

Rendering real-time self-shadowed dynamic hair

Alexandru Voicu
Prof. Duncan Gillies



Contents

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

Conclusions

Q & A



Background

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

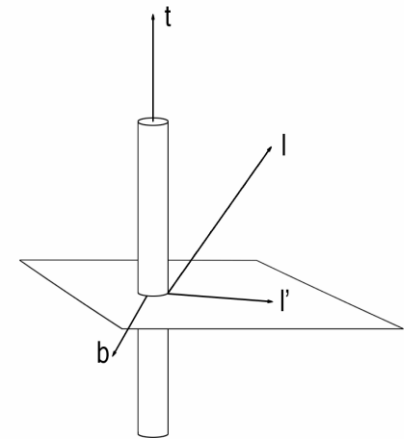
Conclusions

Q & A



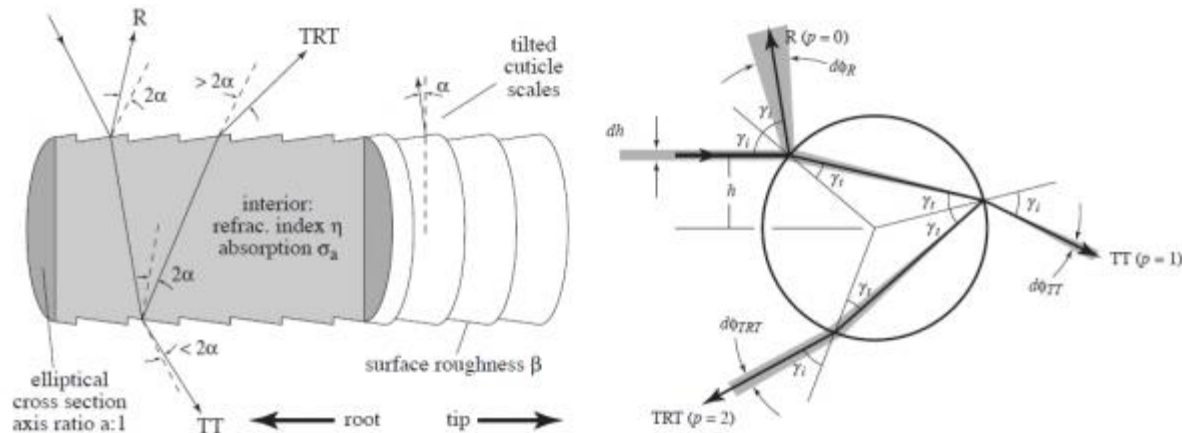
Kajika and Kay '89

- Rendering fur with 3D textures
- Hair strands ~ small smooth cylinders, having a diffuse and specular component
- Only takes into account tangent, binormal and light direction to compute the normal
- Efficient approach that produces acceptable results
- Does not capture the characteristics of human hair



Marschner et al. '03

- Based on a physical model:
 - scattering
 - better geometrical approximation
 - Bidirectional Reflectance Distribution Function
 - 2 refraction rays (R->TT->TRT)



Kajiya and Kay vs Marschner



Kajiya and Kay



Marschner



Real photo

Zinke et al. '04

- Marschner's model falls short for blonde (semi-transparent) hair and close-ups
- Zinke's model takes into account another (3rd) refraction ray



Zinke



Marschner

Zinke vs Self-shadowing

- Zinke's model falls shorts when the light is situated in the opposite direction of the camera
 - it doesn't take into account self-shadowing effects



Zinke



Self-shadowing

Offline implementations

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

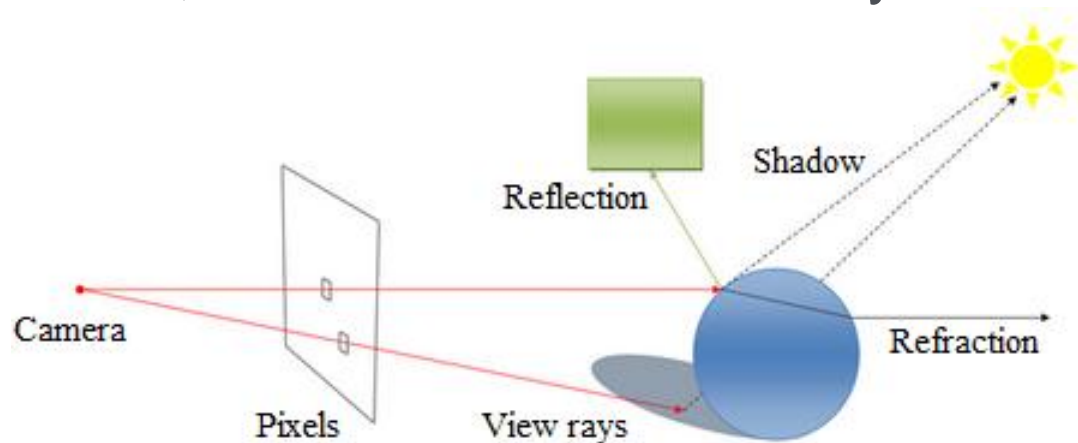
Conclusions

Q & A



Ray tracing (Rubin et al.) '79

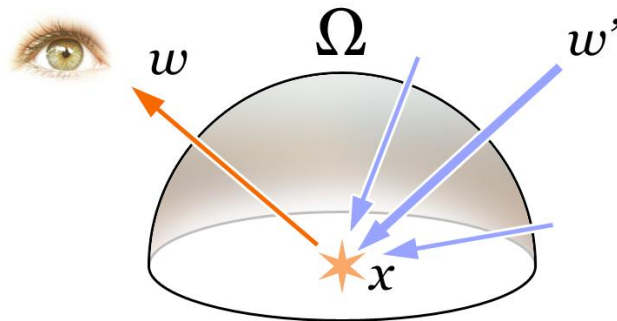
- When a ray hits a surface it can generate:
 - reflection, refraction or shadow rays



- Hair is represented as a volume density (voxels), similar to clouds and smoke
- Slow to compute only offline (3DS Max, Maya)

Photon mapping (Jensen) '00

- Similar to ray tracing, but adds indirect illumination as well (even slower to compute)
- Very realistic results, solves the rendering equation:
 - the total amount of light emitted from a point along a particular viewing direction, given a function for incoming light and a BRDF



Interactive and real-time frame rates

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

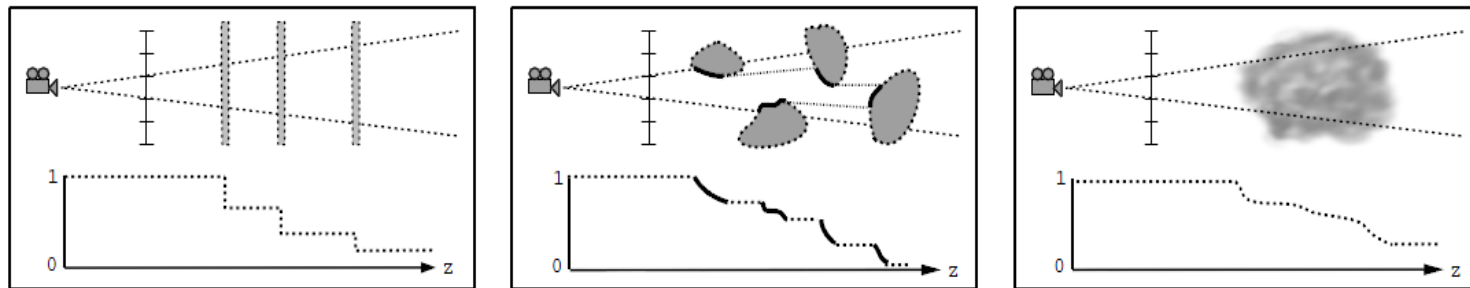
Conclusions

Q & A



Deep shadow maps (Lokovic et al.) '00

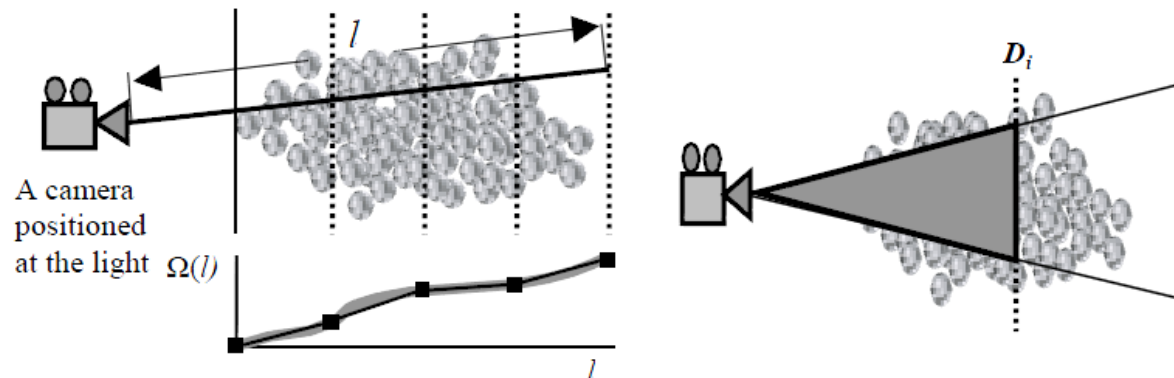
- Visibility function:
 - for every ray, filter the nearby transmittance function and resample at the centre.



- Supports both translucent and transparent objects and mip-mapping
- Geometry needs to be sorted: $O(N(\log N))$

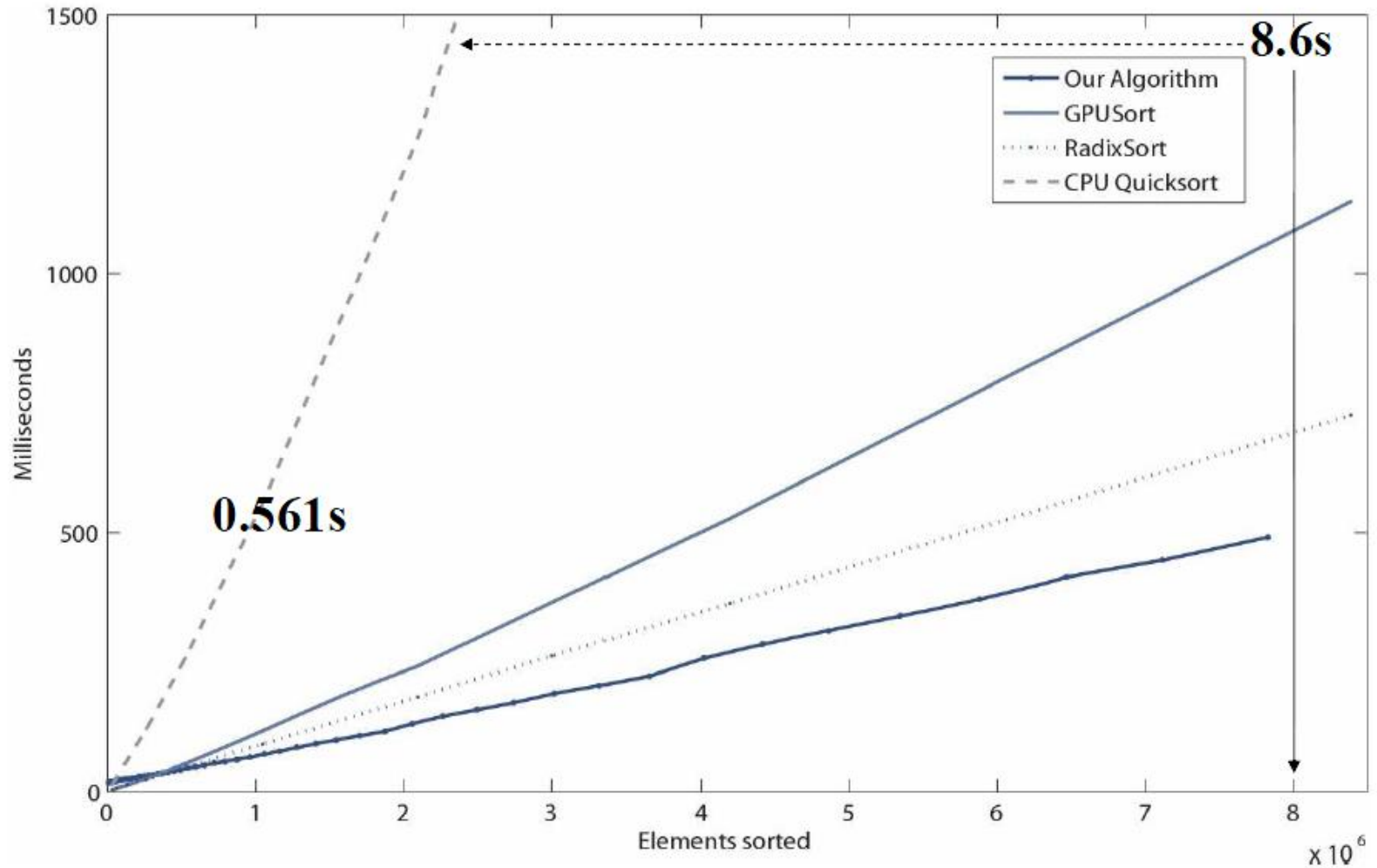
Opacity shadow maps (Kim et al.) '01

- Geometry is rendered to texture for a predefined number of times
 - along the light's direction
 - setting the *near* and *far* clips



- $O(NM)$, no sorting just rendering
- Hardware accelerated

GPU sort vs qsort



Personal work and improvements

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

Conclusions

Q & A

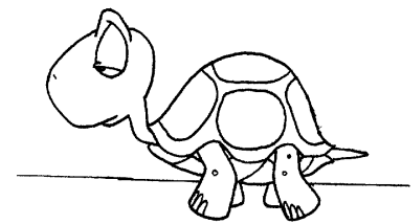


Current bottleneck

- As stated by Sintron and Assarsson in '07 the bottleneck is at sorting the geometry
 - their solution only works on dedicated CUDA hardware
- A fast sorting algorithm would be useful for alpha blending when rendering the final scene as well
- Opacity shadow maps don't require sorting, but require at least 16 maps for convincing results
 - $O(NM) = O(16N) \sim O(N\log N)$, for $N \sim 65K$
 - so it is slightly faster

Low dynamics sorted interval tree

- Hair has low dynamics, especially on a frame by frame basis
- Hair tends to cluster and clusters tend to have the same movement
- Insert in $O(1)$ on a correct assumption, or $O(\log N)$ if a wrong position is specified
 - improved sorted binary (interval) trees
 - skip lists
- Too slow when compared to C qsort
 - bad caching



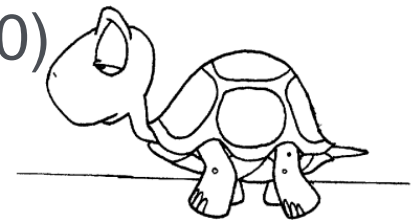
Counting (hash) sort

- Hair geometry is not distributed on a big surface (the hair sticks together) and it is uniformly spread
 - conversion from floating point values to integers without losing a lot of precision
- Hair tends to cluster and clusters tend to have the same movement
- Modified counting / ultra sort to store not just occurrences, but data (pointers) as well ~ hash
- Linear probing, hash table 3 x Input
- Better than C qsort !

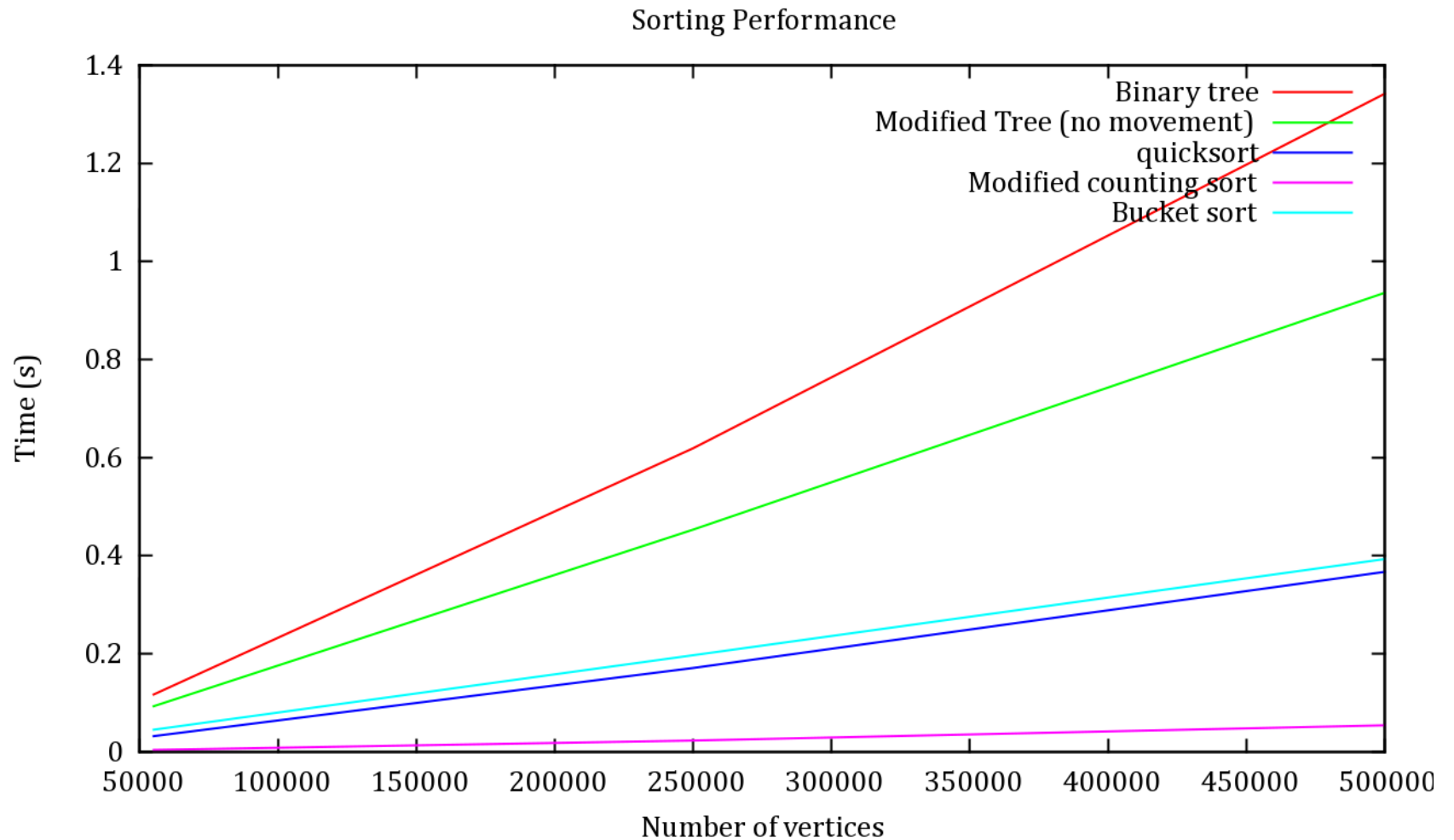


Bucket sort

- Count sort:
 - uses 3x more memory than actually required
 - size of the vector depends on the precision of the approximate sorting
- Bucket sort, or bin sort, solves these issues
- Slower than C qsort
 - uses lists (poor caching)
 - Bucket $O(2 * 7 * 500,000) = O(7,000,000)$
 - Quick $O(18 * 500,000) = O(9,000,000)$



Overall results



Future work

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

Conclusions

Q & A



Future work

- Combination between current approach and bucket sort to remove the counting sort problems
- Integrate it in a real-time rendering engine
 - test if counting sort performs well there as well
 - find the best trade-off between memory used and precision lost when sorting



Conclusions

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

Conclusions

Q & A



Conclusions

- Methods that don't take into account self-shadowing have non-realistic rendering
- Realistic results for offline rendering
 - Photon mapping and ray tracing
- Interactive methods, have good visuals, but only become real-time when rendering fewer details
- Current thesis speeds-up the most expensive component, sorting $O(N \log N)$, closer to real-time

Q & A

Background

Offline implementations

Interactive and real-time frame rates

Personal work and improvements

Future work

Conclusions

Q & A



Q & A

- Thank you for your attention
- For source code, presentation or final report
 - alexandru.voicu10@imperial.ac.uk
- Q & A

