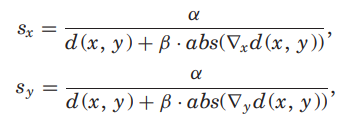
**SSS Techniques**

* Texture-space
  + Satisfying results
  + Does not scale well with increased texture sizes and amount
* Global-space
  + Closer to physical realism
  + Only suited for offline computations
* Screen-space
  + Difficulties with shadows generated by geometry outside the screen-space
  + Worst case performance depends on the screen-space dimensions
  + Fast enough for online use
  + Difficulties with thin layers -> only front facing materials can impact the sss

**http://iryoku.com/sssss/downloads/Screen-Space-Perceptual-Rendering-of-Human-Skin.pdf**

* In contrast, the absence of high-frequency feature softening tends to be taken as a sign of age: This insight could potentially guide some age-dependant future skin shaders.
* Further tests could also possibly compare the results of a shader with gold-standards in the form of real pictures, as opposed to comparing against other existing shaders.
* In summary, our psychophysical experiments show that the obvious loss of physical accuracy from our screen-space approach goes mainly undetected by humans. Based on our findings, we have hypothesized what the most important perceptual aspects of human skin are. We hope that these findings motivate further research on real-time, perceptually accurate rendering.
* Bidirectional Scattering Surface Reflectance Distribution Function (BSSRDF) becomes

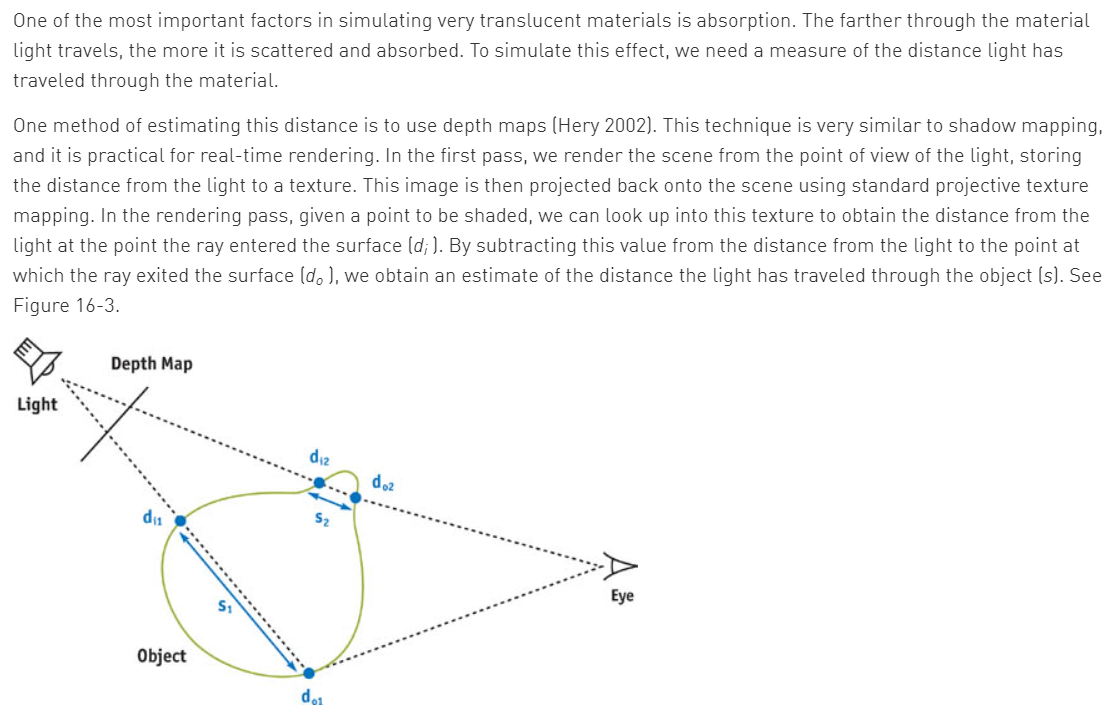
where xi and ωi are the position and angle of the incident light, xo and ωo are the position and angle of the radiated light, Ft is the Fresnel transmittance, and R is the diffusion profile of the material.

* Greater gradients in the depth map should also use narrower kernels. This is similar in spirit to using stretch maps, without the need to actually calculate them. We thus multiply the kernel width by the following stretch factors. We have

where d(x, y) is the depth of the pixel in the depth map, α indicates the global subsurface scattering level in the image, and β modulates how this subsurface scattering varies with depth gradient. The operators ∇x and ∇y compute the depth gradient, and are implemented on the GPU using the functions ddx and ddy, respectively. Note that increasing depth gradients reduce, the size of the convolution kernel as expected; in practice, this limits the effect of background pixels being convolved with skin pixels, given that in the edge, the gradient is very large and thus the kernel is very narrow. The value of α is influenced by the size of the object in 3D space, the field-of-view used to render the scene, and the viewport size (as these parameters determine the projected size of the object). All the images used in this work have empirically fixed values of α = 11 and β = 800.

**Texture-space method**

https://developer.nvidia.com/gpugems/gpugems/part-iii-materials/chapter-16-real-time-approximations-subsurface-scattering



The obvious problem with this technique is that it works only with convex objects: holes within the object are not accounted for correctly. In practice, this is not a big issue, but it may be possible to get around the problem using *depth peeling*, which removes layers of the object one by one (Everitt 2003).