
$$H_{LA(m,n)}(j) = \frac{1}{(2K + 1)(2L + 1)} \sum_{i=0}^{j} h_{LA(m,n)}(i), \qquad j = 0, \dots, P - 1$$

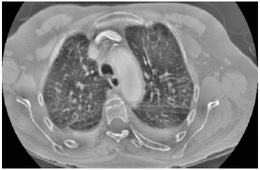
The enhanced image g is obtained by

$$g(m, n) = (P - 1) \cdot H_{LA(m,n)}(f(m, n))$$

Local Area Histogram Equalization

Histogram Equalization vs. Local Area Histogram Equalization

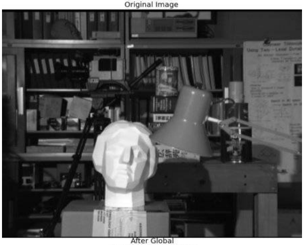




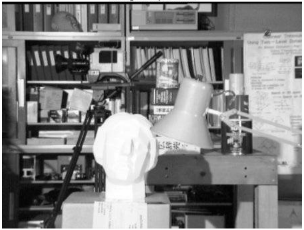
Local Area Histogram Equalization

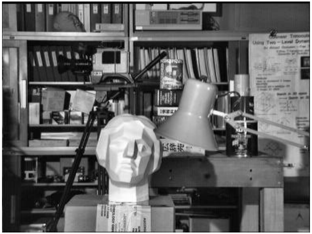
Histogram Equalization vs. Local Area Histogram Equalization

Original Image



After Global
Histogram Equalization





Contrast Limited (Local Area) Histogram Equalization

Contrast Enhancement can be defined via the slope of the function that maps the original intensity i of the image f to the new intensity $(P-1)\cdot H(i)$ of the image g. In other words, contrast enhancement is reflected by the 'derivative'

$$(P-1)\frac{d}{di}H(i) = (P-1)(H(i) - H(i-1)) = \frac{P-1}{MN}h(i)$$

Cutting off the histogram h at some value h_{max} restricts the enhancement of the contrast. This can help to reduce the enhancement of noise.

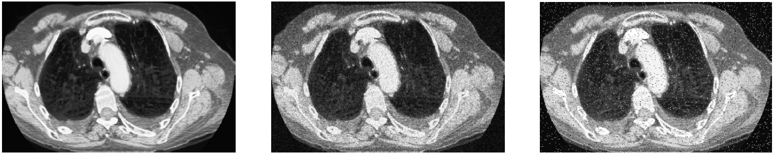
$\longrightarrow$ Amira, Tutorials/BrainMap/DICOM, ImageProcessing $\rightarrow$ GrayscaleTransforms $\rightarrow$ HistogramEqualization (or ... $\rightarrow$ AdaptiveHistogramEqualization), MeasureAndAnalysze $\rightarrow$ Histogram, OrthoSlice

Noise

Often, measurements are contaminated by (additive) **noise** n:

$$f = f_0 + n$$

Denoising produces an output image g from the input image f with (hopefully) $g \approx f_0$. Often used as initial steps for further image processing.



Original Image f_0, Gauss noise image f, and Salt and Pepper noise image f

Notations

- **Convolution**: Kernel $w : \Omega \to \mathbb{R}$, $\Omega = [0, 1]^2 \subset \mathbb{R}^2$,

$$g(x, y) = w * f(x, y) = \int_\Omega w(x - u, y - v) f(u, v) du\, dv$$

- **Fourier Transform**:

$$F(u, v) = \hat{f}(u, v) = \int_\Omega e^{-2\pi i (x,y)\cdot(u,v)} f(u, v) du\, dv, \quad u, v \in \mathbb{Z}$$

- **Convolution and Fourier Transform**:

$$G(u, v) = W(u, v) F(u, v)$$

Notations

- **Convolution, Pixel Representation**: Kernel $w \in \mathbb{R}^{(2K+1)\times(2L+1)}$,

$$g(m,n) = (w * f)(m,n) = \sum_{k=-K}^{K} \sum_{l=-L}^{L} w(k,l) f(m-k, n-l)$$

- **Fourier Transform, Pixel Representation**:

$$F(u,v) = \hat{f}(u,v) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-2\pi i \left(\frac{um}{M} + \frac{vn}{N}\right)}, \qquad \begin{array}{l} u = 0,\ldots,M-1, \\ v = 0,\ldots,N-1 \end{array}$$

- **Convolution and Fourier Transform, Pixel Representation**:

$$G(u,v) = W(u,v)F(u,v)$$

Smoothing Filters

Linear filters: Typically convolutions of form

$$g = T(f) = w * f,$$

w is called **filter/covolution kernel**. Examples are

- **Averaging kernel** in a $(2K+1) \times (2L+1)$-neighborhood. For $L = K = 1$:

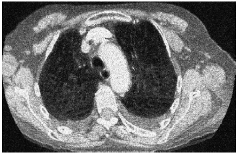
$$w = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- **Gauss kernel**: $w(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}$, with σ the standard deviation. For $K = L = 2$ and $\sigma = 1$, the normalized convolution kernel reads

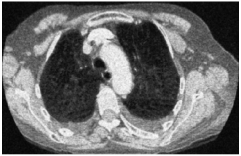
$$w = \frac{1}{273}\begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix}$$

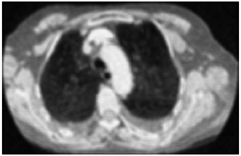
Smoothing Filters

Original Image, Gaussian noise



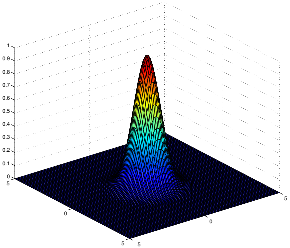
Gauss filtered image, Average filtered image

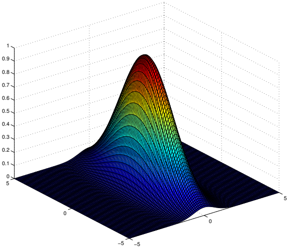




Smoothing Filters

Gaussian functions $e^{-\left(\frac{x^2}{\sigma_1^2}+\frac{y^2}{\sigma_2^2}\right)}$ for $\sigma_1 = \sigma_2 = 1$ and $\sigma_1 = 1,\ \sigma_2 = 3$





Smoothing Filters

- **Diffusion**: Gaussian filtering can be related to the PDE

$$\frac{d}{dt} g(t,x,y) = D\Delta g(t,x,y), \qquad g(0,x,y) = f(x,y)$$

A possible discretization of the PDE leads to a convolution with the filter kernel

$$w = \begin{pmatrix} 0 & \alpha & 0 \\ \alpha & 1-4\alpha & \alpha \\ 0 & \alpha & 0 \end{pmatrix}$$

for some $\alpha > 0$. The PDE can be modified by introducing a spatially varying (matrix valued) diffusion tensor D:

$$\frac{d}{dt} g(t,x,y) = \nabla \cdot \left(D(x,y)\nabla g(t,x,y)\right), \qquad g(0,x,y) = f(x,y)$$

Smoothing Filters

- **Wiener Filter**: Optimze kernel w such that, for $g = w * f$, the mean square error is minimized:

$$E[(g - f_0)^2] \to \min$$

This is satisfied for the kernel w with Fourier transform

$$W(u, v) = \frac{R_{f_0 f_0}(u, v)}{R_{f_0 f_0}(u, v) + \sigma^2},$$

where $r_{f_0 f_0}$ is the auto-correlation of f_0 and n white noise with variance σ^2.

Unsharp Masking

Emphasize local features in an image (does not preserve overall intensity of the image). The following steps are required

1. Low pass (smoothing) filter

$$g_{lp} = w_{lp} * f,$$

   the kernel w_{lp} can be, e.g., the Gauss kernel

2. High pass (local) information

$$g_{hp} = f - g_{lp}$$

3. Unsharp masking:

$$g = g_{lp} + \alpha g_{hp},$$

   where $\alpha < 1$ leads to smoothing, $\alpha = 1$ yields the original image f, $\alpha > 1$ highlights high pass (local) features

⟶ Amira, no_noise.png ImageProcessing → SmoothingAndDenoising → Gaussian Filter, Compute → Arithmetic, ImageProcessing → Sharpening → Unsharp Masking (more options)

Smoothing Filters

Nonlinear filters $T : \{0, \dots, P-1\}^{M \times N} \to \{0, \dots, P-1\}^{M \times N}$ are more difficult to characterize. Improvement for preservation of edges. Examples are:

- **Median Filter**: For each pixel (m, n) define an environment, e.g., $LA(m,n) = \{m-K, \dots, m, \dots, m+K\} \times \{n-L, \dots, n, \dots, n+L\}$ and choose the median intensity in that environment:

$$g(m,n) = \text{median}\{f(k,l) : (k,l) \in LA(m,n)\}$$

- **Nonlinear Diffusion**: The function D may depend on g and ∇g in order to account for edges

$$\frac{d}{dt} g(t,x,y) = \nabla \cdot \left(D(x,y,g(t,x,y), \nabla g(t,x,y)) \nabla g(t,x,y)\right),$$
$$g(0,x,y) = f(x,y)$$

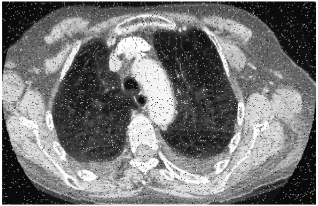
Perona-Malik: e.g. $D(|\nabla g(t,x,y))|) = e^{-\frac{|\nabla g(t,x,y))|^2}{2\sigma^2}}$.

$\longrightarrow$ Amira, sp_noise.png, gauss_noise, ImageProcessing $\to$ SmoothingAndDenoising $\to$ ...

Smoothing Filters

Median Filtered Salt-and-Pepper Noise

(a)



(b)

(c)

What is Segmentation?

- Separating an image into **foreground/background**, subdivide image into regions of **similar properties**, subdividing image into regions **separated by certain structures**
- Two main approaches:
  1. **Discontinuities**: Separate image based on edges
  2. **Similarity**: Separate image based on similar properties
- There is **no universal** segementation technique
- Manual Separation is only feasible for small data sets

Edge Detection

Visualize edges in pictures. Typically based on gradients: prone to noise. Previous smoothing is advisable.

- **Prewitt Filter**:

$$w_x = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \qquad w_y = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

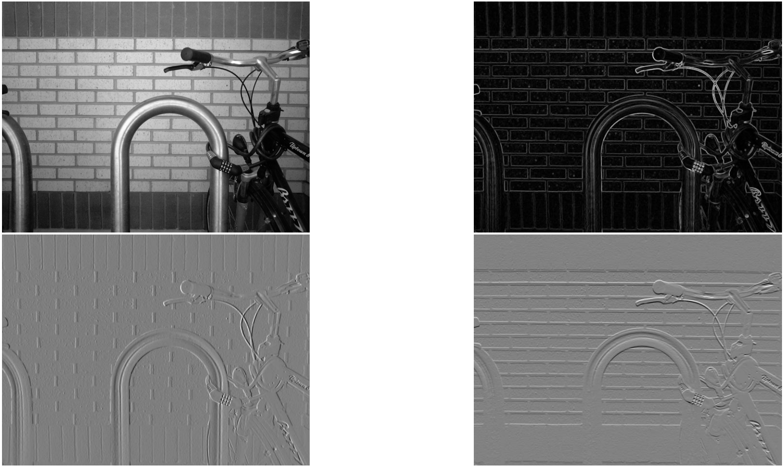
- **Sobel Filter**:

$$w_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}, \qquad w_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$\longrightarrow$ Amira, ImageProcessing $\rightarrow$ EdgeDetection $\rightarrow$ Gradient $\rightarrow$ SobelFilter

Edge Detection

Original picture, Sobel filter in x-direction, Sobel filter in y-direction, Sobel filter in x- and y-direction



Edge Detection

- **Laplace Zero-Crossing** Maximum gradients are indicated by zero-crossings of the second derivative. Use Laplace operator, e.g.,

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

  to find zero-crossings

- **Canny Edge Detection**:
  1. Apply Gaussian Filter for smoothing
  2. Find intensity gradients, e.g., by Sobel filter
  3. Find direction of potential edges in each pixel via $\theta = \arctan(\frac{w_y * f}{w_x * f})$
  4. Apply Nonmaximum Suppression to find edge pixels
  5. Retrace edges

Disadvantage: Edge Detection via gradients very sensitive to noise; no correlation among pixels, hard to find closed curves

Thresholding

- Very basic principle, works well for images with **bimodal** histograms. Choose an **intensity threshold** $T > 0$ and define a segemented image by

$$g(m,n) = \begin{cases} 1, & \text{if } f(m,n) > T \\ 0, & \text{if } f(m,n) \leq T \end{cases}$$

  The areas where $g(m,n) = 1$ denote the foreground, the areas where $g(m,n) = 0$ the background.

- **Automatic determination** of threshold:
  1. Choose initial threshold T
  2. Segment image into regions G_0, G_1 by thresholding with T
  3. Compute average gray level values μ_0 and μ_1 in G_1 and G_2
  4. Compute new threshold $T = \frac{\mu_0+\mu_1}{2}$
  5. Stop if difference between the tresholds is “small enough”, else, iterate the previous steps

Thresholding

- **Otsu's Methods** for automatic determination of threshold:
  1. Initial threshold $T = 0$
  2. Separate image into background G_0 and foreground G_1 with respect to T
  3. Compute the intra-class variance

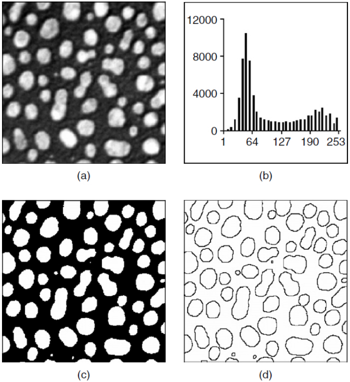
$$\sigma(T)^2 = w_0(T)\sigma_0(T)^2 + w_1(T)\sigma_1(T)^2$$

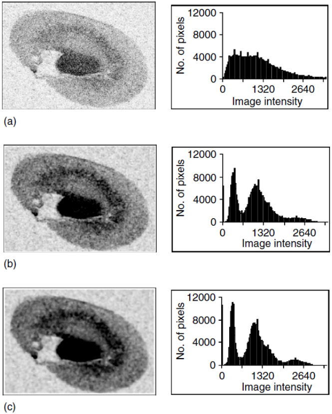
  4. Iterate this for all possible thresholds T
  5. Choose the threshold with the minimal intra-class variance $\sigma(T)$

$\longrightarrow$ Amira, lobus.am, ImageSegmentation / Multi-Thresholding

Threshholding

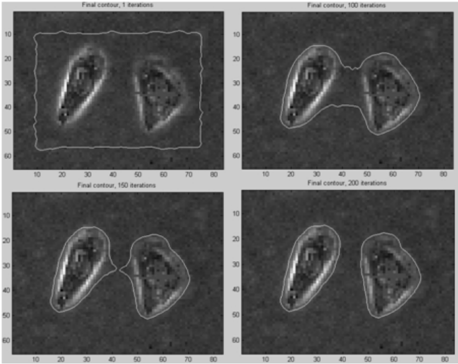
Cell threshold segmentation and Laplace Zero-Crossing (Left)
Thresholding after median filtering (Right)





(Curve Based) Active Contours

Illustration of Evolution of contour Γ



(Curve Based) Active Contours

Minimize energy functional to obtain 'well-behaved' edges Γ: E.g. length prior model

$$\begin{aligned} E_f(\Gamma) &= E_1(\Gamma) + E_2(\Gamma) \\ &= \underbrace{\alpha \int_\Gamma ds}_{\substack{\text{internal energy,} \\ \text{regularization term}}} \underbrace{+\beta \int_\Gamma h(|\nabla f|) ds}_{\substack{\text{external energy,} \\ \text{data term}}} \end{aligned}$$

or Euler's elastica prior model

$$\begin{aligned} E_f(\Gamma) &= E_1(\Gamma) + E_2(\Gamma) \\ &= \int_\Gamma (\alpha + \beta\kappa^2) ds + \mu \int_\Gamma h(|\nabla f|) ds, \end{aligned}$$

with curvature κ and some function $h : \mathbb{R}^2 \to \mathbb{R}$, e.g.

$$h(p) = e^{-\sigma p^2}, \qquad h(p) = \frac{1}{1 + \sigma p^2}$$

Solution: e.g., by gradient descent

Mumford-Shah

Region Based Active Contours + Intrinsic Smoothing (more stable). Minimize a functional of the form

$$E_f(\Gamma, g) = \alpha \int_\Gamma ds + \underbrace{\beta \int_{\Omega\setminus\Gamma} h(|\nabla f|)dx}_{\text{region based counterpart of } E_2(\Gamma)} + \underbrace{\frac{\mu}{2}\int_\Omega (g-f)^2 dx}_{\text{smoothing, data fit}},$$

where, e.g., $\Omega = [0,1]^2$ and $h(p) = \frac{1}{2}p^2$ (penalizing large gradients).

Solution: e.g. by level sets (better at capturing topological changes)

Level Sets

A **level set** $\{x \in \Omega : \Phi(x) = M\}$, $M \in \mathbb{R}$, is defined by a **level set function** $\Phi : \Omega \to \mathbb{R}$. A curve Γ is defined as the zero level set for an adequate Φ, i.e.,

$$\Gamma = \{x \in \Omega : \Phi(x) = 0\}.$$

Furthermore, we set

$$\Omega^{\pm} = \{x \in \Omega : \pm\Phi(x) > 0\},$$

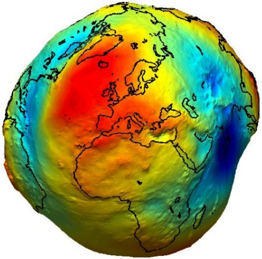
and $g^{\pm}$ are the enhanced images on $\Omega^{\pm}$. The heavyside function is given by $H(z) = 1$, if $z > 0$, and $H(z) = 0$, if $z \leq 0$. Mumford-Shah functional in **Level Set formulation**:

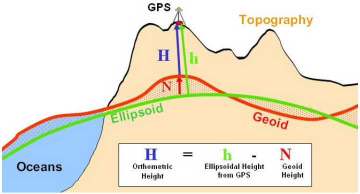
$$\begin{aligned} E_f(\Phi, g^+, g^-) =& \alpha \int_\Omega |\nabla H(\Phi)| dx + \int_\Omega \left(\beta h(|\nabla g^+|) + \frac{\mu}{2}(g^+ - f)^2 \right) H(\Phi) dx \\ &+ \int_\Omega \left(\beta h(|\nabla g^-|) + \frac{\mu}{2}(g^- - f)^2 \right) H(-\Phi) dx \end{aligned}$$

Introduction

- Representation of volume data by surfaces: An isosurface is the level set $L_\Phi(k) = \{x \in \mathbb{R}^3 : \Phi(x) = k\}$ of a function $\Phi : \mathbb{R}^3 \to \mathbb{R}$ or a corresponding data set
- Applications: Visualizing results of Segmentation/Thresholding (2-D or 3-D), Vsualization of Volume data, Geoid
- Problem: How to extract the actual surface from given volume data or a function Φ? E.g. Ray Casting or Marching Cubes

Introduction





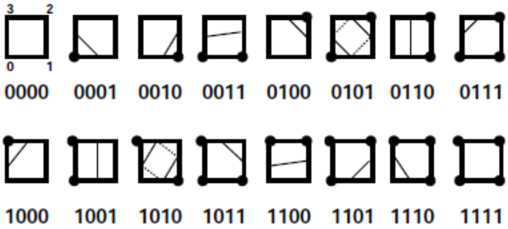
Marching Squares

- 2-D counterpart to marching cubes
- In each grid point decide whether a point is inside or outside an object (by thresholding)
- Find intersecting edges and connect them
- Determine surface normals for shading
- http://undergraduate.csse.uwa.edu.au/units/CITS4241/Handouts/index.html

Marching Squares

Marching Squares

- Let's work out the possibilities:

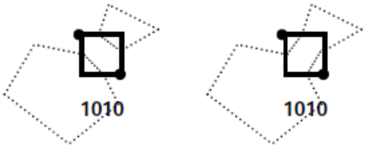


- Point *above* contour (index bit = 1)

Marching Squares

Marching Squares - Ambiguities

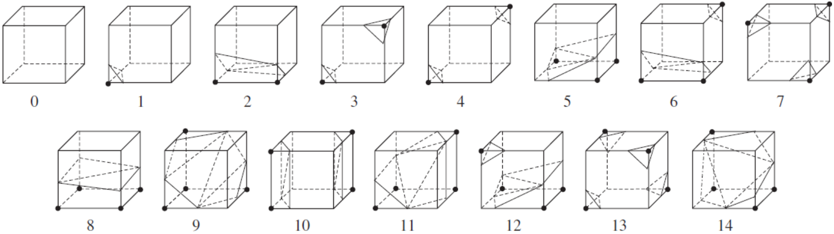
- Two possible contours
- In 2D, choose either one



Break contour Join contour

- Either acceptable
  - Resulting contour lines will be continuous or closed *or* end at dataset boundary

Marching Cubes

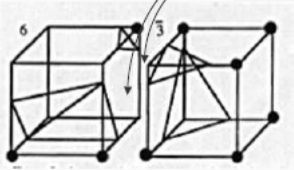


Marching Cubes

But in 3D ...

- We can generate "holes"

This face of the cube
is called the "**ambiguous face**"



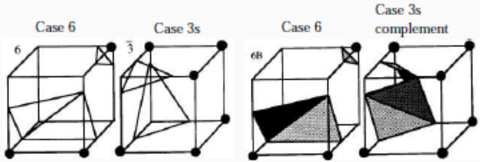
(see also Figure 6.8 of Schroeder et al for another example)

- 6 of the 15 topologies can generate such holes

Marching Cubes

Resolving the ambiguity I (cont.)

- If topology of the current cell is not consistent with the previous neighbour cell then we should consider taking the <u>complementary</u> topology of the current cell (See Figure 6.10 of Schroeder et al for the 6 complementary cases), e.g.

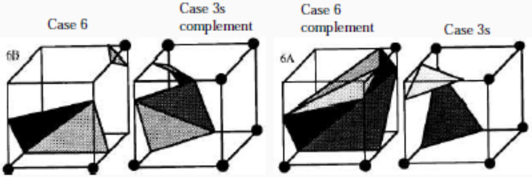


- Note: Case 3s is a symmetric case of Case 3 (see page 5)
- Note: Inconsistency only arises for the 6 'hole-generating' topologies

Marching Cubes

Resolving the ambiguity I (cont.)

- *Using this method, the resultant surface will complete though not necessarily correct.* The correct surface may look like that shown in the right diagram:

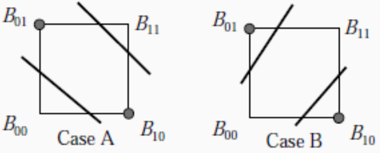


- The degree of incorrectness is likely to be small
- These ambiguous cell faces are not common in medical visualisation applications

Marching Cubes

Resolving the ambiguity II (cont.)

- We know that the contour will be broken into two sections, intersecting all the edges of the square cell. We can find the 4 intersection points by linear interpolation of the function values at the vertices. The intersection may occur in one of the following cases:

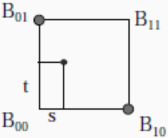


- We need to decide which case actually arises.

Marching Cubes

Resolving the ambiguity II (cont.)

- From bilinear interpolation, we know that for any point p, positioned at (x_p, y_p) inside the square cell, its function value can be interpolated as follows:



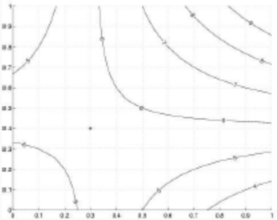
$$f(x_p, y_p) = (1-s)(1-t)\,B_{00} + s\,(1-t)\,B_{10} + (1-s)\,t\,B_{01} + s\,t\,B_{11}$$

- The question we would like to ask is:
  What are the values of *s* and *t* such that $f(x_p, y_p) = \alpha$?

Marching Cubes

Resolving the ambiguity II (cont.)

- **The contour plot of the example on page 13 is shown below:**



All the contours show up as hyperbolae. There are two **asymptotes**, one parallel to the x-axis, another the y-axis.

In the diagram, these two asymptotes intersect at the point $(S_\alpha, T_\alpha) = (0.3, 0.4)$

showing that, for example, if $\alpha = 5$ then the intersection is case B; if $\alpha = 6$ then the intersection is case A

Marching Cubes

Resolving the ambiguity II

- Whether the intersection occurs as case A or case B depends on which *quadrants* the broken contour falls onto. So, we need to find coordinates of the point (S_a, T_a):

$$S_a = \frac{B_{00} - B_{01}}{B_{00} + B_{11} - B_{01} - B_{10}}$$

$$T_a = \frac{B_{00} - B_{10}}{B_{00} + B_{11} - B_{01} - B_{10}}$$

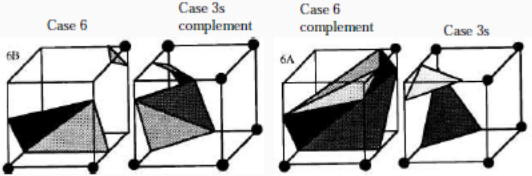
Exercise: use these formulae to verify that $(S_a, T_a) = (0.3, 0.4)$ in our example.

See the values of B_{00}, B_{01}, B_{10}, and B_{11} on Page 13

Marching Cubes

Resolving the ambiguity II

- Having determined where the point (S_a, T_a) is and whether the intersection is case A (or case B), we can decide which one of the following to take:



Time Dependent Isosurfaces

- Allow/Force a surface over time: The Surface $\mathcal{S}_t$ can be defined by time-dependent level sets $L_{\Phi(t,\cdot)}(k)$
- For points $x(t) \in \mathcal{S}_t$, it holds

$$\Phi(t, x(t)) = k \quad \Leftrightarrow \quad \frac{\partial}{\partial t}\Phi(t, x(t)) = -\nabla\Phi(t, x(t)) \cdot \frac{dx(t)}{dt} \tag{1}$$

- The time evolution $\frac{dx}{dt}$ can be imposed by a 'forcing term' $F(x, \Phi, \nabla\Phi, \ldots)$

Surface Morphing

- Morph a surface $\mathcal{S}_i$ into a surface $\mathcal{S}_e$
- Find a good forcing term F to describe the morphing process: Maximize the functional

$$E(\mathcal{S}_t) = \int_{\mathcal{S}_t^{int}} \chi_{\mathcal{S}_e^{int}}(y)dy$$

$$\chi_{\mathcal{S}_e^{int}}(y) \begin{cases} = 0, & y \in \mathcal{S}_e \\ > 0, & y \in \mathcal{S}_e^{int} \\ < 0, & \text{otherwise} \end{cases}$$

- The variational derivative of E is

$$\nabla E(\mathcal{S}_t) = \chi_{\mathcal{S}_e^{int}}(x) N_t(x)$$

Surface Morphing

- Using the steepest decent method, we are lead to the following PDE for the point evolution $x(t) \in \mathcal{S}_t$:

$$\frac{dx(t)}{dt} = \chi_{\mathcal{S}_e^{int}}(x(t))N_t(x(t))$$

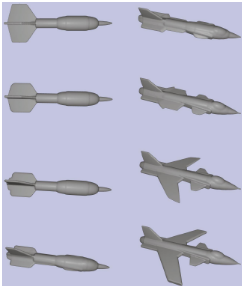
- Inserting the above into Equation (1), we are lead to the following initial value problem that describes the morphing process:

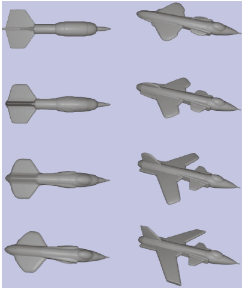
$$\begin{aligned}\frac{\partial}{\partial t}\Phi(t,x) &= -\nabla\Phi(t,x)\cdot\frac{dx(t)}{dt} = -|\Phi(t,x)|\frac{dx(t)}{dt}\cdot N(t,x)\\ &= -|\Phi(t,x)|\chi_{\mathcal{S}_e^{int}}(x)\end{aligned}$$

with an initial value $\Phi(0,\cdot)$ that satisfies $\Phi(0,x) = k$, for all $x \in \mathcal{S}_i$.

Surface Morphing

Examples for two different initial values (taken from David E. Breen and Ross T. Whitaker: A Level-Set Approach for the Metamorphosis of Solid Models, IEEE Trans. Vis. Comp. Graph. 7 (2001))





Introduction

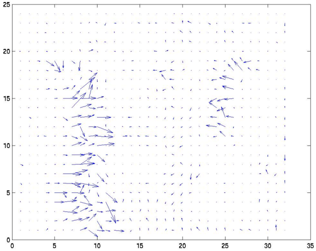
- Given is a sequence of images $f : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$
- Find the motion vector $u : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}^2$ that describes the motion between the succesive images: In other words find trajectories $x(t) \in \mathbb{R}^2$ such that the brightness constancy assumption (BCA) holds true:

$$f(t, x(t)) = \text{const.}$$

  Then the motion vector at the location $x(t)$ in the image $f(t, \cdot)$ is given by $u(t, x(t)) = \frac{d}{dt} x(t)$

Introduction

Taken from http://jonathanmugan.com/GraphicsProject/OpticalFlow/



Optical Flow Equations

Differentiating the BCA with respect to time leads to

$$0 = \frac{d}{dt} f(t, x(t)) = \nabla f(t, x(t)) \cdot \frac{d}{dt} x(t) + \frac{\partial}{\partial t} f(t, x(t)).$$

Therefore, the following equation has to be solved

$$0 = \nabla f(t, x) \cdot u(t, x) + \frac{\partial}{\partial t} f(t, x).$$

Optical Flow Equations

We regard the same problem as before but for a sequence of images $f : \mathbb{R} \times \mathcal{S} \to \mathbb{R}$ on a surface $\mathcal{S} \subset \mathbb{R}^3$. Then

$$0 = \frac{d}{dt} f(t, x(t)) = \nabla_{\mathcal{S}} f(t, x(t)) \cdot \frac{d}{dt} x(t) + \frac{\partial}{\partial t} f(t, x(t))$$

and we have to solve

$$0 = \nabla_{\mathcal{S}} f(t, x) \cdot u(t, x) + \frac{\partial}{\partial t} f(t, x).$$

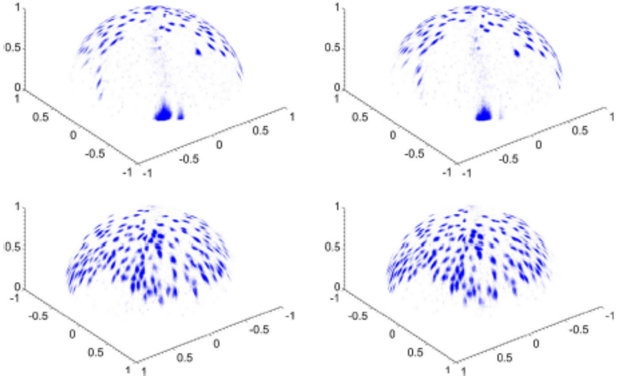
Minimization Problem

The optical flow equations are underdetermined. Therefore one minimizes the following functional to find the motion vector u

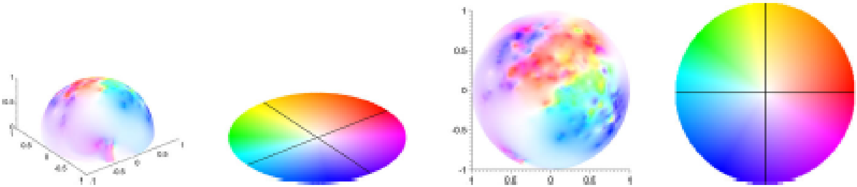
$$\mathcal{F}(u) = \left\| \nabla_{\mathcal{S}} f(t,x) \cdot u(t,x) + \frac{\partial}{\partial t} f(t,x) \right\|_{L^2(\mathbb{R}\times\mathcal{S})} + \mathcal{R}(u),$$

where, e.g., $\mathcal{R}(u) = \|u\|_{H_1(\mathcal{S},\mathcal{T}_{\mathcal{S}})}$.

Surface Example



Surface Example



Optical Flow Equations on Evolving Manifolds

We regard the same problem as before but for a sequence of images $f : \mathbb{R} \times \mathcal{S}_t \to \mathbb{R}$ on a time-dependent surface $\mathcal{S}_t \subset \mathbb{R}^3$. Let $\kappa : \mathbb{R} \times \Omega \to \mathbb{R}^3$ be a smooth map with $\kappa(t, \tilde{x}) \in \mathcal{S}_t$, for all $\tilde{x} \in \Omega \subset \mathbb{R}^2$. Then we set

$$\tilde{f}(t, \tilde{x}) = f(t, \kappa(t, \tilde{x}))$$

and the BCA reads

$$\tilde{f}(t, \tilde{x}(t)) = f(t, \kappa(t, \tilde{x}(t))) = f(t, x(t)) = \text{const.}$$

for a curve $\tilde{x}(t) \in \Omega$. The corresponding differential equation is

$$0 = \frac{d}{dt}\tilde{f}(t, \tilde{x}(t)) = \nabla \tilde{f}(t, \tilde{x}(t)) \cdot \frac{d}{dt}\tilde{x}(t) + \frac{\partial}{\partial t}\tilde{f}(t, \tilde{x}(t))$$

or, in other words

$$0 = \frac{d}{dt}\tilde{f}(t, \tilde{x}) = \nabla \tilde{f}(t, \tilde{x}) \cdot \tilde{u}(\tilde{x}) + \frac{\partial}{\partial t}\tilde{f}(t, \tilde{x}).$$

Afterwards, $\tilde{u}(t, \cdot)$ needs to be mapped back onto $\mathcal{S}_t$.

Optical Flow Equations on Evolving Manifolds

Alternative Representation:

Optical Flow equation intrinsic on the manifold:

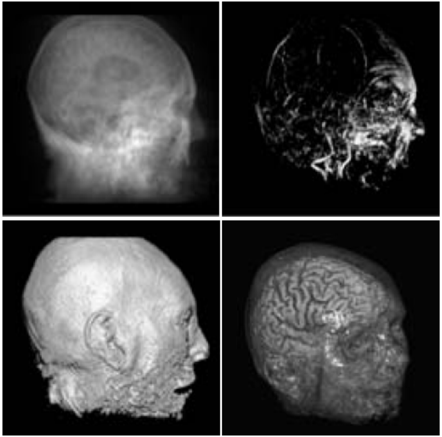
$$0 = \frac{d^{nor}}{dt} f(t,x) + \nabla_{\mathcal{S}_t} f(t,x) \cdot u^{tan}(t,x).$$

Introduction

- Different from an isosurface, the whole volume information is used to visualize a 3-D object
- Rays casted (perspective or parallel) through object onto a projection plane
- Intensities f at grid points need to be interpolated in order to obtain intensities along rays (nearest-neighbour (staircasing), linear (discontinuous derivatives), cubic)
- Use information along the rays to compute values on projection plane (simpest cases: x-ray, maximum intensity projection)

Introdutcion

(a) (b)



(c) (d)

Ray Casting

Image-Order Technique

- **Maximum Intensity Projection (MIP)**: Intensity $I(x, \gamma)$ at some point x on the projection along the ray γ is given as

$$I(x, \gamma) = \max_{t \in [0, L]} f(\gamma(t))$$

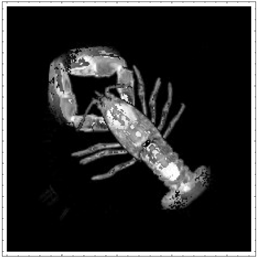
  Local Maximum Intensity Projection (LMIP) takes the first local maximum along the ray above some predefined threshold.

- **X-Ray Projection**: Intensity $I(x, \gamma)$ at some point x on the projection plane along the ray γ is given as

$$I(x, \gamma) = \int_{\gamma} f \, ds$$

Ray Casting

MIP vs. LMIP



Ray Casting

- **Full Volume Rendering** Intensity $I(x, \gamma)$ (depending on wavelength λ) at some point x on the projection along the ray γ is given as

$$I(x, \gamma) = \int_0^L C(s)\mu(s)e^{-\int_0^s \mu(t)dt} ds$$

where $\mu(s)$ is the mass density (or light extinction value) at the point $\gamma(s)$ (relates to $f(\gamma(s))$ and the coefficient $C(s) = E(s) + R(s)$ defines the emmision and reflection properties at the location $\gamma(s)$.

Ray Casting

Numerical Realization: Substitue Integration by Riemann sum:

$$
\begin{aligned}
I(x,\gamma) &= \sum_{i=0}^{\frac{L}{\Delta s}-1} C(i\Delta s)\mu(i\Delta s)\Delta s \prod_{j=0}^{i-1} e^{-\mu(j\Delta s)\Delta s} \\
&\approx \sum_{i=0}^{\frac{L}{\Delta s}-1} C(i\Delta s)\alpha(i\Delta s) \prod_{j=0}^{i-1} (1-\alpha(j\Delta s))
\end{aligned}
$$

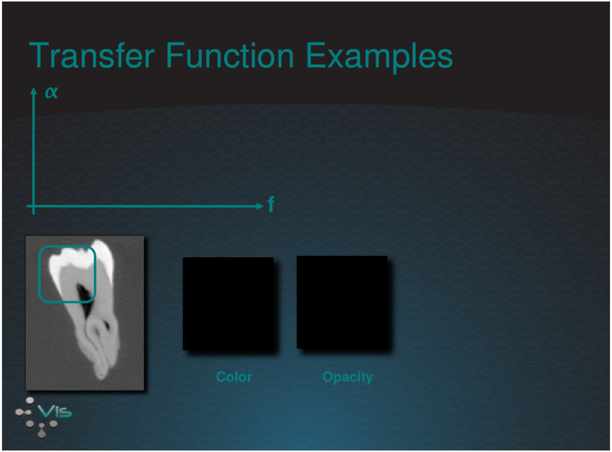
where α is the opacity. **Front-to-back** compositing formula, $k = 1, \ldots, \frac{L}{\Delta s} - 1$:

$$
\begin{aligned}
\bar{I}_{k+1} &= \bar{I}_k + C((k+1)\Delta s)\alpha((k+1)\Delta s)(1-\bar{\alpha}_k) \\
\bar{\alpha}_k &= \alpha(k\Delta s)(1-\bar{\alpha}_{k-1}) + \bar{\alpha}_{k-1}
\end{aligned}
$$

The design of the **transfer functions** C, α is crucial.

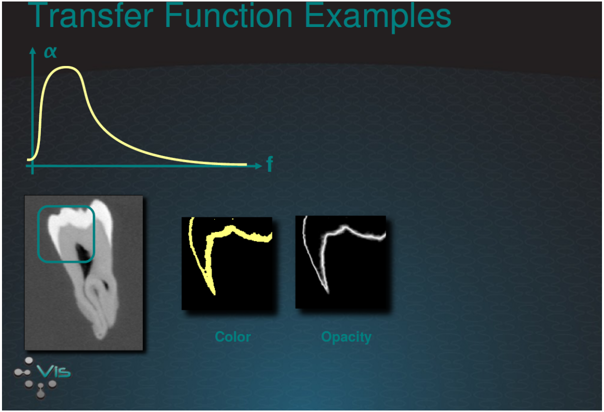
Ray Casting

Slide taken from http://www-pequan.lip6.fr/~tierny/stuff/teaching/tierny_intro_vol_rend09.pdf



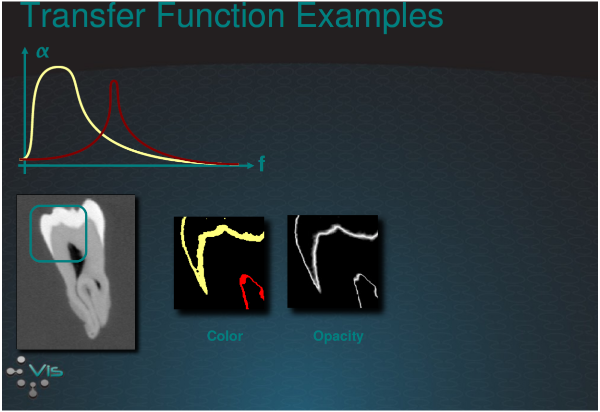
Ray Casting

Slide taken from http://www-pequan.lip6.fr/~tierny/stuff/teaching/tierny_intro_vol_rend09.pdf



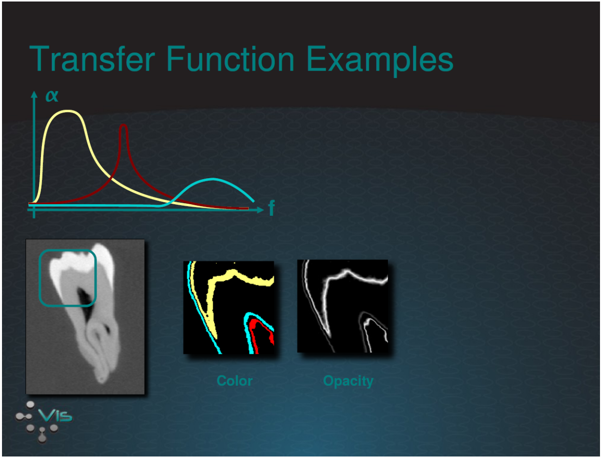
Ray Casting

Slide taken from http://www-pequan.lip6.fr/~tierny/stuff/teaching/tierny_intro_vol_rend09.pdf



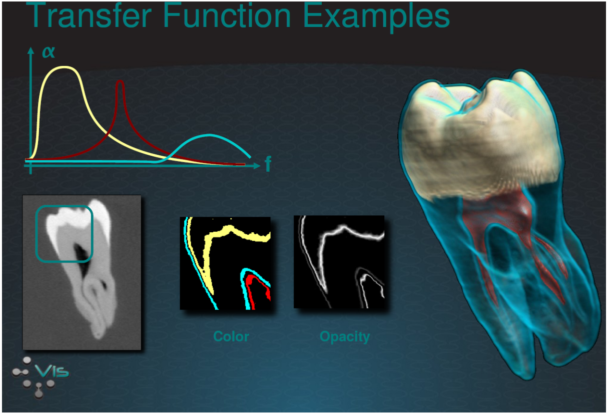
Ray Casting

Slide taken from http://www-pequan.lip6.fr/∼tierny/stuff/teaching/tierny_intro_vol_rend09.pdf



Ray Casting

Slide taken from http://www-pequan.lip6.fr/~tierny/stuff/teaching/tierny_intro_vol_rend09.pdf



Ray Casting

The **reflection coefficient** $R(s)$ can be modeled by the standard illumination equation (cf. J. Foley, A. Dam, S. Feiner, and J. Hughes. Computer Graphics: Principles and Practice, 1996):

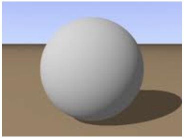
$$R(s) = k_a C_a + k_d C_l C_0(s)(N(s) \cdot L(s)) + k_s C_l (N(s) \cdot H(s))^p$$

where k_a, C_a are ambient material and color coefficients, C_l color of the light source, $C_0(s)$ the color of the object at location $\gamma(s)$, k_d the diffuse material coefficient, $N(s)$, $L(s)$, $H(s)$ the normal vector, light direction vector, and halfvector at location $\gamma(s)$, respectively, k_s the spherical material coefficient, and p is the Phong coefficient.

The design of R is crucial.

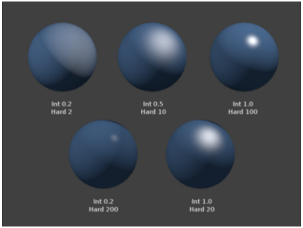
Ray Casting

Diffuse Reflection with and without ambient light



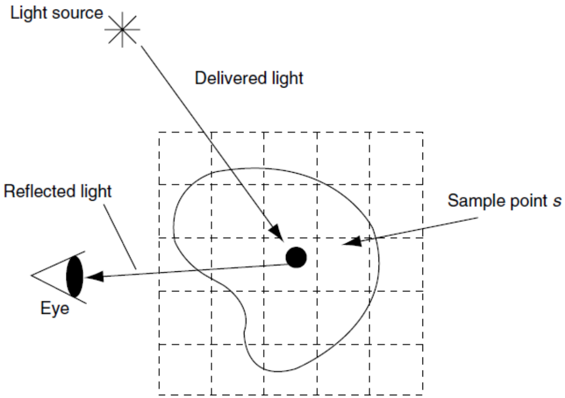
Ray Casting

Specular shading



Ray Casting

Previous model does not include attenuation of light from source to $\gamma(s)$



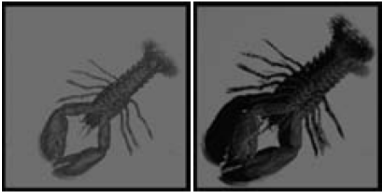
Ray Casting

Volumetric Shadows: Make C_l dependent on $\gamma(s)$

$$C_l(s) = \tilde{C}_l e^{-\int_s^D \mu(t)dt}$$

where $\mu(t)$ is the mass density at point $\tilde{\gamma}(t)$ along the ray connecting $\gamma(s)$ to the light source at distance D.

Rendering without vs. rendering with shadows



Ray Casting

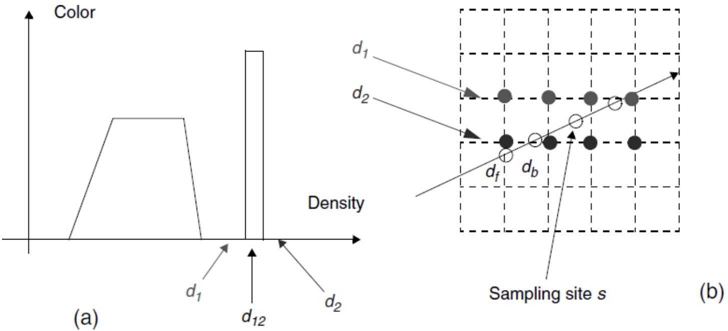
The previous model is called **pre-classification** since α, C are mapped to voxels before interpolation. In **post-classification**, first the volume intensities f are interpolated along the ray and then they are mapped to α, C:

$$I(x,\gamma) \approx \sum_{i=0}^{\frac{L}{\Delta s}-1} C(f(i\Delta s), \nabla f(i\Delta s))\alpha(f(i\Delta s)) \prod_{j=0}^{i-1}(1-\alpha(f(j\Delta s)))$$

Post-classification is typically better at capturing high-frequency details.

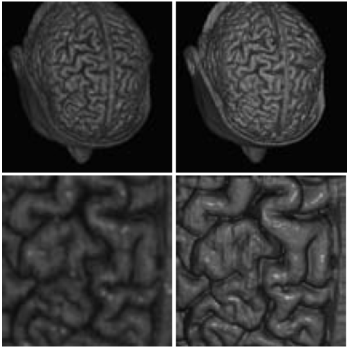
Ray Casting

Transfer function aliasing in pre-classified rendering



Ray Casting

Pre-classified vs. post-classified rendering



Ray Casting

Multiple Scattering: For clouds, e.g., the pervious single-scattering (low albedo) scenario is not correct. Scattering after the first initial scattering has to be taken into account:

$$C(s)\mu(s) = \int_{\mathbb{S}^2} W(\gamma(s), \xi) I(\gamma(s), \xi) dS(\xi)$$

where ξ denotes the direction of incoming light at point $\gamma(s)$. Then:

$$I(x, \gamma) = \int_0^L \left(\int_{\mathbb{S}^2} W(\gamma(s), \xi) I(\gamma(s), \xi) dS(\xi) \right) e^{-\int_0^s \mu(t) dt} ds$$

This is an integral equation that needs to be solved.