SCREEN-SPACE EFFECTS

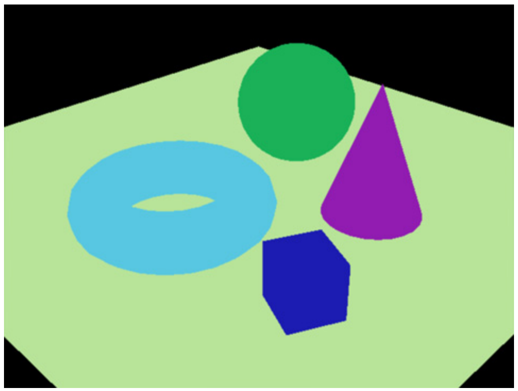
Jaroslav Křivánek

Charles University in Prague

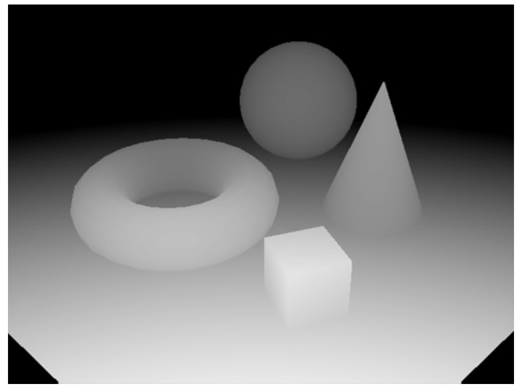
Deferred shading

Deferred shading

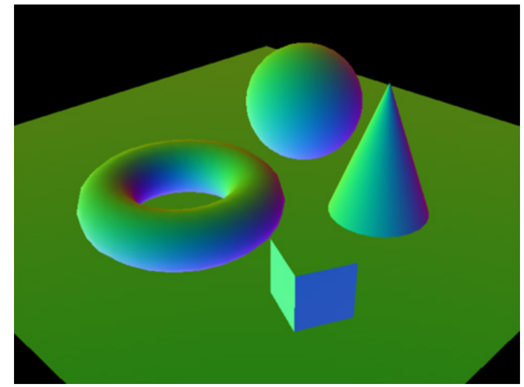
diffuse color G-buffer

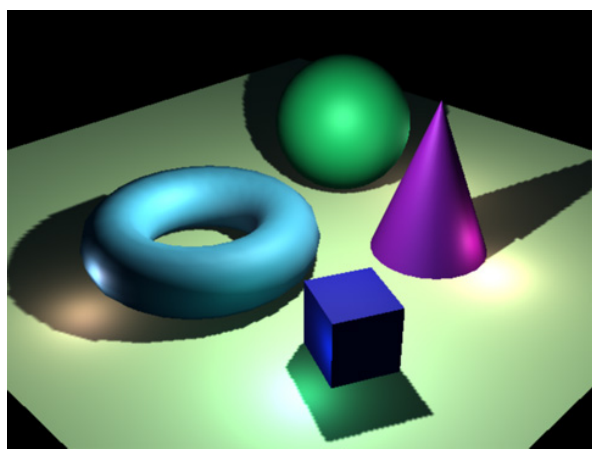


z-buffer



surface normal G-buffer





final rendering

Deferred shading

- Geometry rendered first into geometry buffers (G-buffers)
  - Depth (position relative to camera)
  - Normal
  - Material properties (diffuse texture, roughness etc.)

- Shading is completed in another pass
  - Purely screen-space operation
  - Multi-pass shading does not require multiple passes over geometry
  - No hidden fragments are ever shaded

Deferred shading

- Provides a framework for many screen space techniques

- Pros
  - Reduces cost of operations by implementing effects dependent only on screen size not scene complexity

- Cons
  - May results in higher memory usage and bandwidth
  - Cannot handle transparency
  - HW anti-aliasing does not produce correct results anymore

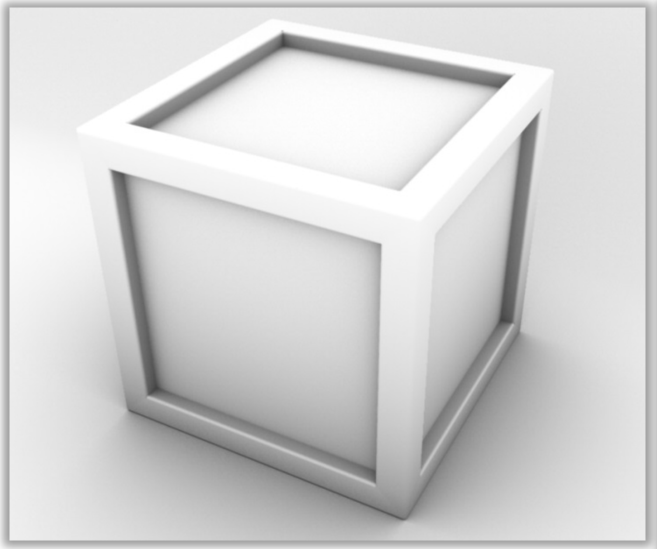
Deferred shading – Use in games

Deferred shading – Further reading

- Wikipedia "**Deferred shading**"
  - https://en.wikipedia.org/wiki/Deferred_shading
- Brent Owens "**Forward Rendering vs. Deferred Rendering**"
  - http://gamedevelopment.tutsplus.com/articles/forward-rendering-vs-deferred-rendering--gamedev-12342
- Oles Shishkovtsov "**Deferred Shading in *S.T.A.L.K.E.R.*"**, GPUGems2, NVIDIA
  - http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter09.html
- Rusty Koonce "**Deferred shading in Tabula Rasa**", GPUGems3, NVIDIA
  - http://http.developer.nvidia.com/GPUGems3/gpugems3_ch19.html

Screen-space ambient occlusion (SSAO)

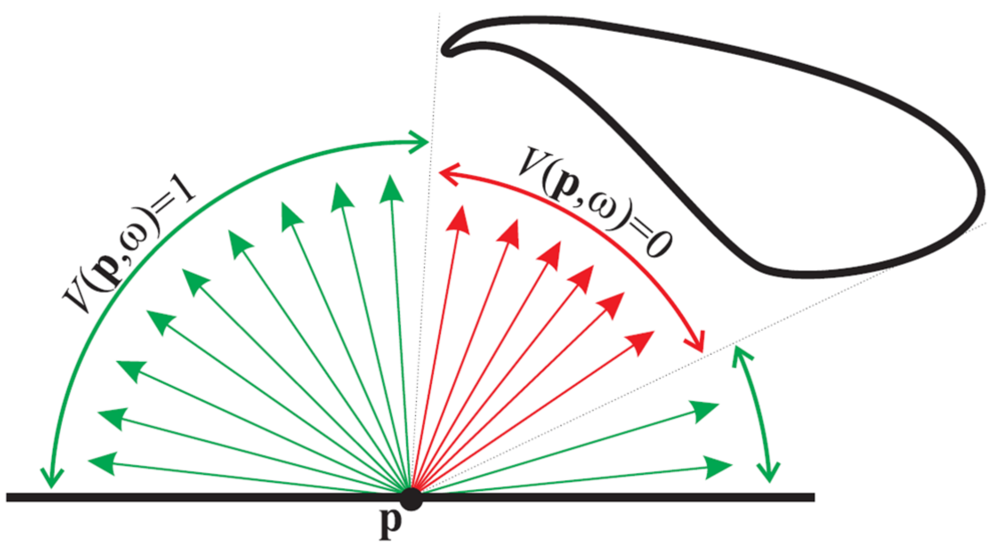
What is ambient occlusion?



What is ambient occlusion?

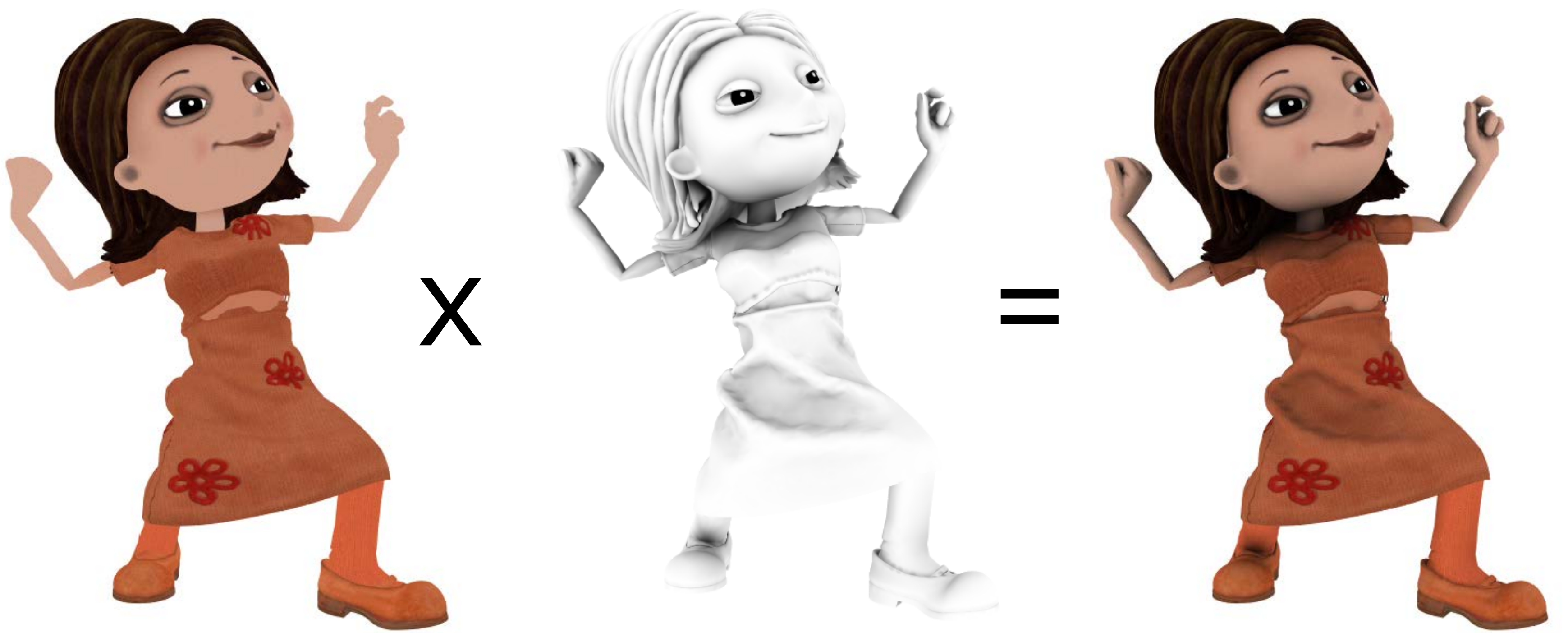
- Degree of occlusion of the environment as visible from a given point

$$A(\mathbf{p}) = \frac{1}{\pi} \int_{H^+} V(\mathbf{p}, \omega) \cos\theta \, d\omega$$



Why is ambient occlusion useful?

- Improves the impression of the object shape
- Cheap way to approximate GI



Why is ambient occlusion useful?

- Gives perceptual cues of depth, curvature and spatial proximity

No AO

With AO

Why is ambient occlusion useful?

Original model With ambient occlusion Extracted ambient occlusion map

AO in real-time: screen-space AO (SSAO)

- Developed by Crytek for Crysis 1 (2007)
- Used in tens of titles since then
- Not very accurate but provides a convincing effect

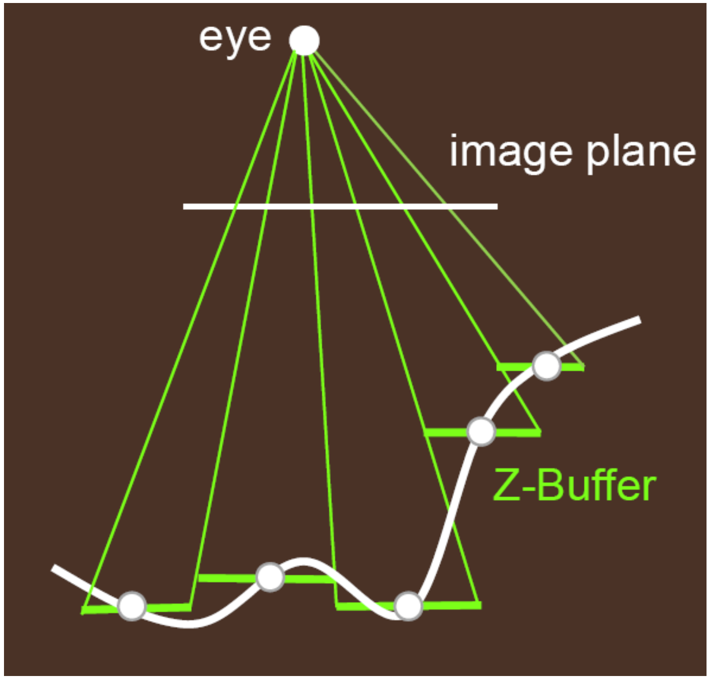
How does SSAO work?

- Depth-buffer image processing
- Compare depth of current pixel with the depth of neighbors

- Optimizations
  - Use randomly rotated kernel for each pixel
  - Remove noise by post-filtering the AO result

Depth buffer = height field

- Approximation of scene geometry as seen from the current camera location

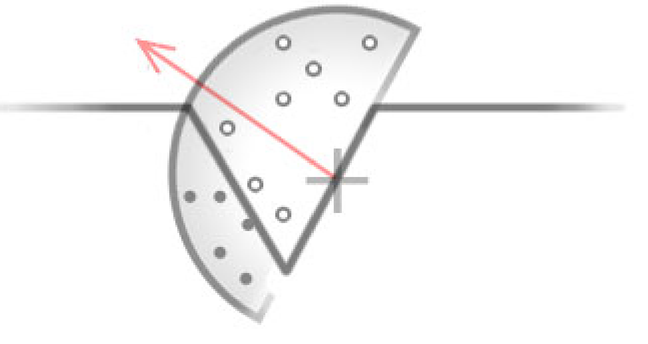


Sampling of neighbor values

- Generate samples in a sphere or hemisphere around surface normal

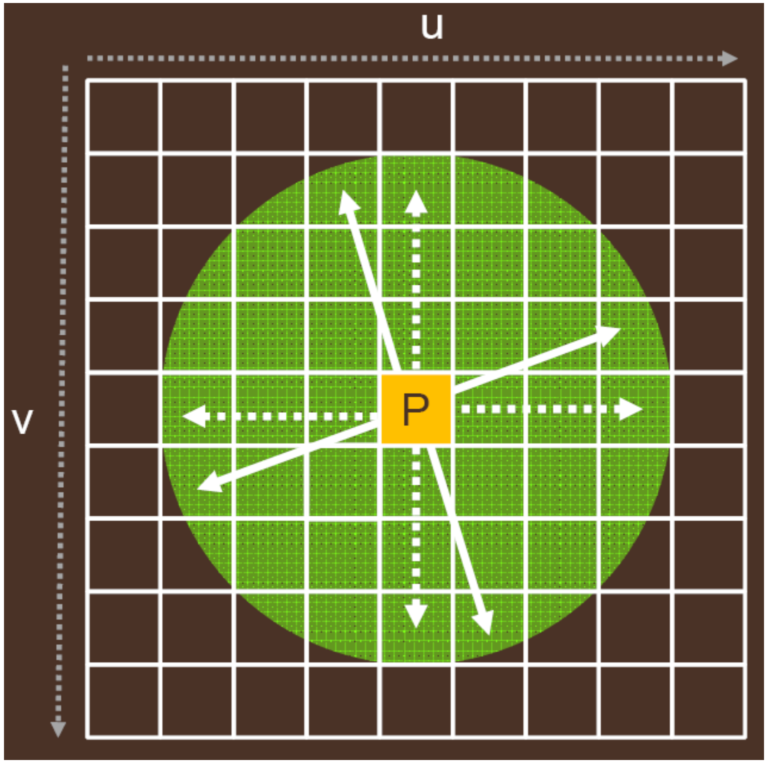
- For each sample, find out the depth buffer coords

- Compare depth of the sample with depth buffer value
  - If sample closer, no occlusion
  - If sample farther, occlusion



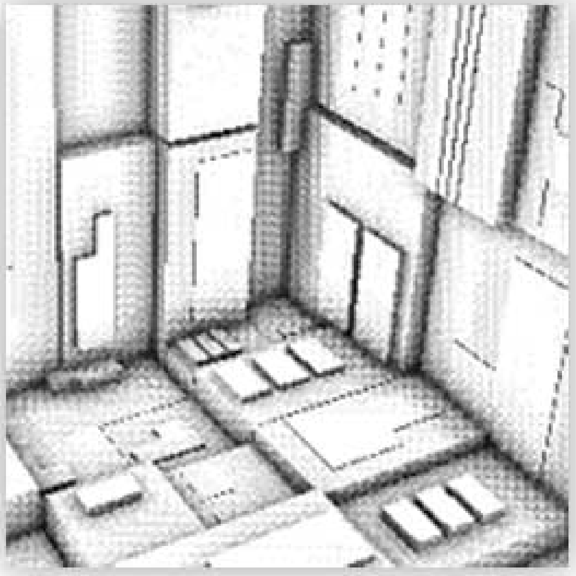
Random kernel rotation

- Randomization = classic trick to convert artifacts into incoherent image noise
- In SSAO: Randomly rotate the sample set around the surface normal

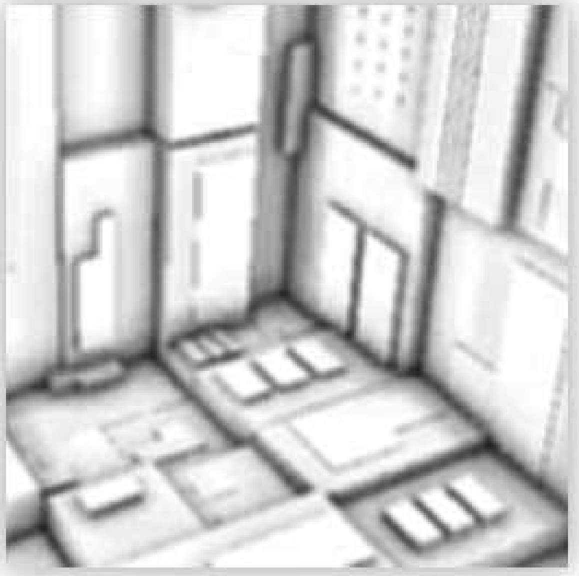


Noise post-filtering

- Remove noise by post-filtering the AO result



pre-blur



post-blur

Half-Resolution AO
6x6 samples / AO pixel
No Blur

AO = 3.5 ms @ 800x600
On GeForce GTX 280

Half-Resolution AO
6x6 samples / AO pixel
15x15 Blur

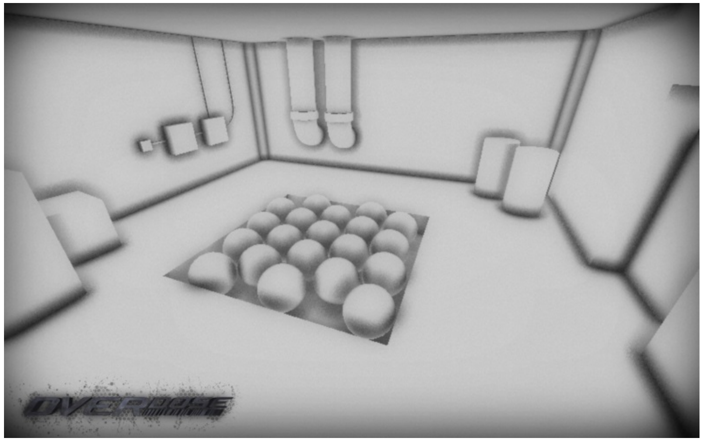
AO = 3.5 ms @ 800x600
Blur = 2.5 ms @ 1600x1200
On GeForce GTX 280

SSAO advantages

- Independent from scene complexity.
- Works with dynamic scenes.
- Works in the same consistent way for every pixel.
- Can be executed completely on the GPU.
- Easily integrated into any modern graphics pipeline.
- Complements the real-time GI techniques (e.g. light propagation volumes) which can only simulate global, ambient light

SSAO disadvantages

- Rather **localized** effect
- In many cases **view-dependent**, as it is dependent on adjacent texel depths.
- Hard to correctly smooth/blur out the noise without interfering with depth discontinuities, such as object edges (the occlusion should not "bleed" onto objects).



SSAO is rather localized

Further references on SSAO

- Additional slides with more technical details
  - http://developer.download.nvidia.com/presentations/2008/SIGGRAPH/HBAO_SIG08b.pdf
- Implementation & video
  - http://john-chapman-graphics.blogspot.cz/2013/01/ssao-tutorial.html

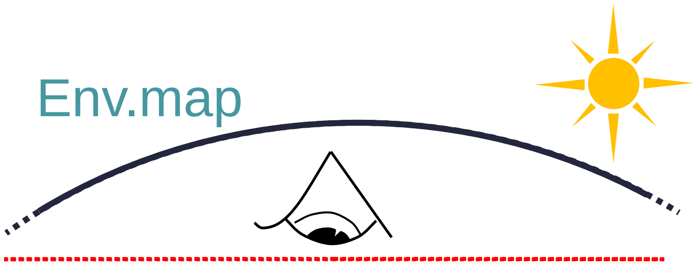
Scree-space GI

Screen-space global illumination

- Ritschel et al. “Approximating Dynamic Global Illumination in Image Space”
  - http://people.mpi-inf.mpg.de/~ritschel/SSDO/
- Takes the idea of scree-space AO a bit further

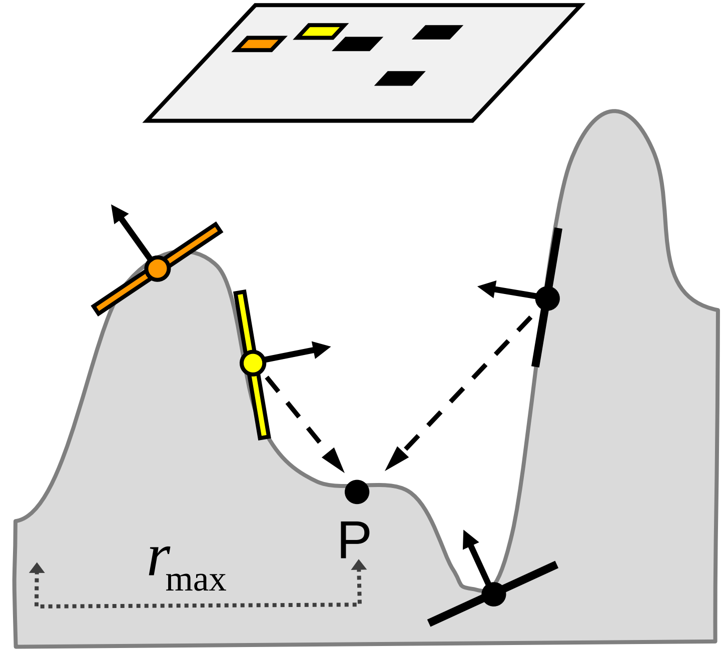
Screen space indirect illumination calculation

- Each sample is a small area light
  - Oriented around pixel normal
  - Radiance = direct light



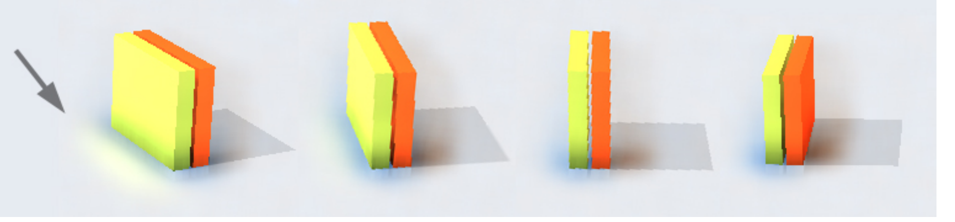
- For each sample
  - Compute form factor to P and accumulate contributions

- Results in one indirect bounce of light for nearby geometry

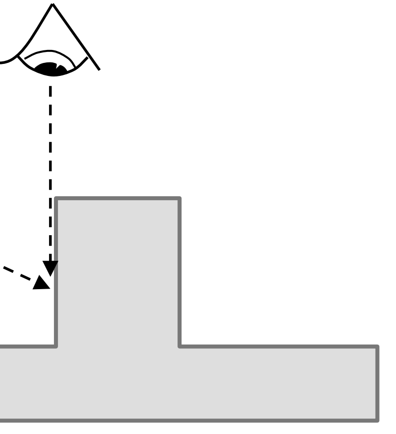


Limitations

- Only visible senders can contribute to indirect illumination
  - Especially grazing angles
  - Fade in/out of indirect light
  - Fortunately no abrupt changes visible



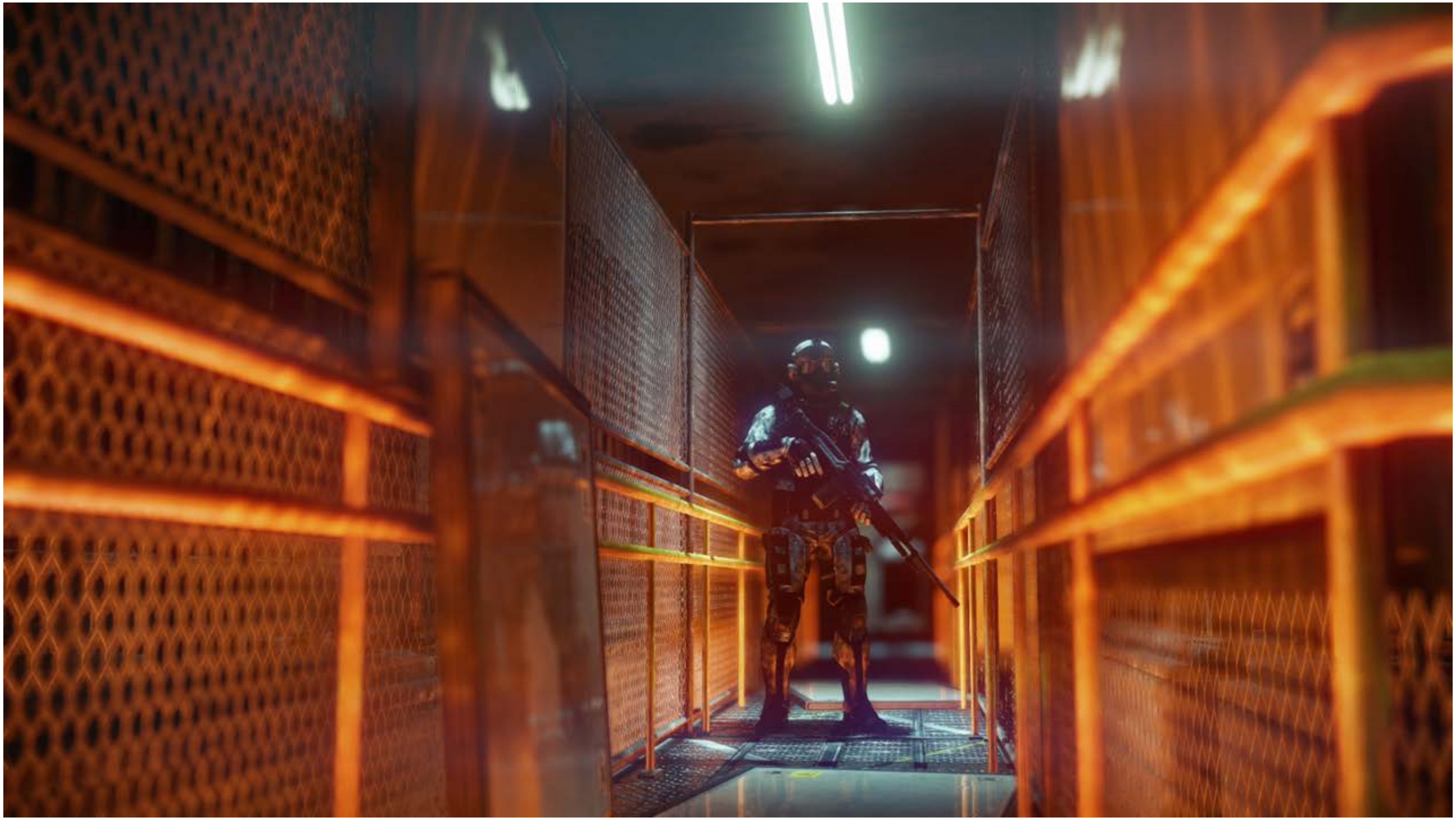
- Solution:
  Use multiple cameras



SSGI – Example result

Real-time local reflections

CryENGINE 3



Real-time local reflections

- Idea: **Raytracing in screen space** to approximate local reflections

- Basic Algorithm
  - Compute reflection vector for each pixel
    - Uses deferred normal and depth buffers
  - Raymarch along reflection vector

Screen-space multiple scattering in sparse media

Elek, Ritschel, Seidel, IEEE Computer Graphics & Applications 2012

Motivation: Sparse media

Idea

- Approximate only the phenomenological consequences of multiple scattering
  - Color shifts
  - Blurring

input: no scattering simulation

result

What's this?

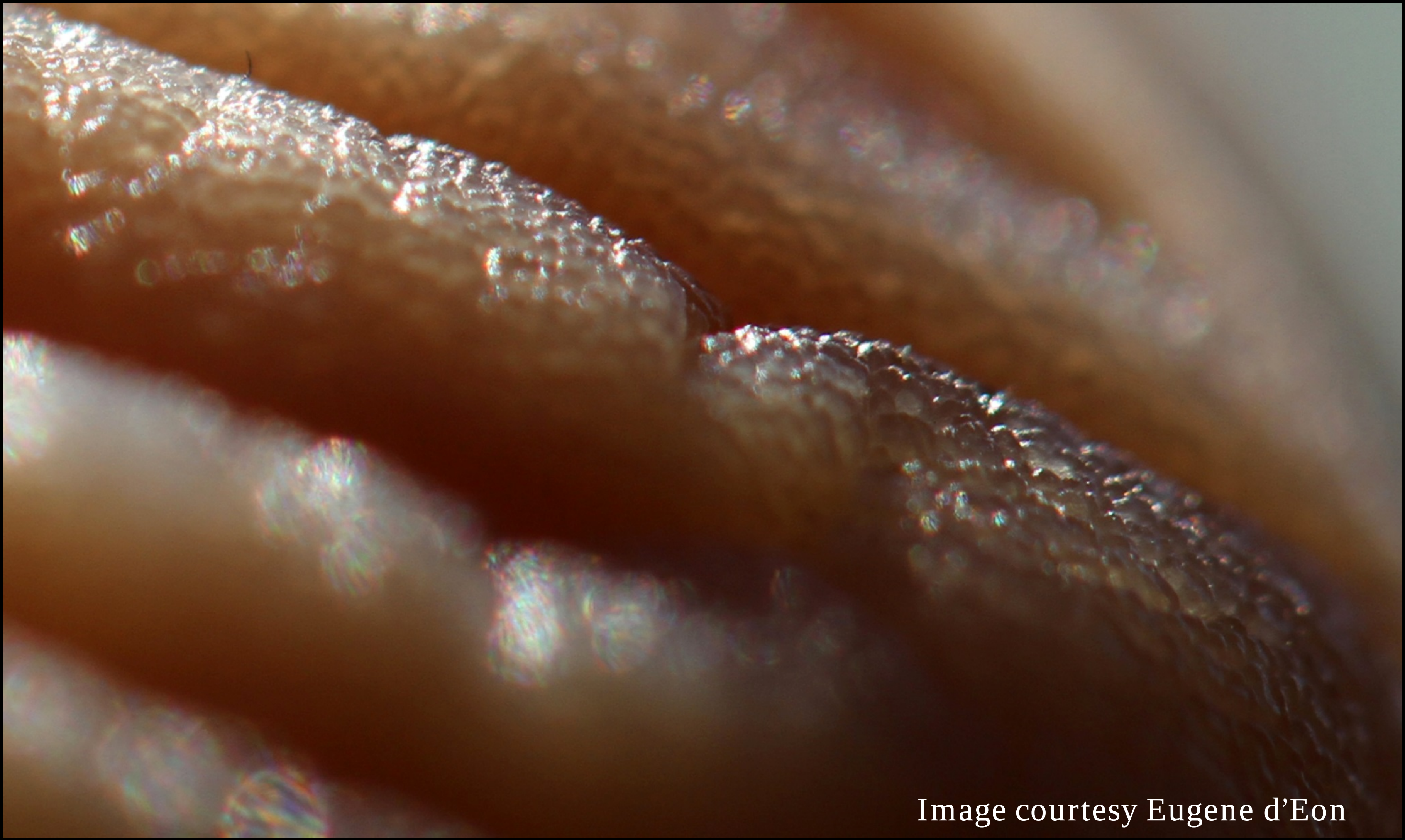
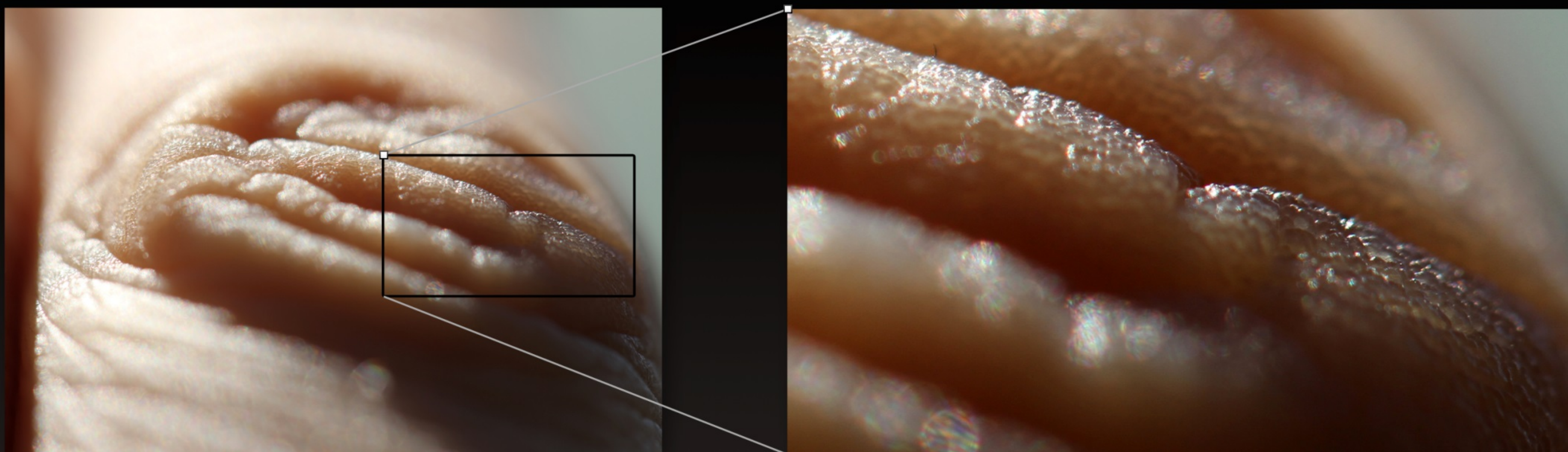
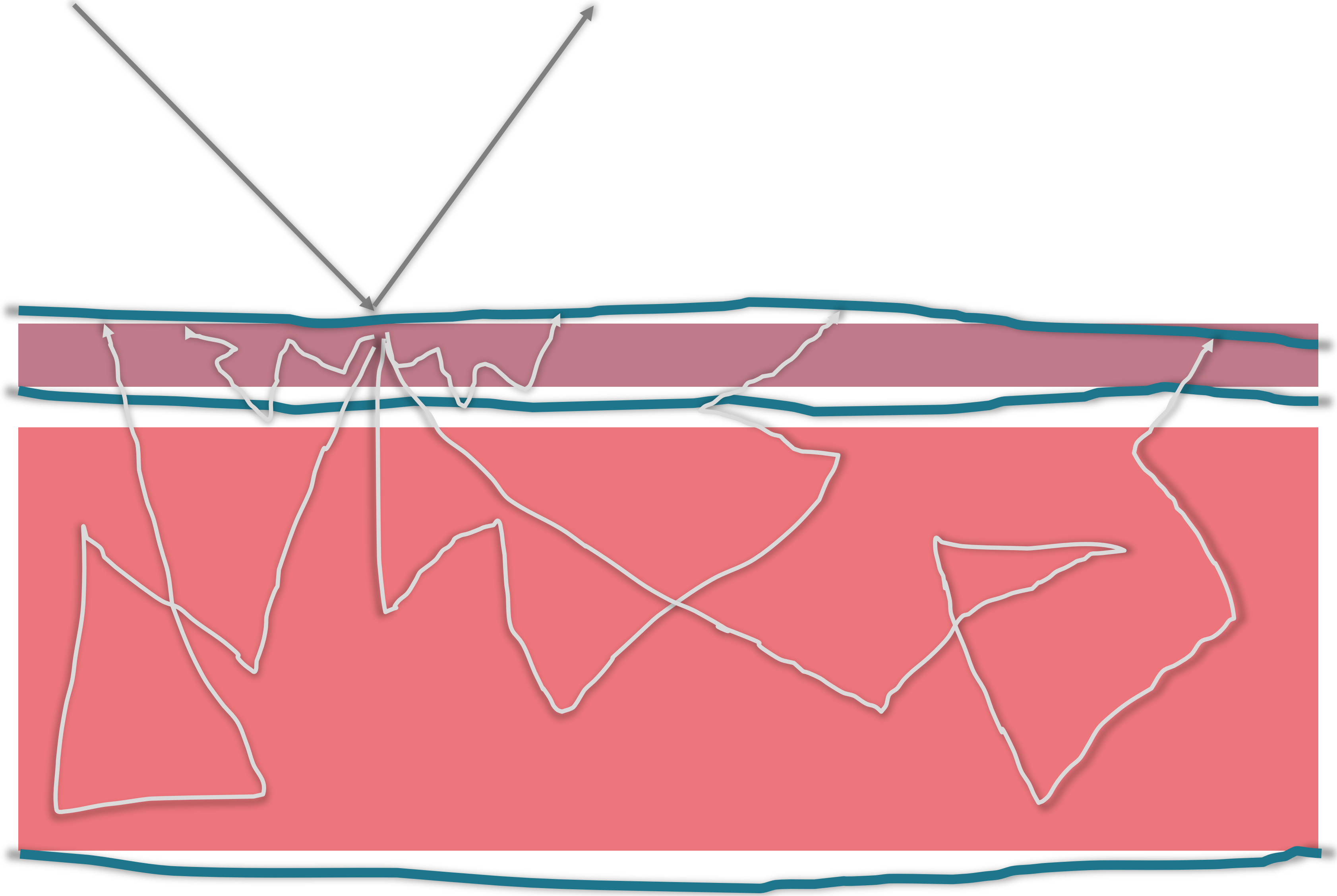
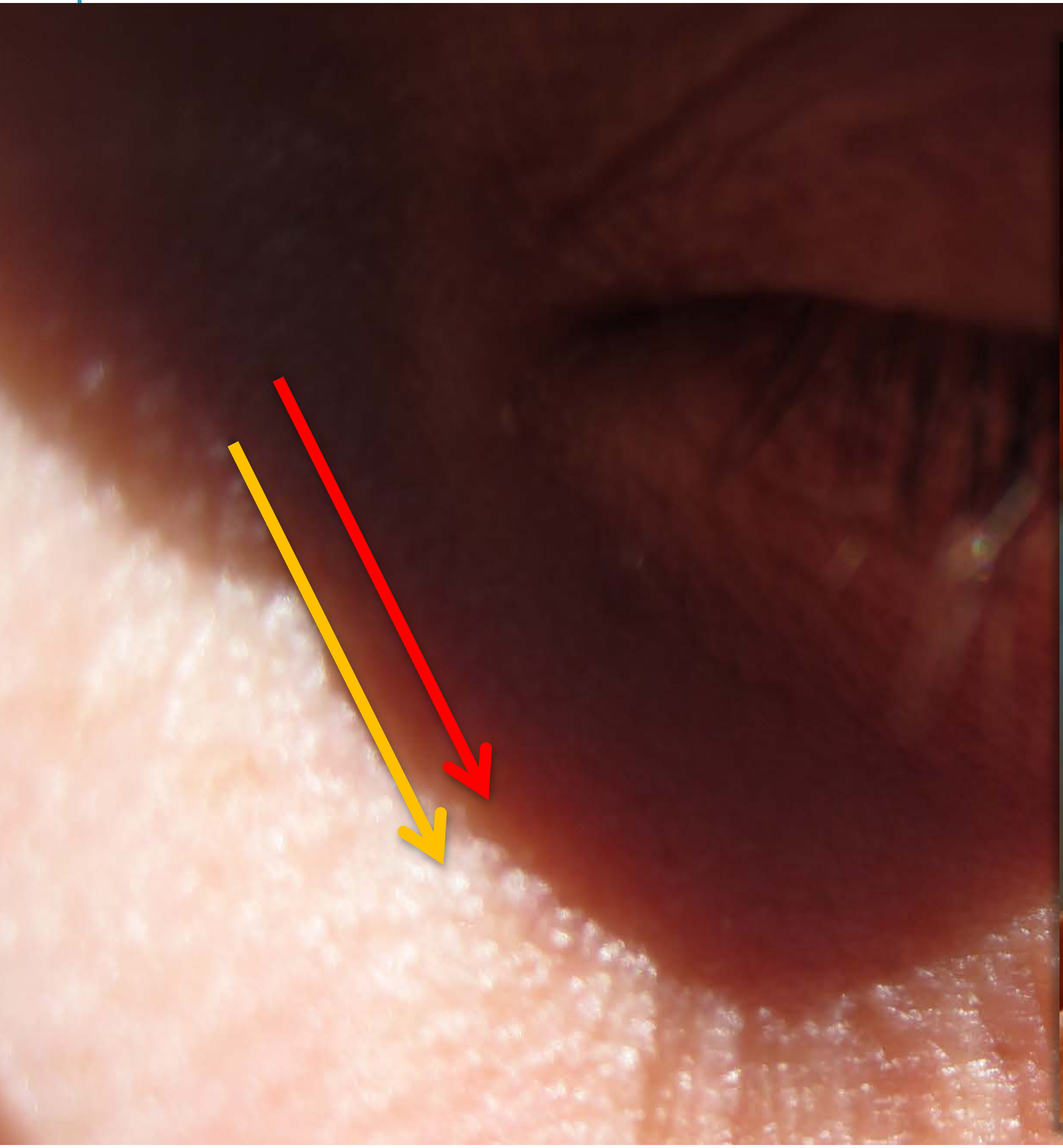


Image courtesy Eugene d’Eon



(Screen-space) subsurface scattering

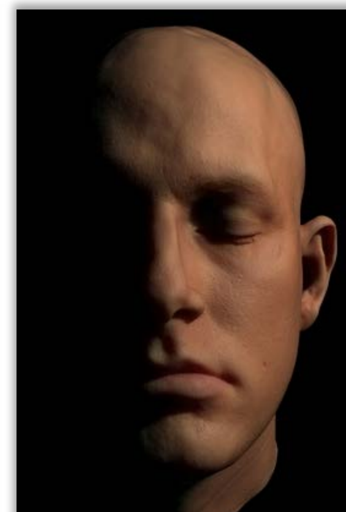




Subsurface scattering examples

Real

Simulated



Subsurface scattering in VFX

Seminal paper

- Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A Practical Model for Subsurface Light Transport, In SIGGRAPH '01

- **SIGGRAPH ‘01 electronic theatre video**

BSSRDF vs. BRDF

- BRDF is a special case of the so-called **BSSRDF**
  - BRDF assumes light enters and exists at the same point
  - BSSRDF does not make any such assumption
- BSSRDF = Bidirectional surface scattering distribution function [Nicodemus 1977]

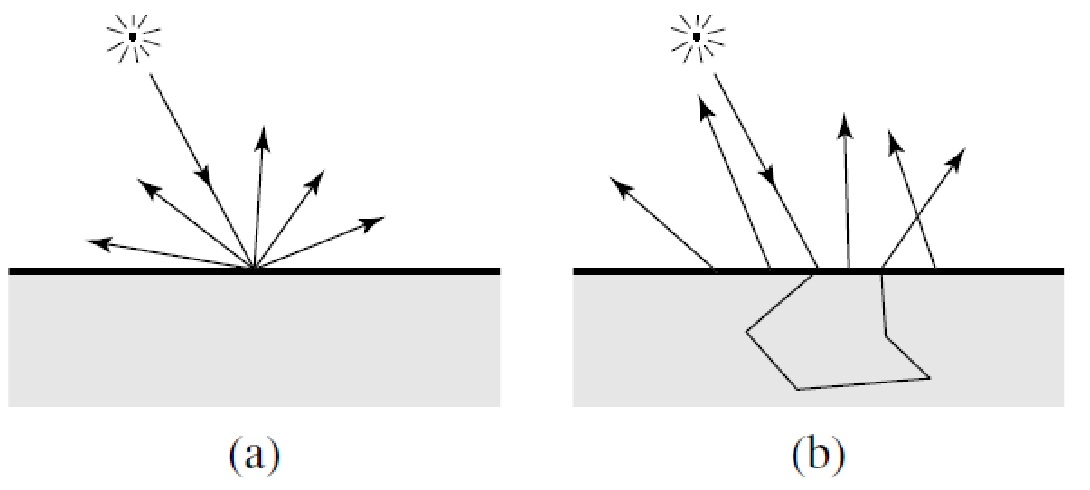


Figure 1: Scattering of light in (a) a BRDF, and (b) a BSSRDF.

BSSRDF

- 8D function (2x2 DOFs for surface + 2x2 DOFs for dirs)
- Differential outgoing radiance per differential incident flux (at two possibly different surface points)

$$dL_o(x_o, \vec{\omega}_o) = S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)\, d\Phi_i(x_i, \vec{\omega}_i)$$

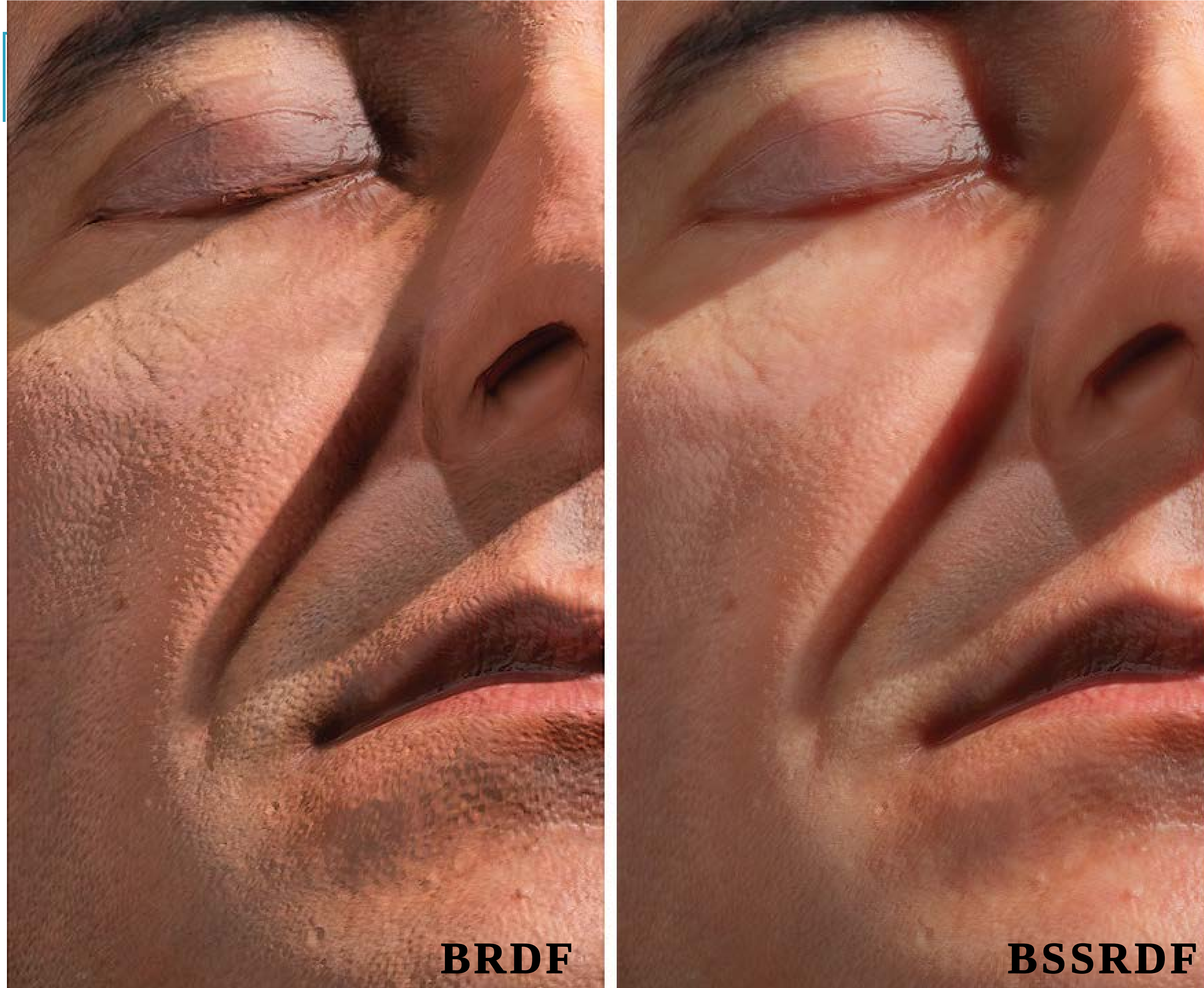
- Encapsulates all light behavior under the surface

BSSRDF vs. BRDF example

BSSRDF **BRDF**

BSSRDF vs. BRDF examples

- BRDF – hard, unnatural appearance


BRDF
BSSRDF

BSSRDF vs. BRDF examples

Generalized reflection equation

- So total outgoing radiance at x_o in direction ω_o is

$$L_o(x_o, \vec{\omega}_o) = \int_A \int_{2\pi} S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)\, L_i(x_i, \vec{\omega}_i)\, (\vec{n} \cdot \vec{\omega}_i)\, d\omega_i dA(x_i)$$

  - Over the original reflection equation, we've added the integration over the surface

Subsurface scattering simulation

- Path tracing – way too slow
- Photon mapping – more practical but still slow
  [Dorsey et al. 1999]


Problems with MC simulation of SS

- MC simulations (path tracing, photon mapping) can get very expensive for high-albedo media (skin, milk)
- High albedo means little energy lost at scattering events
  - Many scattering events need to be simulated (hundreds)

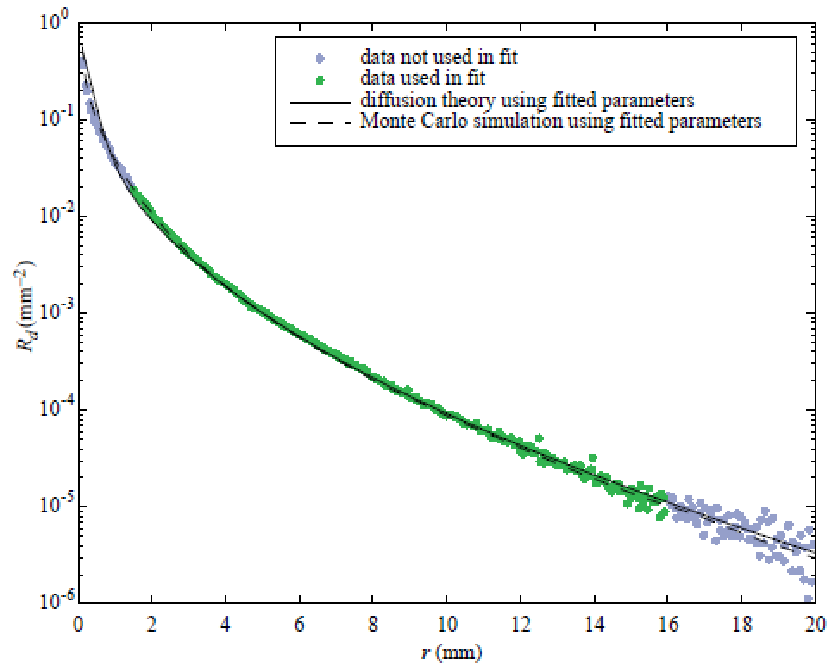
- Example: albedo of skim milk, a = 0.9987
  - After 100 scattering events, 87.5% energy retained
  - After 500 scattering events, 51% energy retained
  - After 1000 scattering events, 26% energy retained

- (compare to surfaces, where after 10 bounces most energy is usually lost)

Practical model for subsurface scattering

- Jensen, Marschner, Levoy, and Hanrahan, 2001
  - Won Academy award (Oscar) for this contribution

- Can find a diffuse BSSRDF $R_d(r)$, where r = $||\mathbf{x}_0 - \mathbf{x}_i||$
  - 1D instead of 8D !



Practical model for subsurface scattering

- Several **key approximations** that make it possible

  - Principle of similarity
    - Approximate multiple directional scattering in a medium by **isotropic** medium with modified ("reduced") coefficients

  - Diffusion approximation
    - Multiple scattering can be modeled as **diffusion** (simpler equation than full RTE)

  - Dipole approximation
    - Closed-form solution of diffusion can be obtained by placing two virtual point sources in and outside of the medium

Approx. #1: Principle of similarity

- Observation
  - Even highly anisotropic medium becomes isotropic after many interactions because every scattering blurs light

- **Isotropic approximation**

Approx. #2: Diffusion approximation

- We know that radiance mostly isotropic after multiple scattering; assume homogeneous, optically thick
- Approximate radiance at a point with just a weighted sum of:
  - Constant term: **scalar** irradiance, or **fluence**

$$\phi(x) = \int_{4\pi} L(x,\vec{\omega})\, d\omega$$

  - Linear term: **vector irradiance**

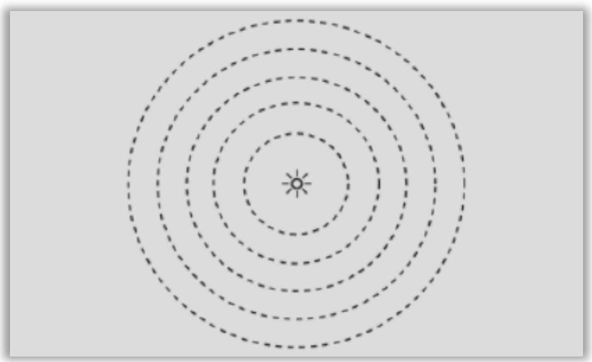
$$\vec{E}(x) = \int_{4\pi} L(x,\vec{\omega})\vec{\omega}\, d\omega$$

Diffusion approximation

- With the assumptions from previous slide, the full RTE (radiative transfer equation) can be approximated by the **diffusion equation**
  - Simpler than RTE (we're only solving for the scalar fluence, rather than directional radiance)
  - Skipped here, see [Jensen et al. 2001] for details

Solving diffusion equation

- Can be solved numerically
- Simple analytical solution for **point source in infinite homogeneous medium**:



$$\phi(x) = \frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r(x)}}{r(x)}$$

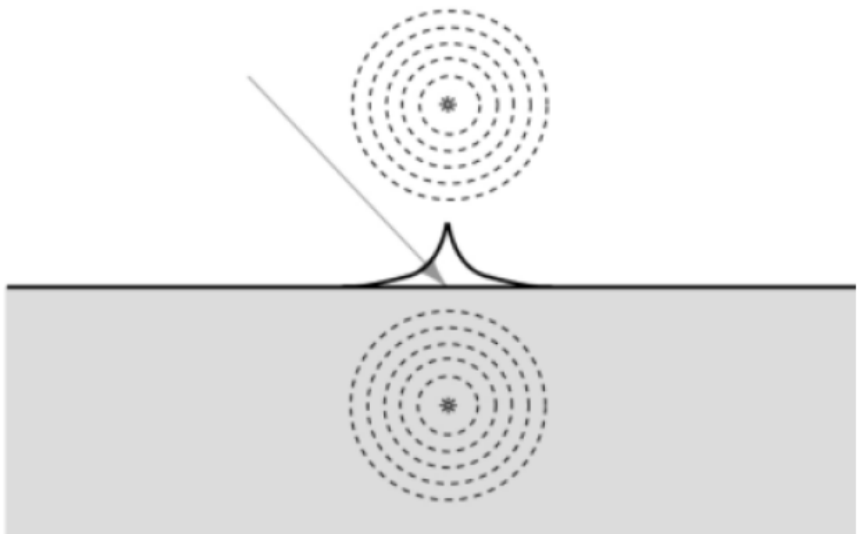
source flux

distance to source

- Material constants
  - Diffusion coefficient D
  - Effective transport coefficient σ_{tr}

Dipole approximation

- For surfaces, need to take boundary into account
- Solution can be approximated by a “dipole”
  - Two point sources, one above and one below the surface



Dipole approximation

- Dipole approximation leads to an analytic solution of the form

$$R(r) = \frac{e^{-r/d} + e^{-r/(3d)}}{8\,\pi\,d\,r}$$

  - Referred to as the "**diffusion profile**"
- Plot of $R(r)$

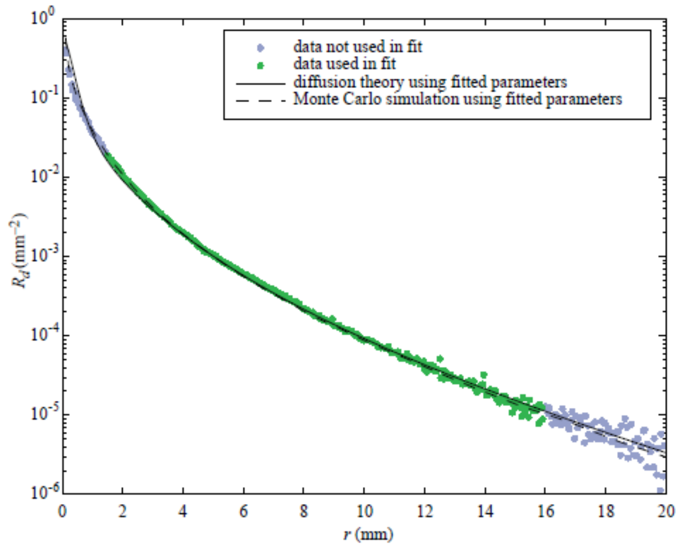


Figure 6: Measurements for marble (green wavelength band) plotted with fit to diffusion theory and confirming Monte Carlo simulation.

Final diffusion BSSRDF

Normalization term
(like for surfaces)

Diffuse multiple-scattering
reflectance

$$S_d(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o) = \frac{1}{\pi} F_t(\eta, \vec{\omega}_i) R_d(||x_i - x_o||) F_t(\eta, \vec{\omega}_o)$$

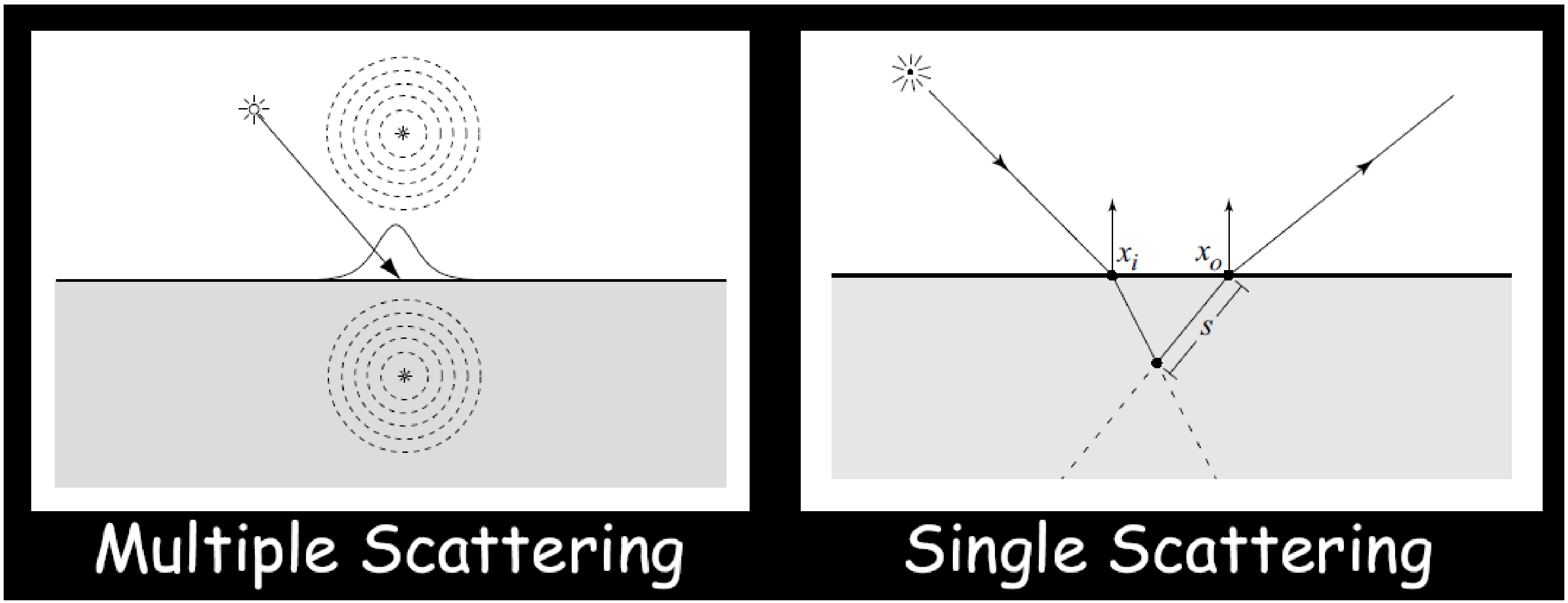
Fresnel term for
incident light

Fresnel term for
outgoing light

Single scattering term

- Cannot be accurately described by diffusion
- Much shorter influence than multiple scattering
- Computed by classical MC techniques (marching along ray, connecting to light source)

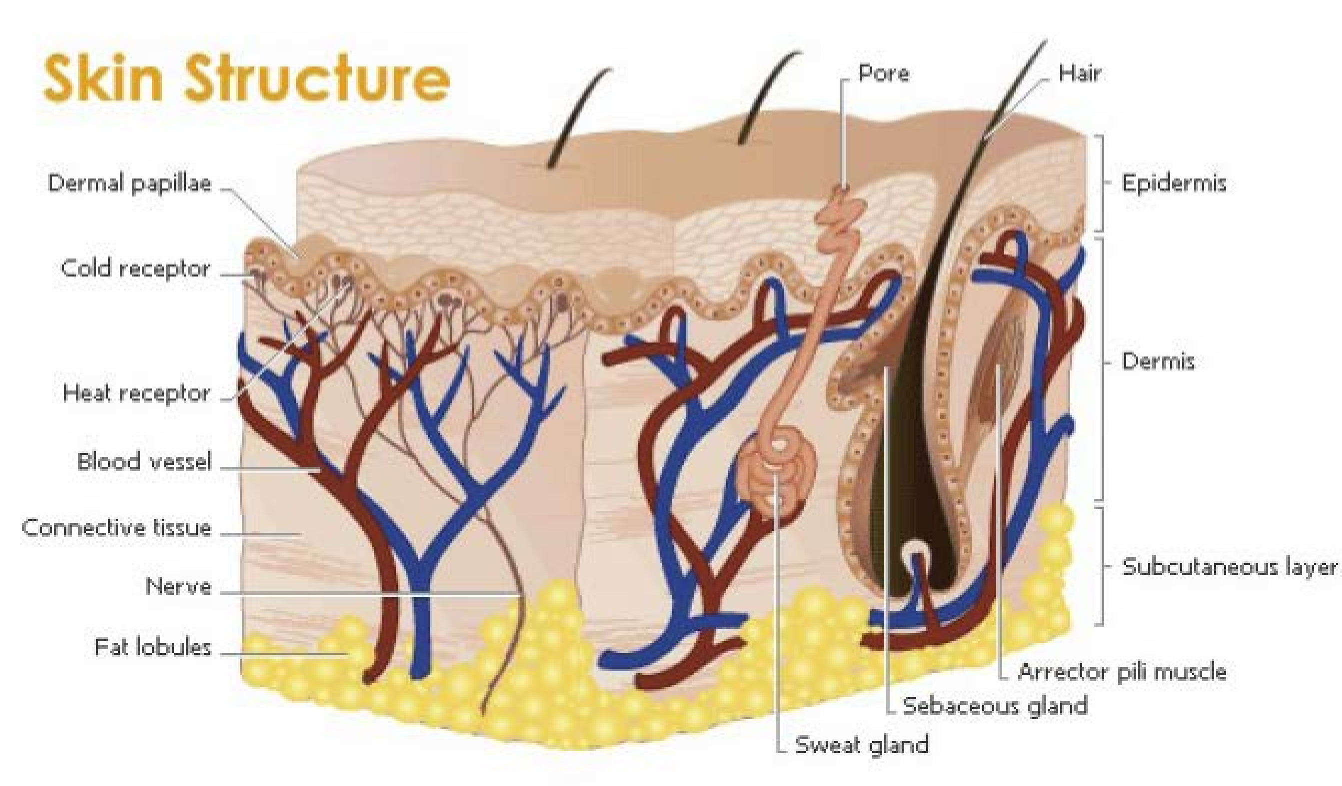
Complete BSSRDF model



MC simulation vs. BSSRDF model

Multiple Dipole Model

- Skin is NOT an semi-infinite slab



Multiple Dipole Model

- [Donner and Jensen 2005]
- Dipole approximation assumed semi-infinite homogeneous medium
- Many materials, namely skin, has multiple layers of different optical properties and thickenss
- Solution: infinitely many point sources

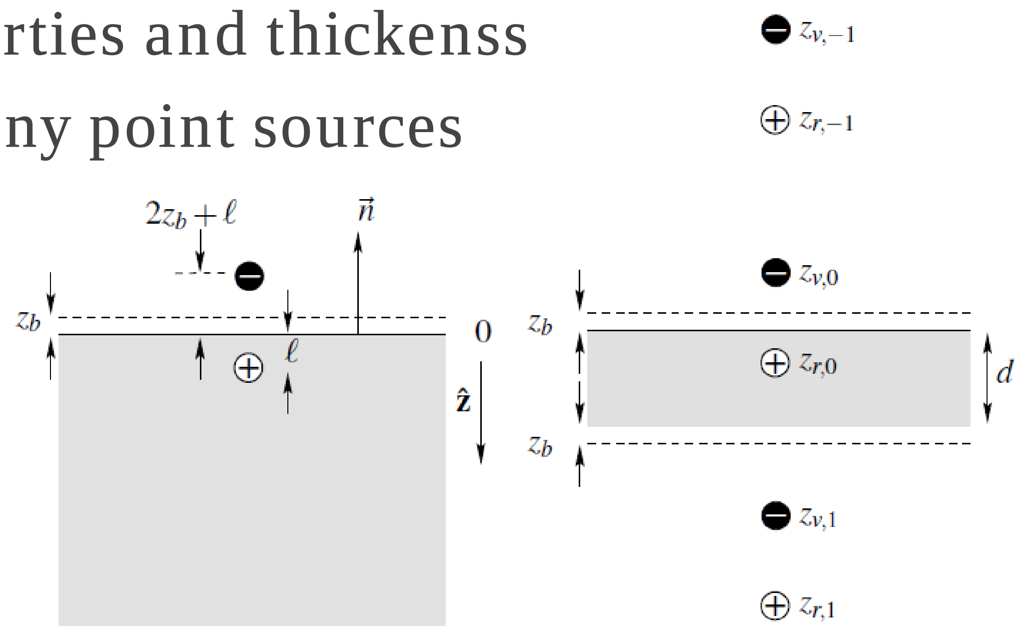
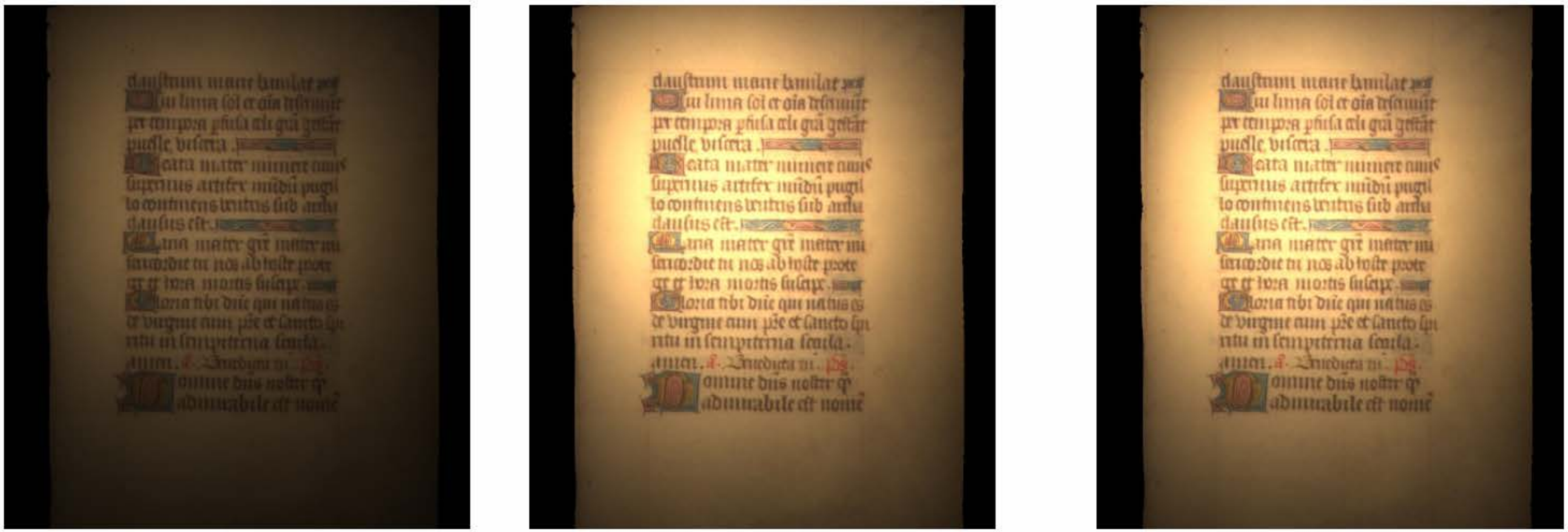


Figure 1: *Dipole configuration for semi-infinite geometry (left), and the multipole configuration for thin slabs (right).*

Diplole vs. multipole

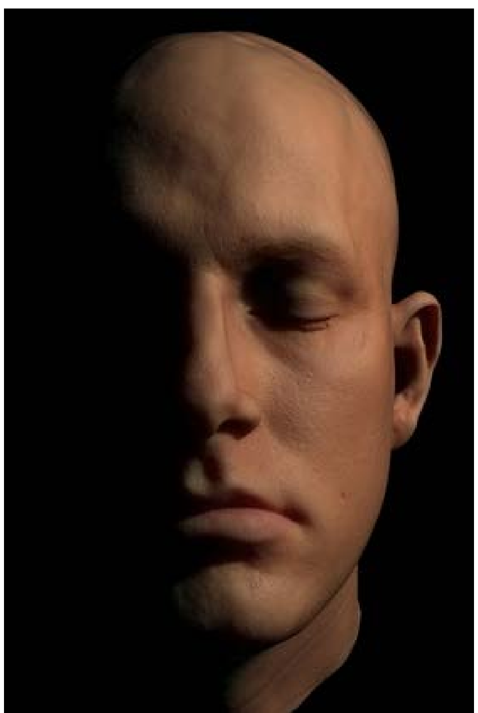


Dipole model | Multipole model | Monte Carlo reference

Figure 4: *A piece of parchment illuminated from behind. Note, how the dipole model (left) underestimates the amount of transmitted light, while the multipole model (middle) matches the reference image computed using Monte Carlo photon tracing (right).*

Multiple Dipole Model - Results

Epidermis Reflectance | Epidermis Transmittance | Upper Dermis Reflectance | Upper Dermis Transmittance | Bloody Dermis Reflectance | Surface Roughness | All Layers



Rendering with BSSRDFs

Rendering with BSSRDFs

1. Monte Carlo sampling of the diffusion profile [Jensen et al. 2001]
2. Point-based SSS solution [Jensen and Buhler 2002]
3. Real-time approximations
   1. Texture space [d'Eon et al. 2007]
   2. Screen space [Jimenez et al. 2009]

Monte Carlo sampling of the diffusion profile

- Original method by [Jensen et al. 2001]

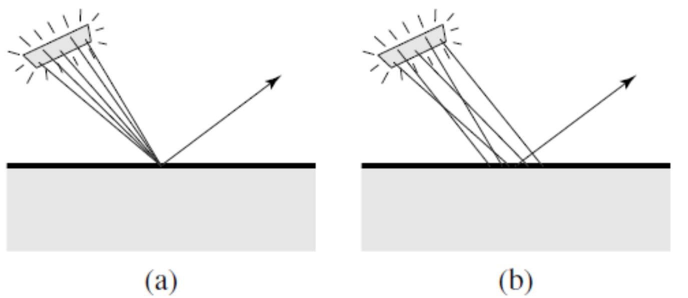


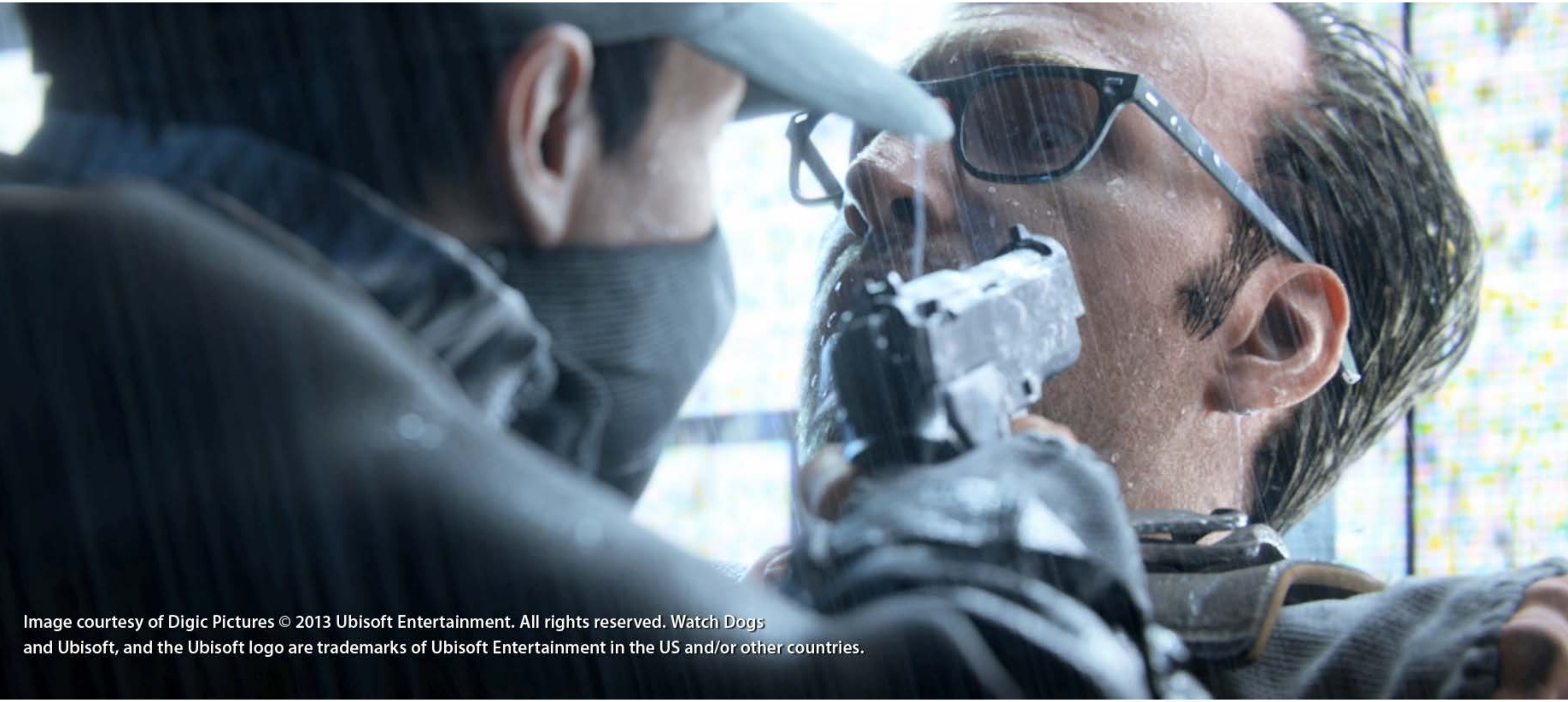
Figure 7: (a) Sampling a BRDF (traditional sampling), (b) sampling a BSSRDF (the sample points are distributed both over the surface as well as the light).

- Back in 2001, this was considered too slow
- By now, used in all major offline renderers (e.g. Arnold)

Production results made with Arnold

Oz the Great and Powerful (c 2013 Disney Enterprises, Inc.)

Production results made with Arnold



Point-based SSS solution

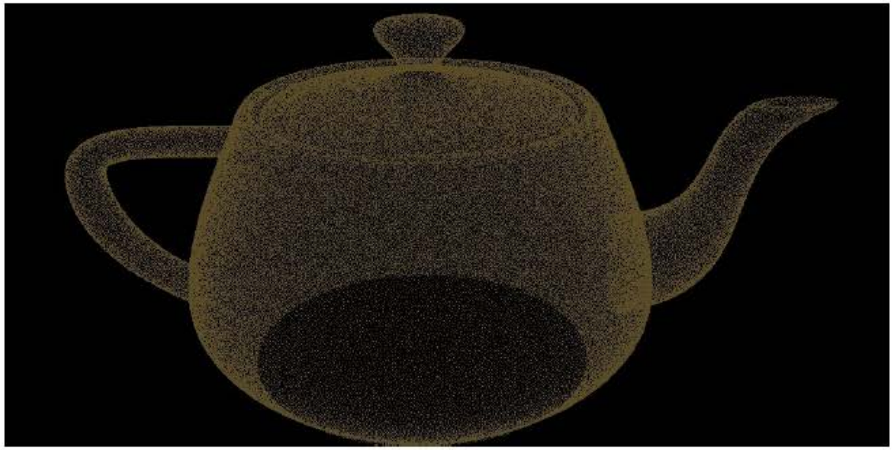
- [Jensen and Buhler 2002]
- Key idea: decouple computation of surface irradiance from integration of BSSRDF
- Algorithm
  - Distribute many points on translucent surface
  - Compute irradiance at each point
  - Build hierarchy over points (partial avg. irradiance)
  - For each visible point, integrate BSSRDF over surface using the hierarchy (far away point use higher levels)

Point-based SSS solution – Results

BSSRDF: sampled evaluation - 18 minutes

Illumination from a HDR environment

BSSRDF: hierarchical evaluation - 7 seconds



The sample locations on the teapot

Point-based SSS solution – Results

Point-based SSS solution

- Drawbacks
  - Additional memory
  - Requires a pre-pass
  - Unfriendly to progressive rendering
  - Point density mean free path
  - Flickering artifacts

- Considered obsolete by now
  - Too slow for real-time rendering
  - Above drawback for offline rendering

Real-time SSS: texture-space filtering

- [d'Eon et al. 2007]
- Idea
  - Approximate diffusion profile with a sum of Gaussians
  - Blur irradiance in texture space on the GPU
    - Fast because 2D Gaussians is separable
    - Have to compensate for stretch

Real-time SSS: texture-space filtering

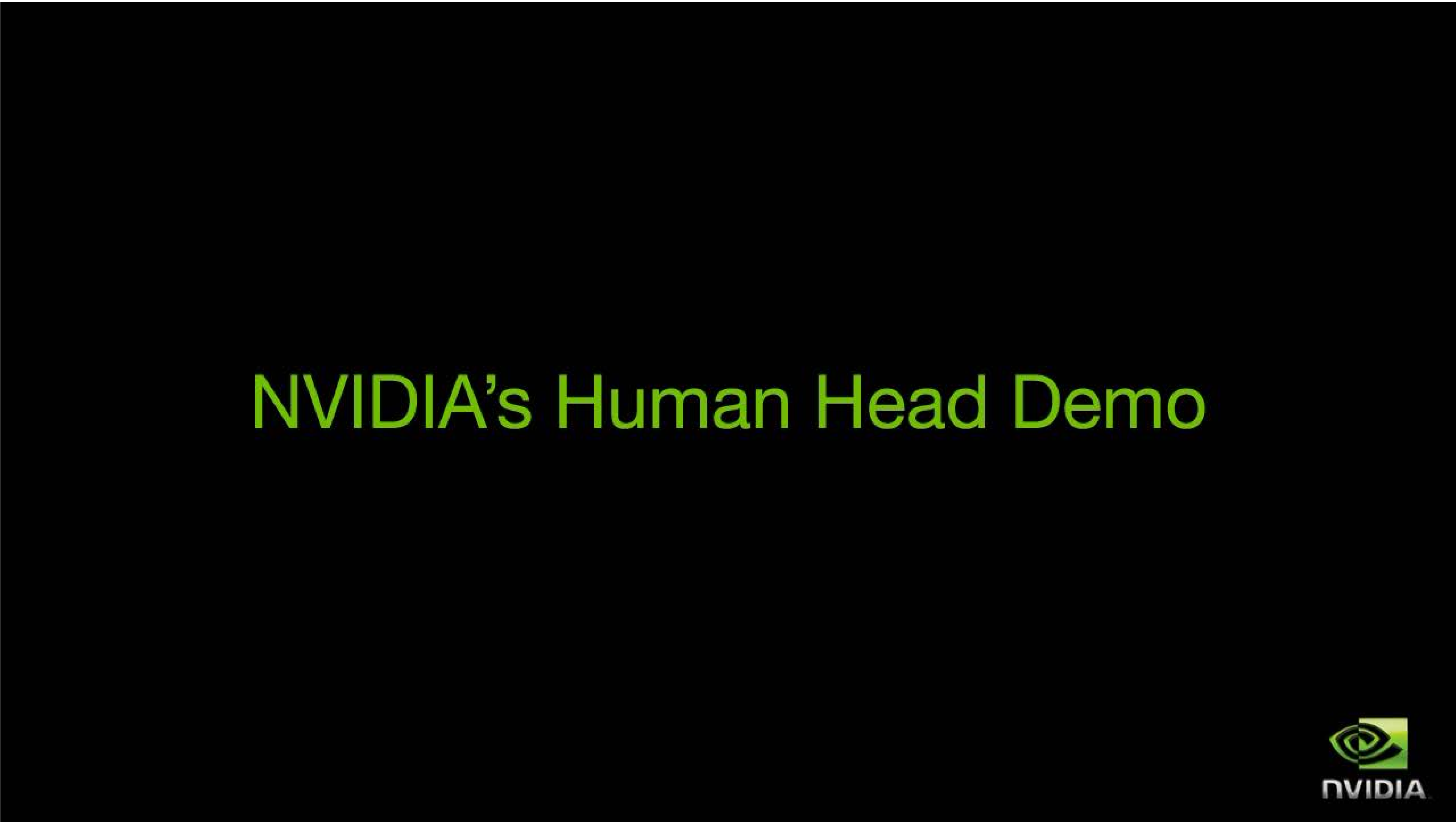
- Irradiance video

Real-time SSS: texture-space filtering

- Albedo (reflectance) map, i.e. texture
- Illumination
- Radiosity (=albedo * illum) filtered by the individual Gaussian kernels
- Specular reflectance

Real-time SSS: texture-space filtering

Real-time SSS: texture-space filtering

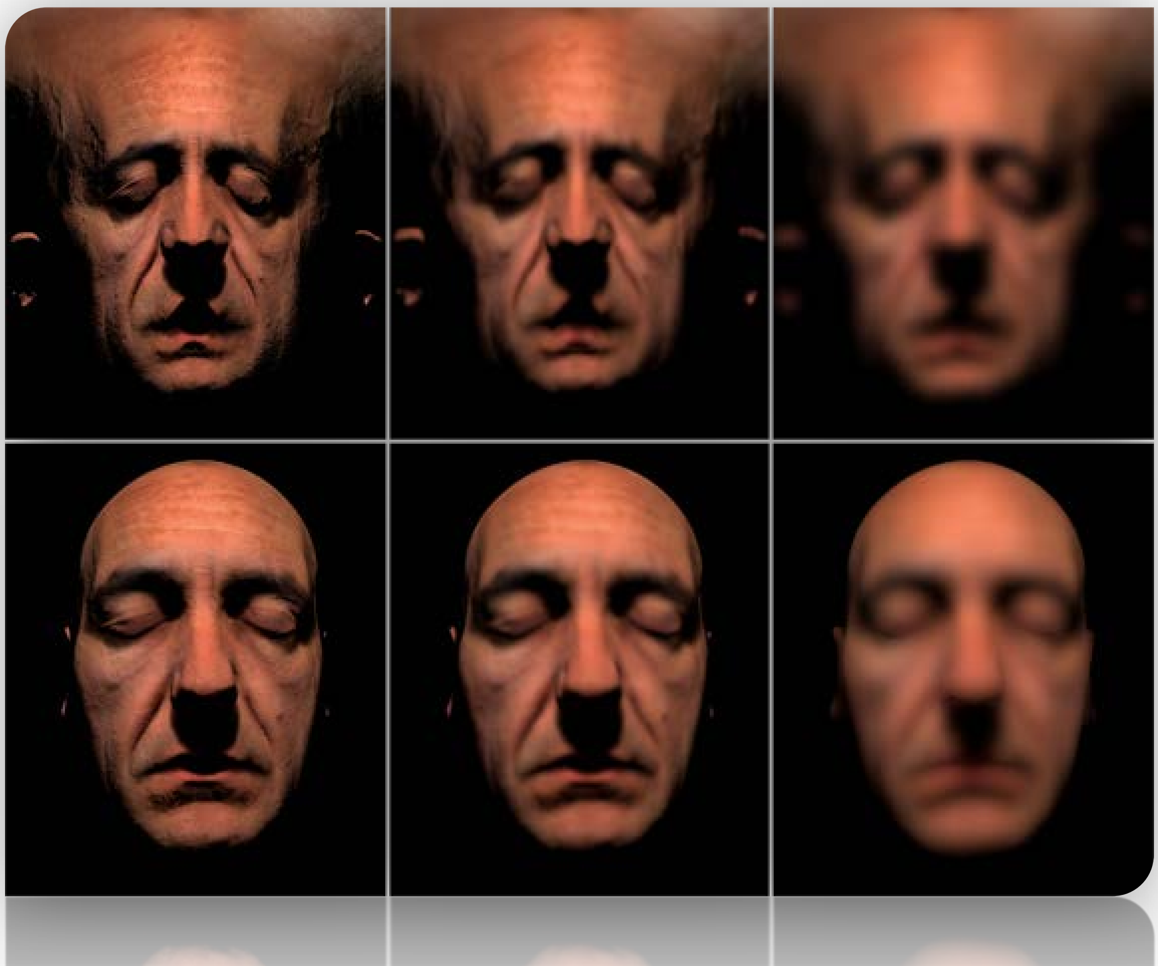


Real-time SSS: Image space filtering

- [Jimenez et al. 2009]
- Addresses scalability (think of many small characters in a game)
- Used in CryEngine 3 and other

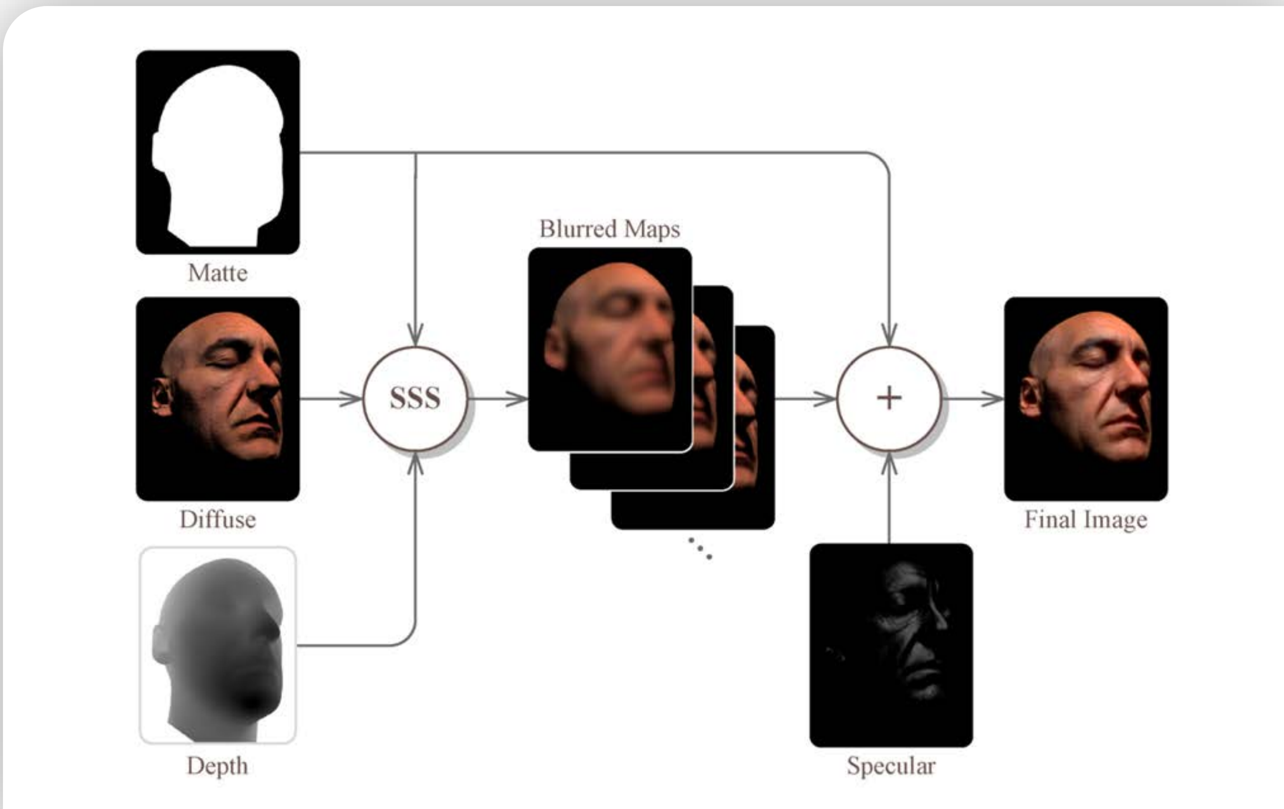
Real-time SSS: Image space filtering

- From texture space to screen space filtering



Real-time SSS: Image space filtering

- Method overview



Real-time SSS: Image space filtering

- Show video & real-time demo

- **More details**
  - **Jorge Jimenez:** Separable Subsurface Scattering
    http://advances.realtimerendering.com/s2012/

Additional reading on screen-space techniques

- Secrets of CryENGINE 3 Graphics Technology
  - http://www.crytek.com/cryengine/presentations/secrets-of-cryengine-3-graphics-technology

THANK YOU!

Questions?

Computer Graphics Charles University

[*cgg.mff.cuni.cz/~jaroslav*]