# A Model for Volume Lighting and Modeling

Presentation by

Abon Chaudhuri
Teng-Yok Lee

# Outline

- Motivation
- Review of direct volume rendering
- Theoretical contribution
- Implementation
- Results & Discussion

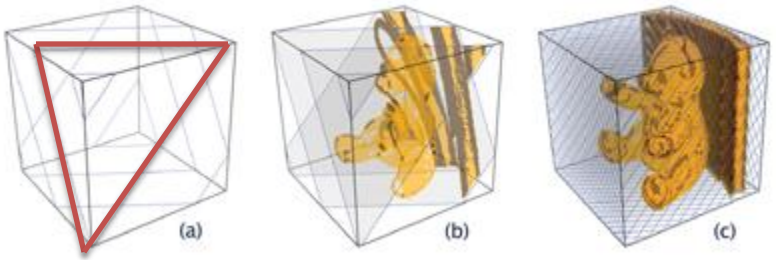
# Motivation

- Gradient-based surface shading has limitations
  - Normal is not well-defined in uniform regions
  - Low-magnitude gradients are affected by noise
- Shadow computation is expensive and hence, hinders interactive rendering
- This paper tries to achieve
  - a more sophisticated shading model for direct rendering of translucent volumes while retaining interactivity

# Outline

- Motivation
- Review of direct volume rendering
- Theoretical contribution
- Implementation
- Results & Discussion

# Direct volume rendering: Review




**1. Initialize**
1.1 Load and Process Data
1.2 Download Textures
1.3 Create Shaders

**2. Update**
2.1 Update Proxy Geometry
2.2 Update Textures

**3. Draw**
3.1 Set Up Rendering State
3.2 Draw Proxy Geometry
3.3 Restore Rendering State

**User Input**

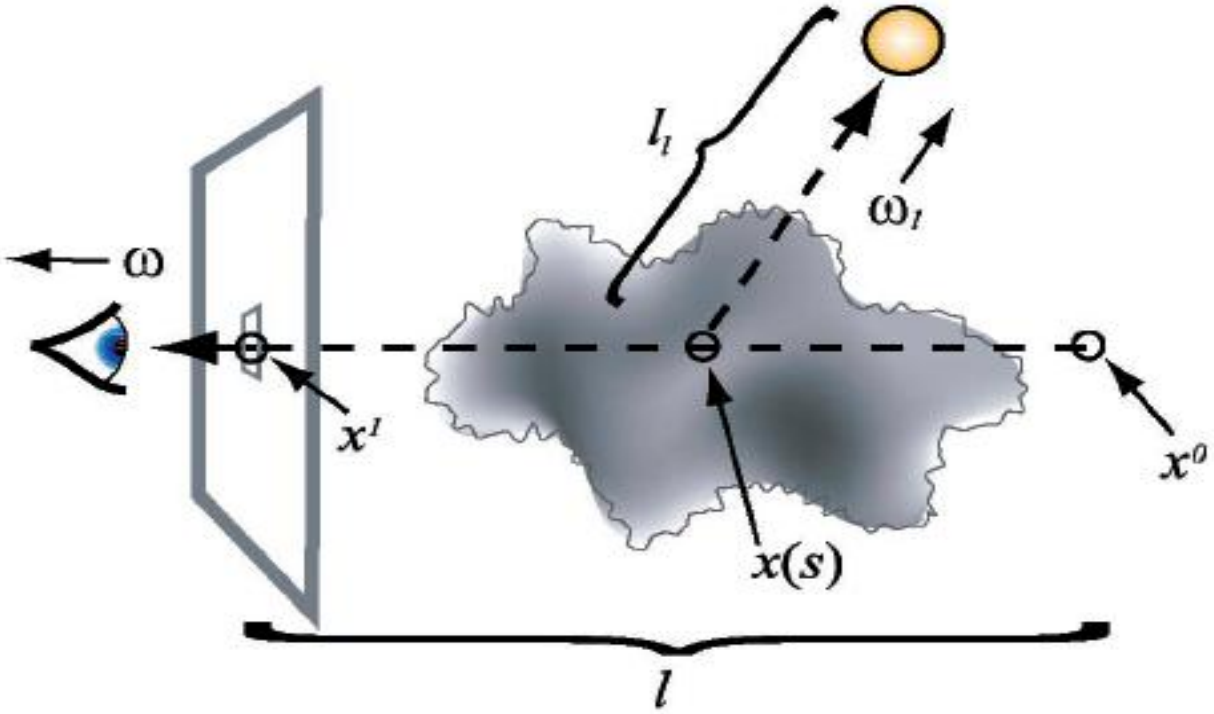
Viewing Parameters
Sampling Rate
Rendering Mode
Transfer Function

Image courtesy:
GPU Gems 1: Chapter 39

# Outline

- Motivation
- Review of direct volume rendering
- Theoretical contribution
- Implementation
- Results & Discussion

# Shading for direct volume rendering




$$\mathbf{x}(s) = \mathbf{x}_0 + s\vec{\omega}$$

Parametric form of ray

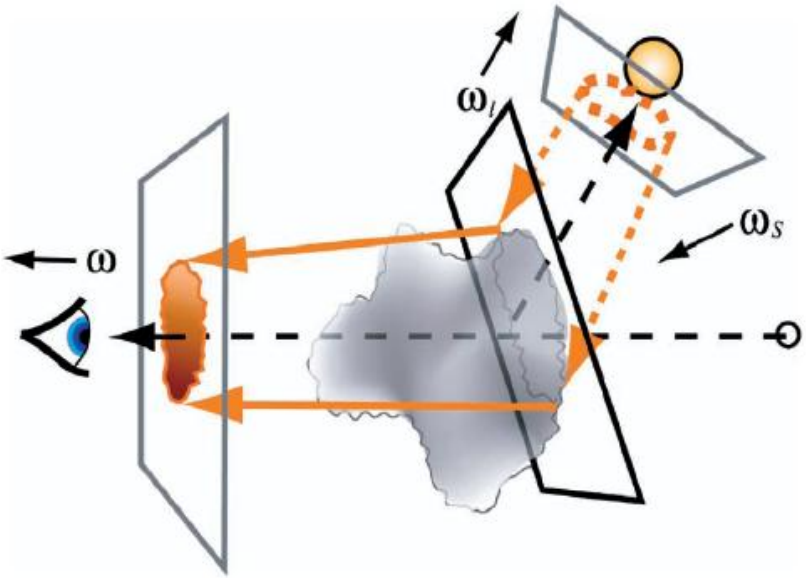
**R:** surface reflectivity
**T:** attenuation between two points
**f:** Blinn-Phong shading based on local gradient
**$L_l$:** Intensity of light source

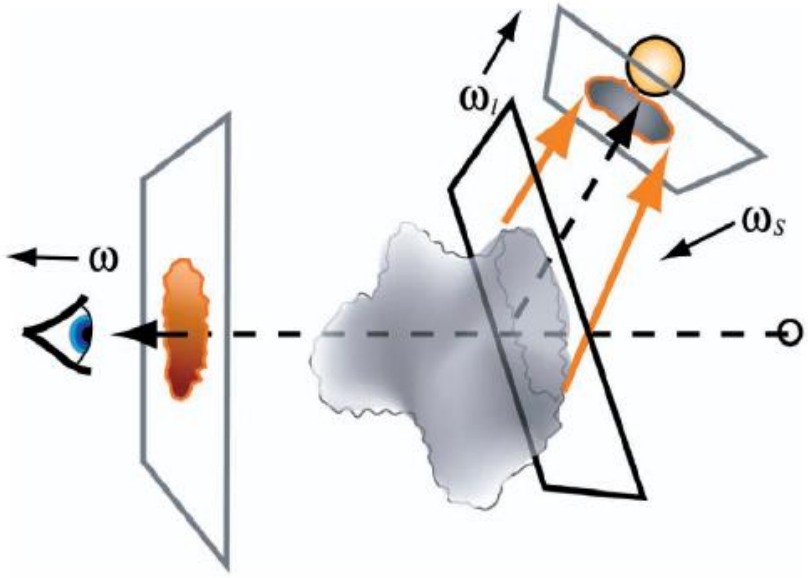
$$L(\mathbf{x}_1, \vec{\omega}) = T(0,l)L(\mathbf{x}_0, \vec{\omega}) + \int_0^l T(s,l)R(\mathbf{x}(s))f_s(\mathbf{x}(s))L_l ds$$

Where:

$$T(s,l) = \exp\left(-\int_s^l \tau(s')ds'\right)$$

# Shading for direct volume rendering with shadow




**First step:**
1. Render for eye view
2. Sample light buffer

**Second step:**
1. Render for eye view
2. Sample light buffer

**Half-angle slicing** of the volume is required to ensure front-to-back rendering from the light source

# Shading for direct volume rendering with shadow

$$L(\mathbf{x}_1, \vec{\omega}) = T(0,l)L(\mathbf{x}_0, \vec{\omega}) + \int_0^l T(s,l)R(\mathbf{x}(s))f_s(\mathbf{x}(s))T_l(s,l_l)L_l\,ds$$

Where:

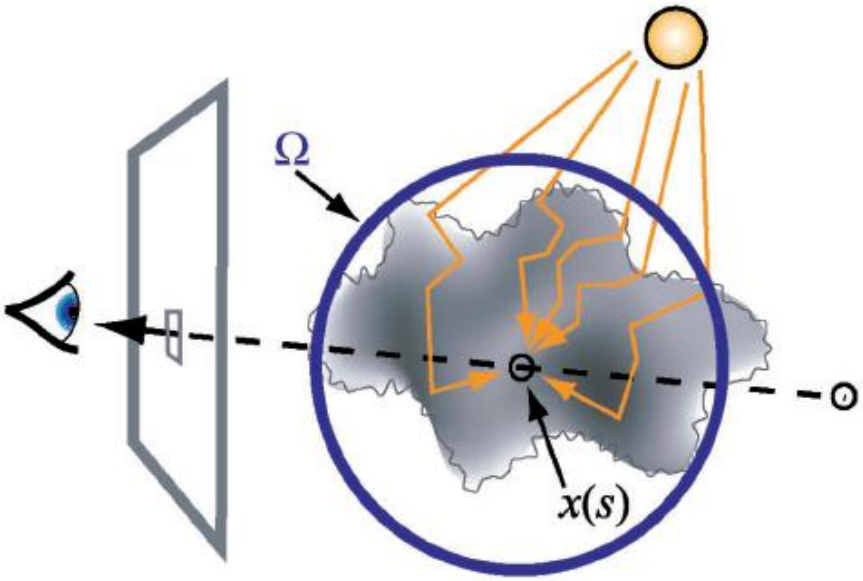
$$T_l(t,t') = exp\left(-\int_0^{t'} \tau(\mathbf{x}(t) + \vec{\omega}_l s)ds\right)$$

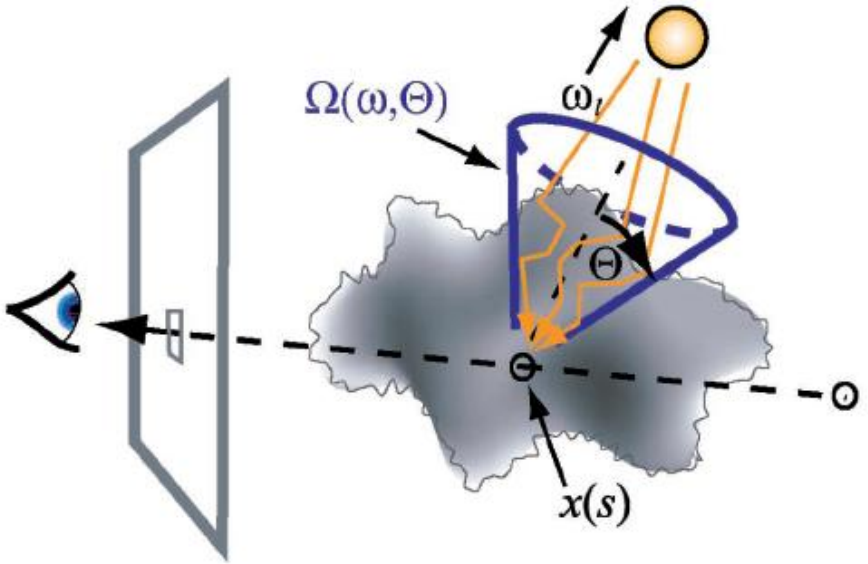
New term for attenuation of light coming from light source

Cons:

- Computation of shadow volume is expensive and consumes high memory
- Does not account for translucency in the volume

# Shading for volume rendering with indirect light contribution




Scattering is the reason for indirect lights coming from all possible directions

- Only a section of all possible directions Is considered for implementation
- Model does not capture backscattering

# Shading for volume rendering with indirect light contribution

$$L(\mathbf{x}_1, \omega) = T(0, l)L(\mathbf{x}_0, \omega) + \int_0^l T(0, s) * C(s) * L_l(s)ds$$

Blurred indirect light contribution

Where:

$$C(s) = E(s)((1 - S(s)) + f_s(s)S(s))$$

$$L_l(s) = L_l * \exp\left(-\int_s^{lt} \tau(x)dx\right) * P(\Theta) +$$

**P:** Phase function

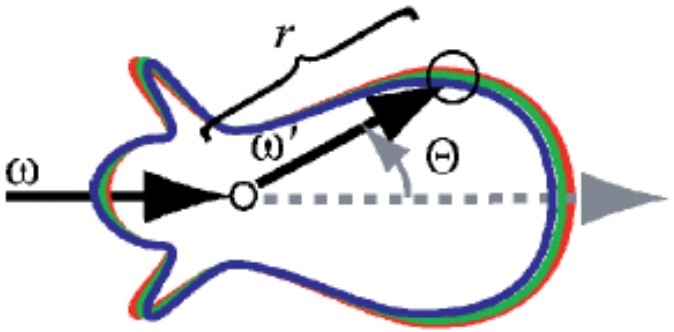
$$L_l * exp\left(-\int_s^{lt} \tau_i(x)dx\right)\mathbf{Blur}(\theta)$$

Phase function driven Direct attenuation

Indirect attenuation

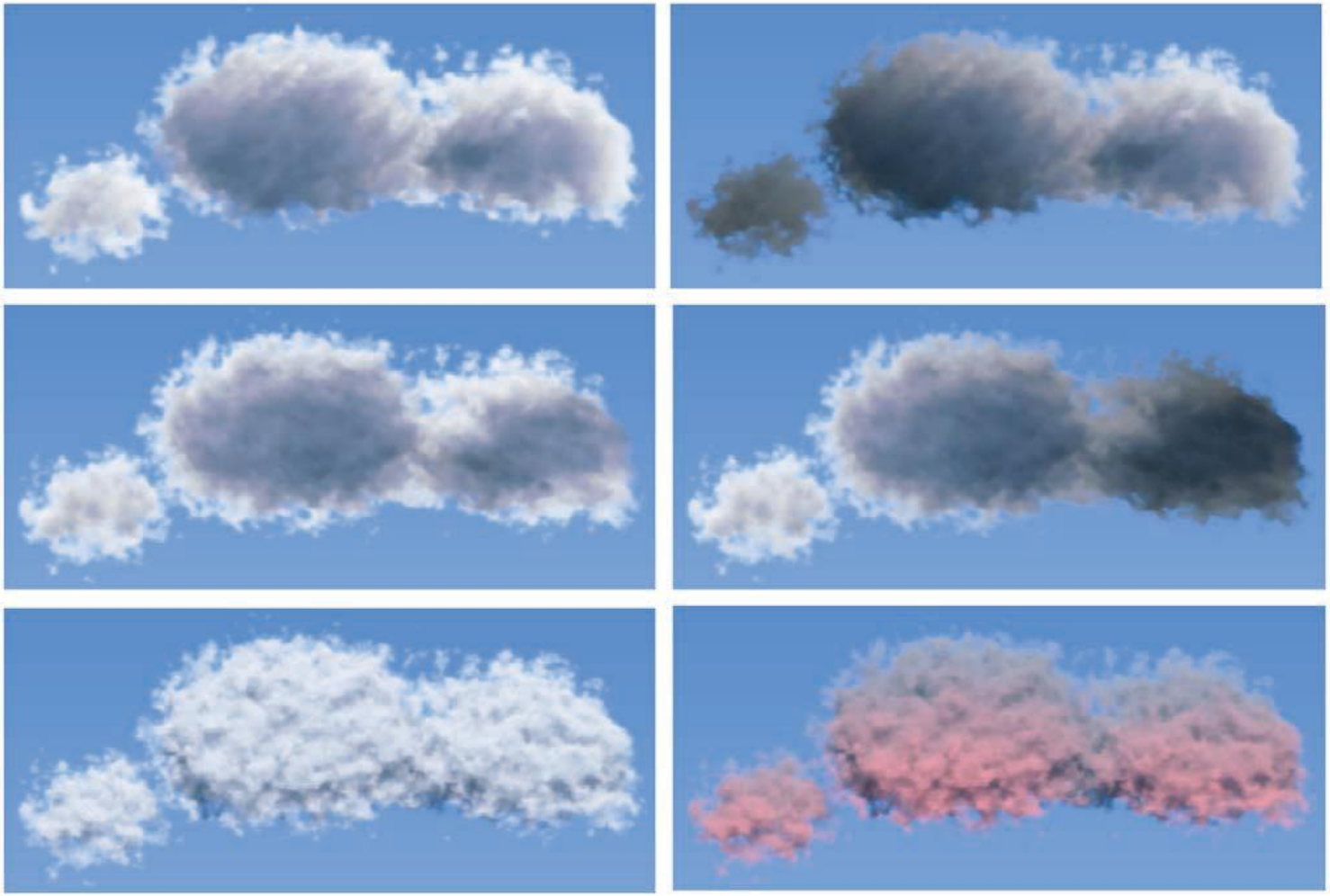
# Phase Function

- Distribution of outgoing light directions due to scattering over the entire sphere of directions, given an incoming light direction
- Dependent on the cosine of the angle between the incoming and outgoing directions




Example of a symmetric phase function

# Effect of phase function

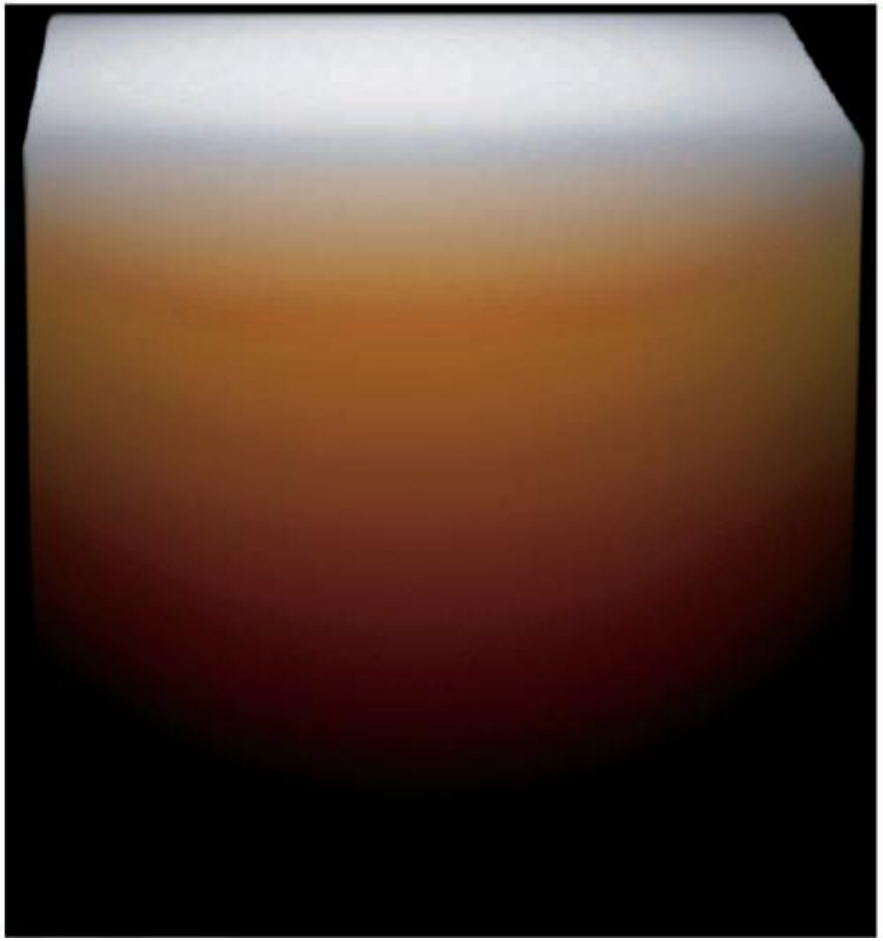


# Indirect Attenuation

- Actual phenomenon
  - Diffusion of light through the volume which is a translucent medium
  - Light moves farther through the volume than it would without indirect diffusion
- How to achieve?
  - Reduce the attenuation along the light direction by randomly blurring the light in the local neighborhood

# Effect of indirect attenuation

Photograph



Volume rendering

# Outline

- Motivation
- Review of direct volume rendering
- Theoretical contribution
- Implementation
- Results & Discussion

# Implementation

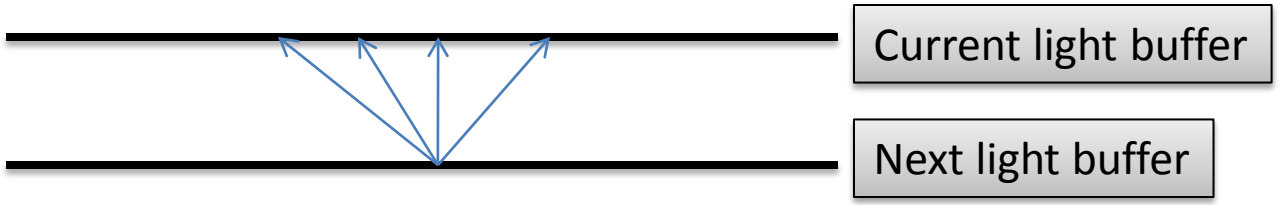
- Set-up:
  - Texture coordinates needed while slicing the volume
    - 3D texture co-ordinates for volume data
    - 2D texture co-ordinates  for projection of vertices on to light buffer
    - 3D texture coordinate to represent view direction from eye to vertex
    - 3D texture coordinate to represent view direction from light to vertex (if light is at finite distance)

# Implementation

- Pass 1: Observer Update
  - Render a slice from observer's point of view
    - Render target - eye buffer
    - Bound texture – light buffer
  - Total light intensity at a fragment
    - **$I' = (L_i + S().L_d)\ I_0$**
    - I': total RGB intensity at current sample
    - $I_0$: original light color
    - $L_i$: indirect RGB light intensity from current light buffer
    - $L_d$: direct achromatic light intensity from current light buffer
    - S: Phase function, if applicable

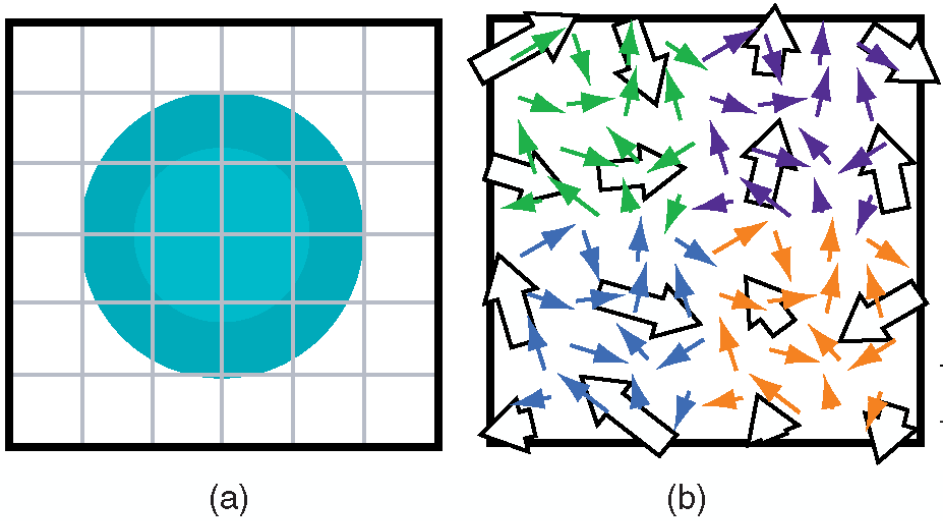
# Implementation

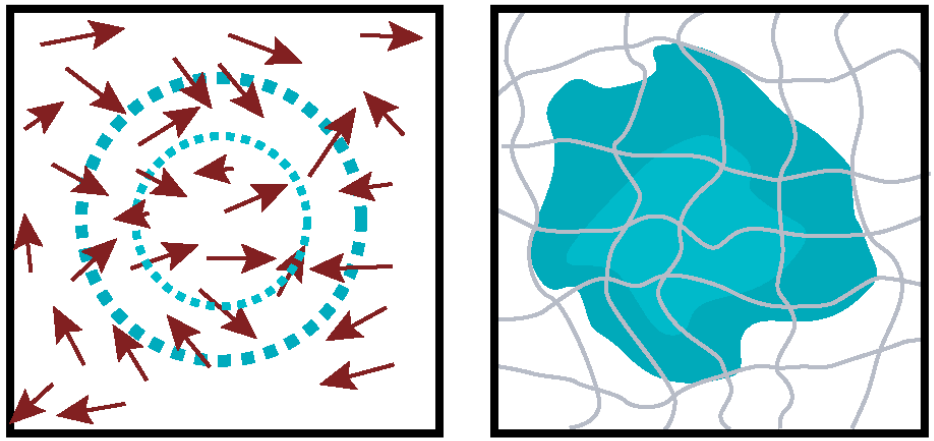
- Pass 2: Light Update
  - Same slice is rendered from light's point of view
    - Render target – next light buffer
    - Bound texture – current light buffer
  - To imitate scattering, current sample location is perturbed to obtain at least 4 neighboring sample locations
  - Next light buffer replaces the current one
  - The technique is known as ***ping pong blending***



# Volume Perturbation

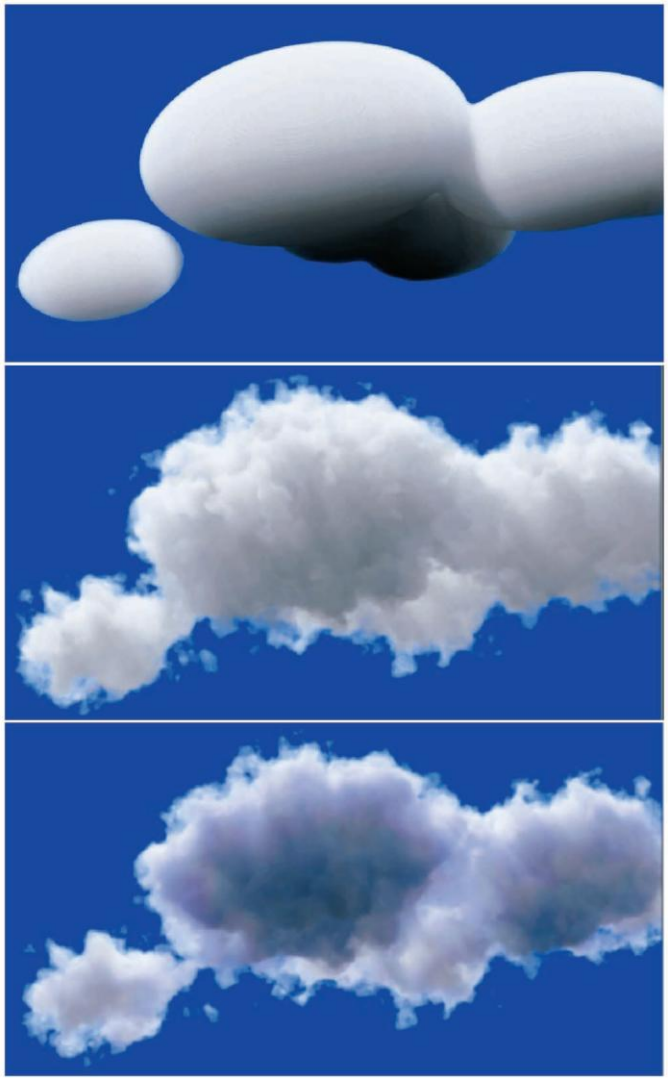
Input
Light buffer



Perturbed
Light buffer

# Volume Perturbation Effect

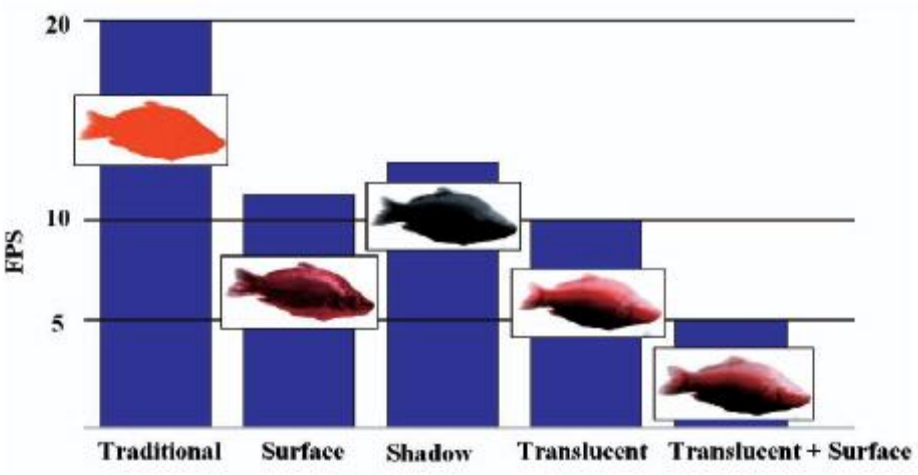


# Outline

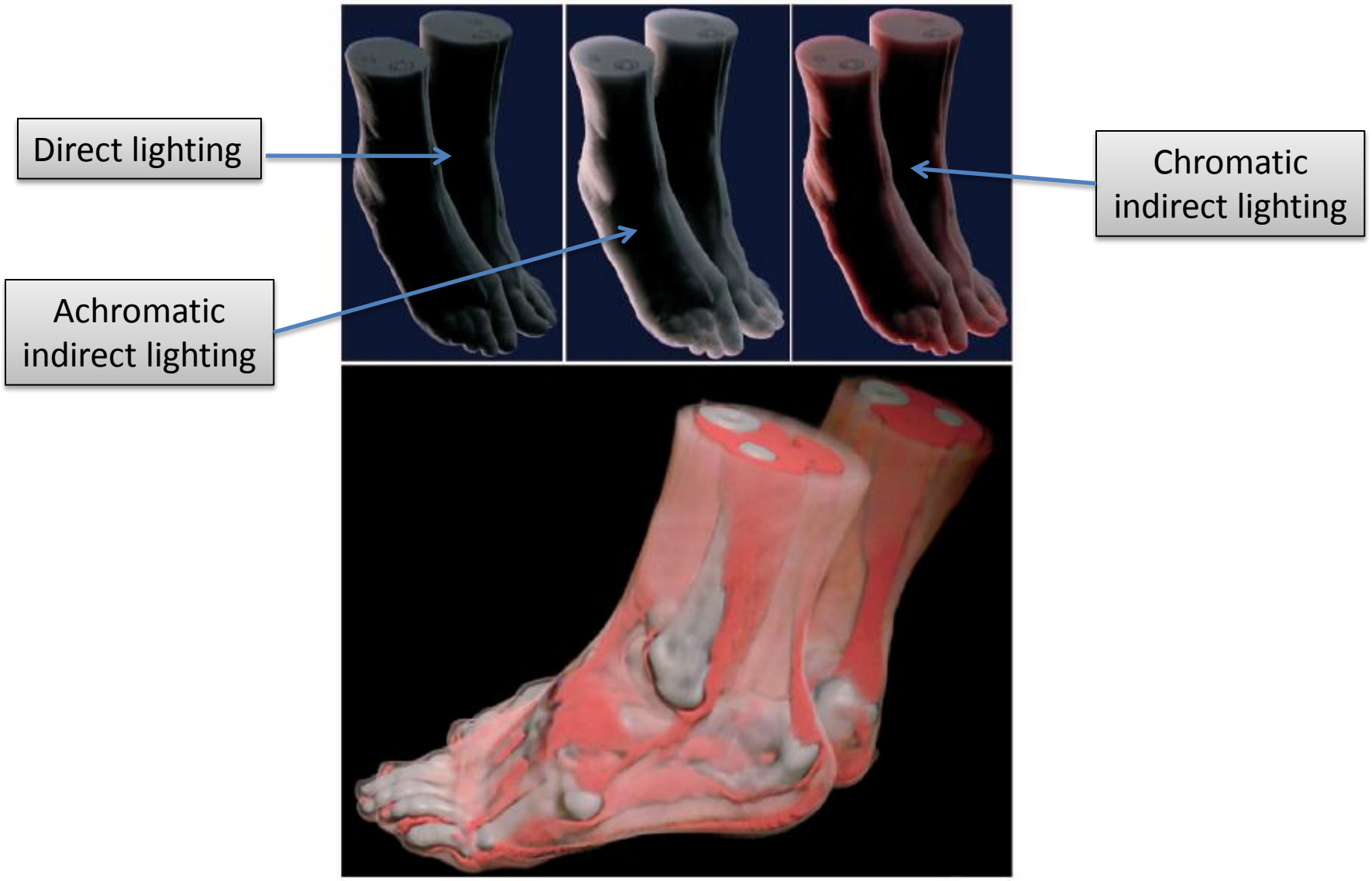
- Motivation
- Review of direct volume rendering
- Theoretical contribution
- Implementation
- Results & Discussion

# Results

- Author's implementation on
  - NVIDIA GeForce 3
  - ATI/Radeon 8500/9500
- Render-to-texture extension used as primary technique
- Frame rate: 50-60% slower than volume rendering with no shading at all
- Detailed report of timing not present in the paper though

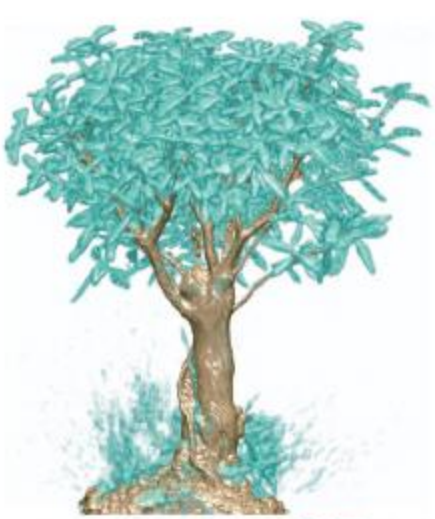


# Results


Direct lighting
Achromatic indirect lighting
Chromatic indirect lighting

# Results

Surface shading



Direct lighting

Direct + Indirect lighting

Direct + Indirect Lighting,
Surface shaded leaves

# Discussion

- Trade-off between accuracy and speed
  - More accurate software-based approaches cannot provide interactivity
- This method computes and stores light transport in image spaces
  - Compared to previous methods which stored light information in 3D volume textures
- If you are interested in GPU-based shading for ray casting, this VIS'08 tutorial may be useful:
  - http://www.voreen.org/240-Vis08-Tutorial.html