Visual Computing Group

Part 4.5

Scalable Mobile Visualization:
Smart precomputation for complex lighting

Pere-Pau Vázquez, UPC

High quality illumination

- **Consistent illumination for AR**
- **Soft shadows**
- **Deferred shading**
- **Ambient Occlusion**

Consistent illumination for AR

- **High-Quality Consistent Illumination in Mobile Augmented Reality by Radiance Convolution on the GPU [Kán, Unterguggenberger & Kaufmann, 2015]**

- **Goal**
  - Achieve realistic (and consistent) illumination for synthetic objects in Augmented Reality environments

Consistent illumination for AR

- **Overview**
  - Capture the environment with the mobile
  - Create an HDR environment map
  - Convolve the HDR with the BRDF’s of the materials
  - Calculate radiance in realtime
  - Add AO from an offline rendering as lightmaps
  - Multiply with the AO from the synthetic object

Consistent illumination for AR

- **Capture the environment with the mobile**
  - Rotational motion of the mobile
    - In yaw and pitch angles to cover all sphere directions
  - Images accumulated to a spherical environment map
- **HDR environment map constructed while scanning**
  - Projecting each camera image
    - According to the orientation and inertial measurement of the mobile
  - Low dynamic range imaging is transformed to HDR
    - Camera uses auto-exposure
      - Two overlapping images will have slightly different exposure
  - Alignment correction based on feature matching
  - All in the device

Consistent illumination for AR

- **Convolve the HDR with the BRDF's of the materials**
  - Use MRT to support several convolutions at once
  - Assume distant light
  - One single light reflection on the surface
  - Scene materials assumed non-emissive
  - Use a simplified rendering equation
- **Weight with AO (obtained offline)**
  - Built for real and synthetic objects
  - Nee the geometry of the scene
    - Use a proxy geometry for the objects of the real world
    - Cannot be simply done on the fly

Consistent illumination for AR

- **Results**

Without AO

With AO

Images courtesy of Peter Kán

Consistent illumination for AR

- **Performance**

3D model	# triangles	Framerate
Reflective cup	25.6K	29 fps
Teapot	15.7K	30 fps
Dragon	229K	13 fps

- **Limitations**
  – Materials represented by Phong BRDF
  – AO and most shading (e.g. reflection maps) is baked

Soft shadows using cubemaps

- **Efficient Soft Shadows Based on Static Local Cubemap [Bala & Lopez Mendez, 2016]**
- **Goal**
  - Soft shadows in realtime

Taken from https://community.arm.com/graphics/b/blog/posts/dynamic-soft-shadows-based-on-local-cubemap

Soft shadows using cubemaps

- **Overview**
  - Create a local cube map
    - Offline recommended
    - Stores color and transparency of the environment
    - Position and bounding box
      - ***Approximates the geometry***
    - Local correction
      - Using proxy geometry
  - Apply shadows in the fragment shader

Soft shadows using cubemaps

- **Generating shadows**
  - Fetch texel from cubemap
    - Using the fragment-to-light vector
    - Correct the vector before fetching
      - Using the scene geometry (bbox) and cubemap creation position
        - To provide the equivalent shadow rays

  - Apply shadow based on the alpha value
  - Soften shadow
    - Using mipmapping and addressing according to the distance

Soft shadows using cubemaps

- **Conclusions**
  - Does not need to render to texture
    - Cubemaps must be pre-calculated
  - Requires reading multiple times from textures
  - Stable
    - Because cubemap does not change

- **Limitations**
  - Static, since info is precomputed

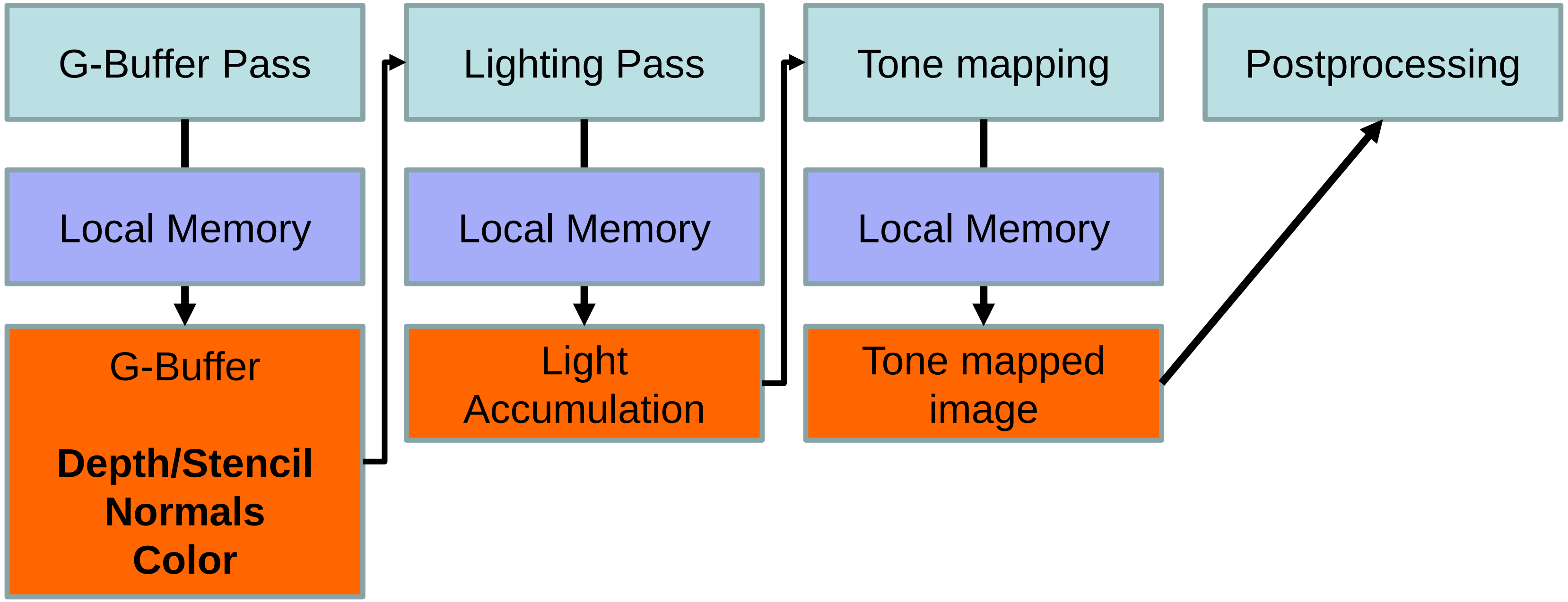
Physically-based Deferred Rendering

- **Physically Based Deferred Shading on Mobile [Vaughan Smith & Einig, 2016]**

- **Goal:**
  - Adapt deferred shading pipeline to mobile
  - Bandwidth friendly
  - Using Framebuffer Fetch extension
    - Avoids copying to main memory in OpenGL ES

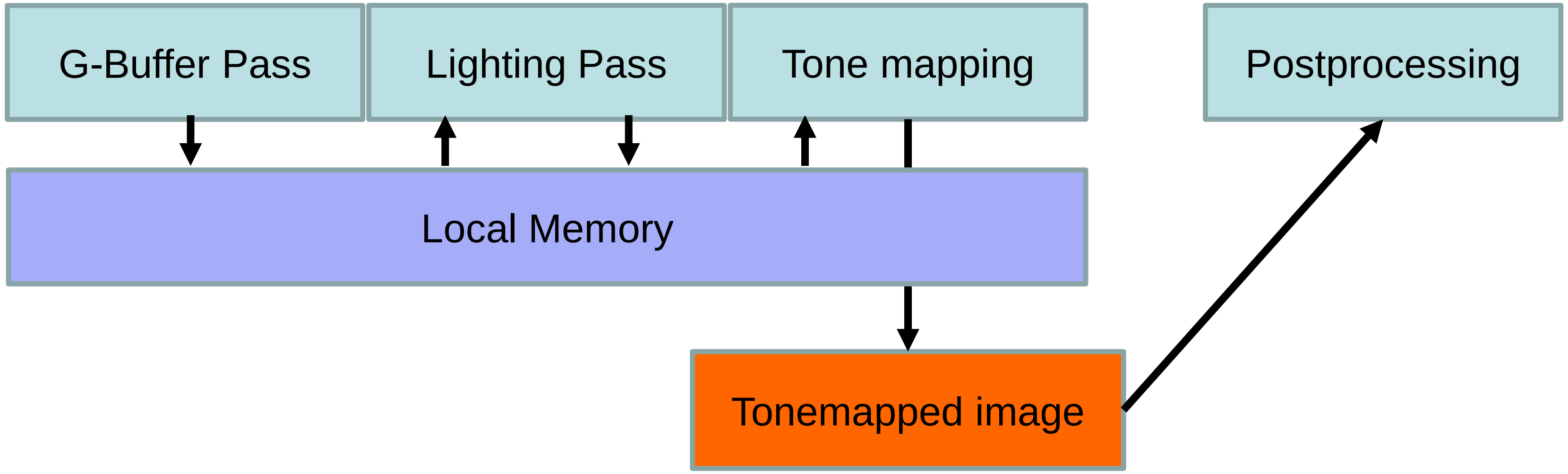
Physically-based Deferred Rendering

- **Overview**
  - Typical deferred shading pipeline



Physically-based Deferred Rendering

- **Main idea: group G-buffer, lighting & tone mapping into one step**
  - Further improve by using Pixel Local Storage extension
    - G-buffer data is not written to main memory
    - Usable when multiple shader invocations cover the same pixel
  - Resulting pipeline reduces bandwidth



Physically-based Deferred Rendering

- **Two G-buffer layouts proposed**
  - Specular G-buffer setup (160 bits)
    - Rgb10a2 highp vec4 light accumulation
    - R32f highp float depth
    - 3 x rgba8 highp vec4: normal, base color & specular color
  - Metallicness G-buffer setup (128 bits, more bandwidth efficient)
    - Rgb10a2 highp vec4 light accumulation
    - R32f highp float depth
    - 2 x rgba8 highp vec4: normal & roughness, albedo or reflectance metallicness

Physically-based Deferred Rendering

- **Lighting**
  - Use precomputed HDR lightmaps to represent static diffuse lighting
    - Shadows & radiosity
  - Can be compressed with ASTC (supports HDR data)
    - PVRTC, RGBM can also be used for non HDR formats
  - Geometry pass calculates diffuse lighting
  - Specular is calculated using Schlick’s approximation of Fresnel factor

Physically-based Deferred Rendering

- **Results (PowerVR SDK)**
  - Fewer rendering tasks
    - meaning that the G-buffer generation, lighting, and tonemapping stages are properly merged into one task.
    - reduction in memory bandwidth
      - 53% decrease in reads and a 54% decrease in writes
- **Limitations**
  - Still not big frame rates

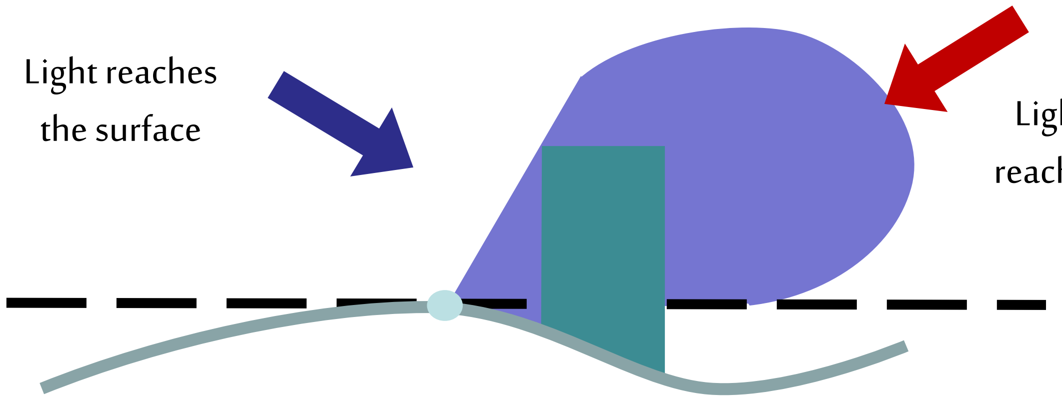
Ambient Occlusion in mobile

- **Optimized Screen-Space Ambient Occlusion in Mobile Devices [Sunet & Vázquez, Web3D 2016]**

- **Goal: Study feasibility of real time AO in mobile**
  - Analyze most popular AO algorithms: Crytek's, Alchemy's, Nvidia's Horizon-Based AO (HBAO), and Starcraft II (SC2)
  - Evaluate their AO pipelines step by step
  - Design architectural improvements
  - Implement and compare

Ambient Occlusion in mobile

- **Ambient Occlusion. Simplification of rendering equation**
  - The surface is a perfect diffuse surface (BRDF constant)
  - Light potentially reaches a point *p* equally in all directions
    - But takes into account point's visibility



$$L_o(p,\omega_o) = \frac{1}{\pi}\int_\Omega \rho(d(p,\omega_i))\ \cos\theta_i\, \mathrm{d}\omega_i$$

$$\rho(d) = \begin{cases} f(d) \in [0,1] & d < \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

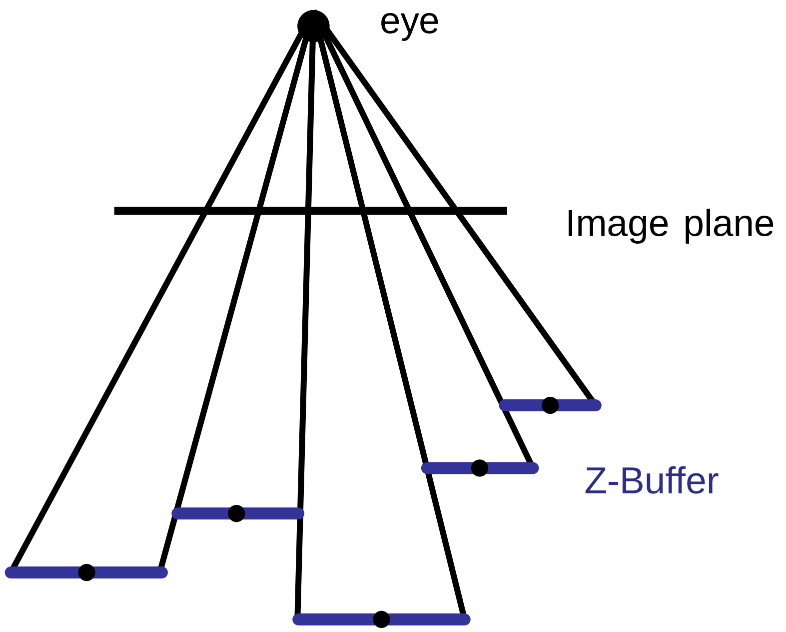
Ambient Occlusion in mobile

- **AO typical implementations**
  - Precomputed AO: Fast & high quality, but static, memory hungry
  - Ray-based: High quality, but costly, visible patterns…
  - Geometry-based: Fast w/ proxy structures, but lower quality, artifacts/noise…
  - Volume-based: High quality, view independent, but costly

  - Screen-space:
    - Extremely fast
    - View-dependent
    - [mostly] requires blurring for noise reduction
    - Very popular in video games (e.g. Crysis, Starcraft 2, Battlefield 3…)

Ambient Occlusion in mobile

- **Screen-space AO:**
  - Approximation to AO implemented as a screen-space post-processing
    - ND-buffer provides coarse approximation of scene's geometry
    - Sample ND-buffer to approximate (estimate) ambient occlusion instead of shooting rays



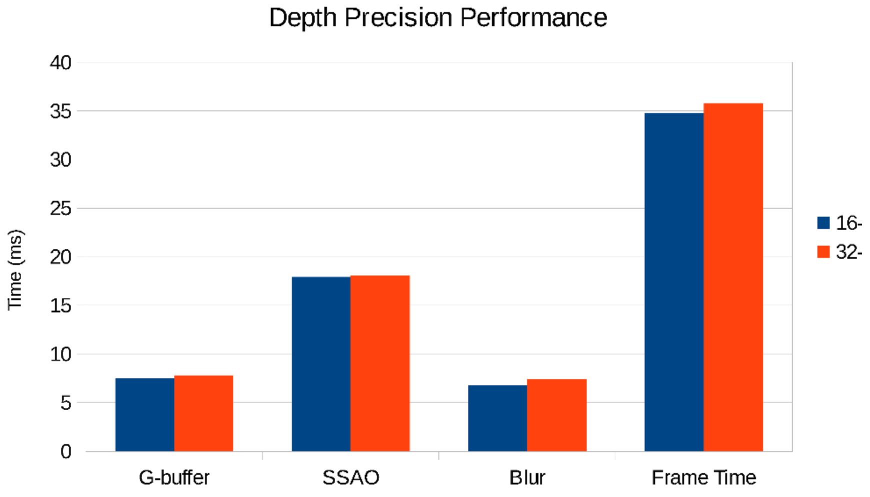
Assassin's Creed Syndicate

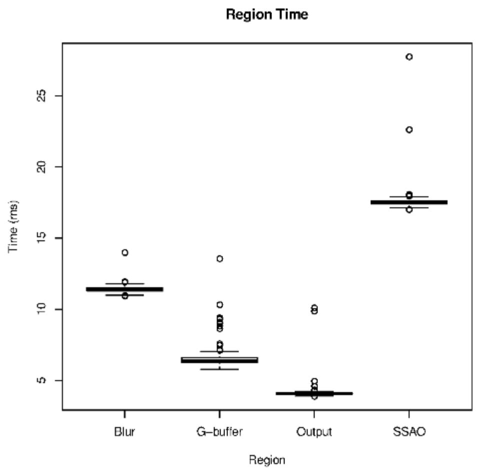
Ambient Occlusion in mobile

- **SSAO pipeline**
  1. Generate ND (normal + depth, OpenGL ES 2) or G-Buffer (ND + RGB…, OpenGL ES 3.+)
  2. Calculate AO factor for visible pixels
     a. Generate a set of samples of positions/vectors around the pixel to shade.
     b. Get the geometry shape (position/normal…)
     c. Calculate AO factor by analyzing shape…
  3. Blur the AO texture to remove noise artifacts
  4. Final compositing

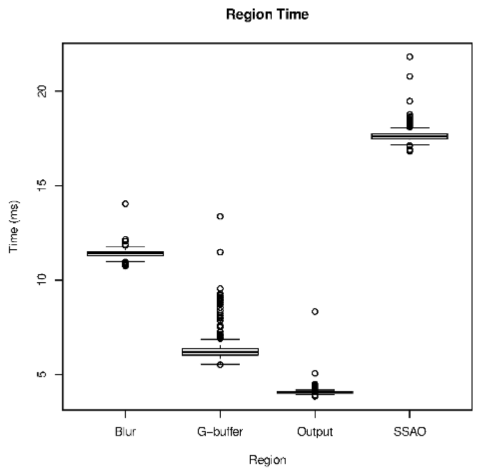
Ambient Occlusion in mobile

- **Optimizations. G-Buffer storage**
  - G-Buffer with less precision (32, 16, 8)
    - 8 not enough
    - 16 and 32 similar quality
  - Normal storage (RGB vs RG)
    - RGB normals are faster





RGB normals.



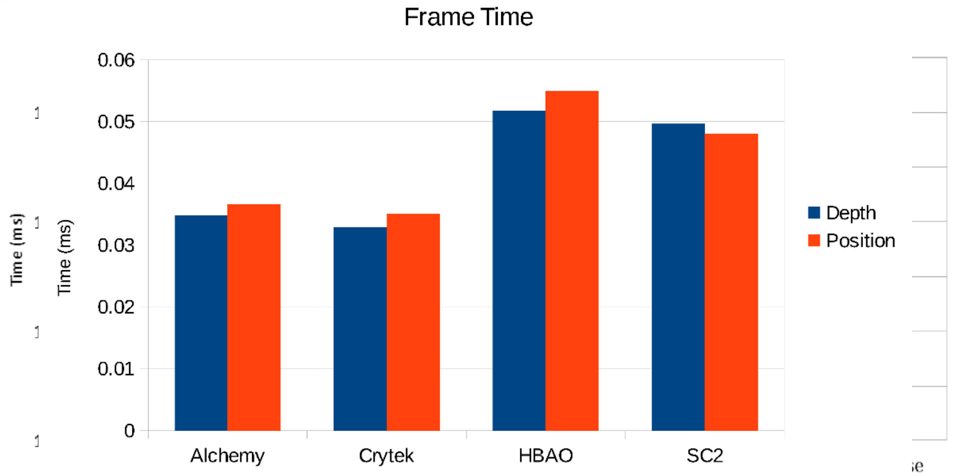
RG normals.

Ambient Occlusion in mobile

- **Optimizations. Sampling**
  - AO samples generation (disc and hemisphere)
    - Desktops use up to 32
    - With mobile, 8 is the affordable amount
      - Pseudo-random samples produces noticeable patterns
  - Our proposed solution
    - Compute sampling patterns offline
      - 2D: 8-point Poisson disc
      - 3D: 8-point cosine-weighted hemisphere (Malley's approach, as in [Pharr and Humprheys, 2010])
    - Scaling and rotating the resulting pattern ([Chapman, 2011])
    - Predictable, reproducible, robust

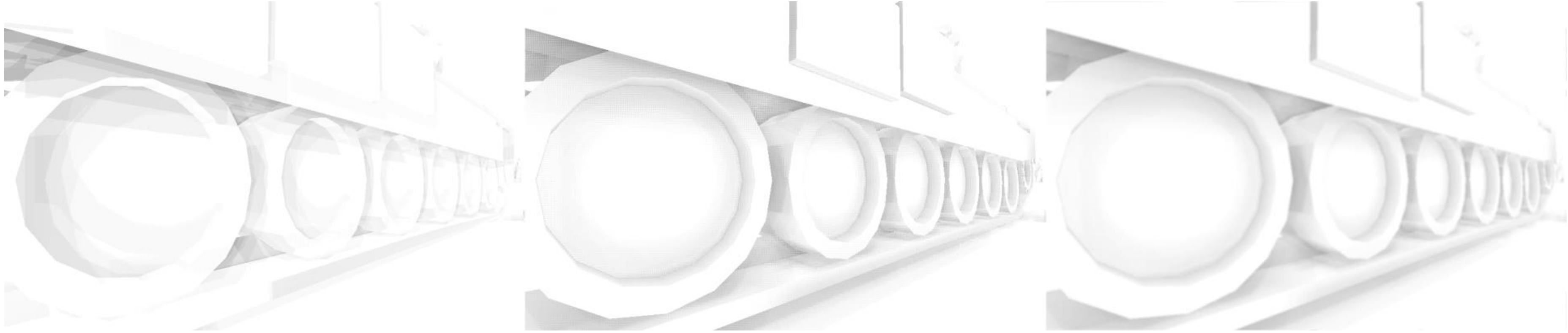
Ambient Occlusion in mobile

- **Optimizations. Getting geometry positions**
  – Transform samples to 3D
    - Inverse transform vs similar triangles
      – Precision for speed
    - Similar triangles are faster
  – Storing depth vs storing 3D positions in G-Buffer
    - Trades bandwidth for memory
    - Depth slightly better
    - Better profile for the application



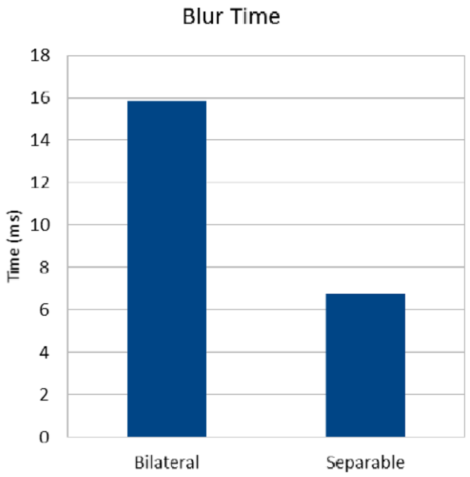
Ambient Occlusion in mobile

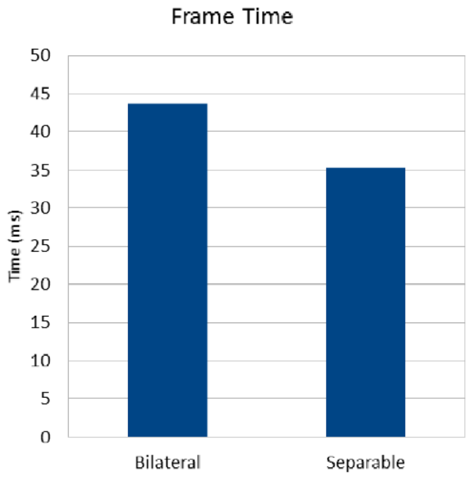
- **Optimizations. Banding & Noise**
  - Fixed sampling pattern produces banding (left)
  - Random sampling reduces banding but adds noise (middle)
  - SSAO output is typically blurred to remove noise (right)
    - But blurs edges



Ambient Occlusion in mobile

- **Optimizations. Banding & Noise**
  - User bilateral filter instead
    - Works better
    - Improve timings with separable filter



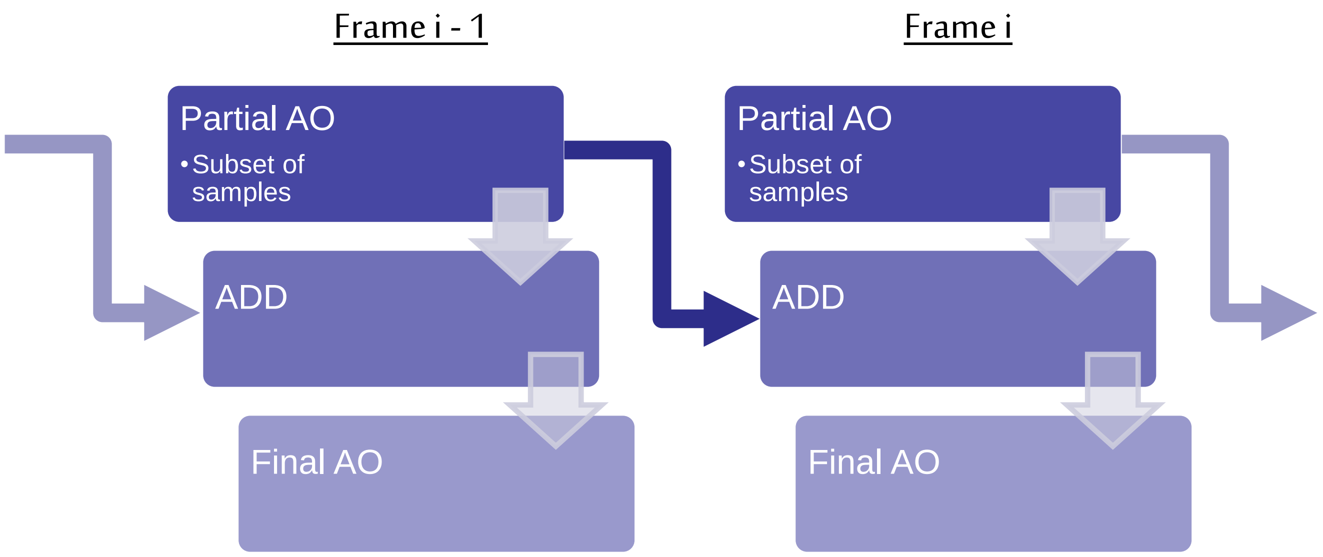


$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(||p - q||)\, G_{\sigma_r}(|I_q - I_p|)\, I_q$$

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Ambient Occlusion in mobile

- **Optimizations. Progressive AO**
  - Amortize AO throughout many frames



Ambient Occlusion in mobile

- **Optimizations**
  - Naïve improvement: Reduce the calculation to a portion of the screen
    - Mobile devices have a high PPI resolution
    - Reduction improves timings dramatically while keeping high quality
  - Typical reduction:
    - Offscreen render to 1/4th of the screen
    - Scale-up to fill the screen

Ambient Occlusion in mobile

- **Results**

Algorithm	Optimized (not progressive)	Optimized + progressive
Starcraft 2	17.8%	38.5%
HBAO	25.6%	39.2%
Crytek	23.4%	35.0%
Alchemy	24.8%	38.2%

Ambient Occlusion in mobile

- **Conclusions**
  - Developed an optimized pipeline for mobile AO
    - Analyzed the most popular AO techniques
      - Improved several important steps of the pipeline
      - Proposed some extra contributions (e.g. progressive AO)
    - Achieved realtime framerates with high quality
    - Developed techniques can be used in WebGL
  - Future Work
    - Further improvement of the pipeline
    - Developing “Homebrew” method
      - With all known improvements
      - Some extra tricks
      - Not ready for prime time yet

Visual Computing Group

Part 4.5

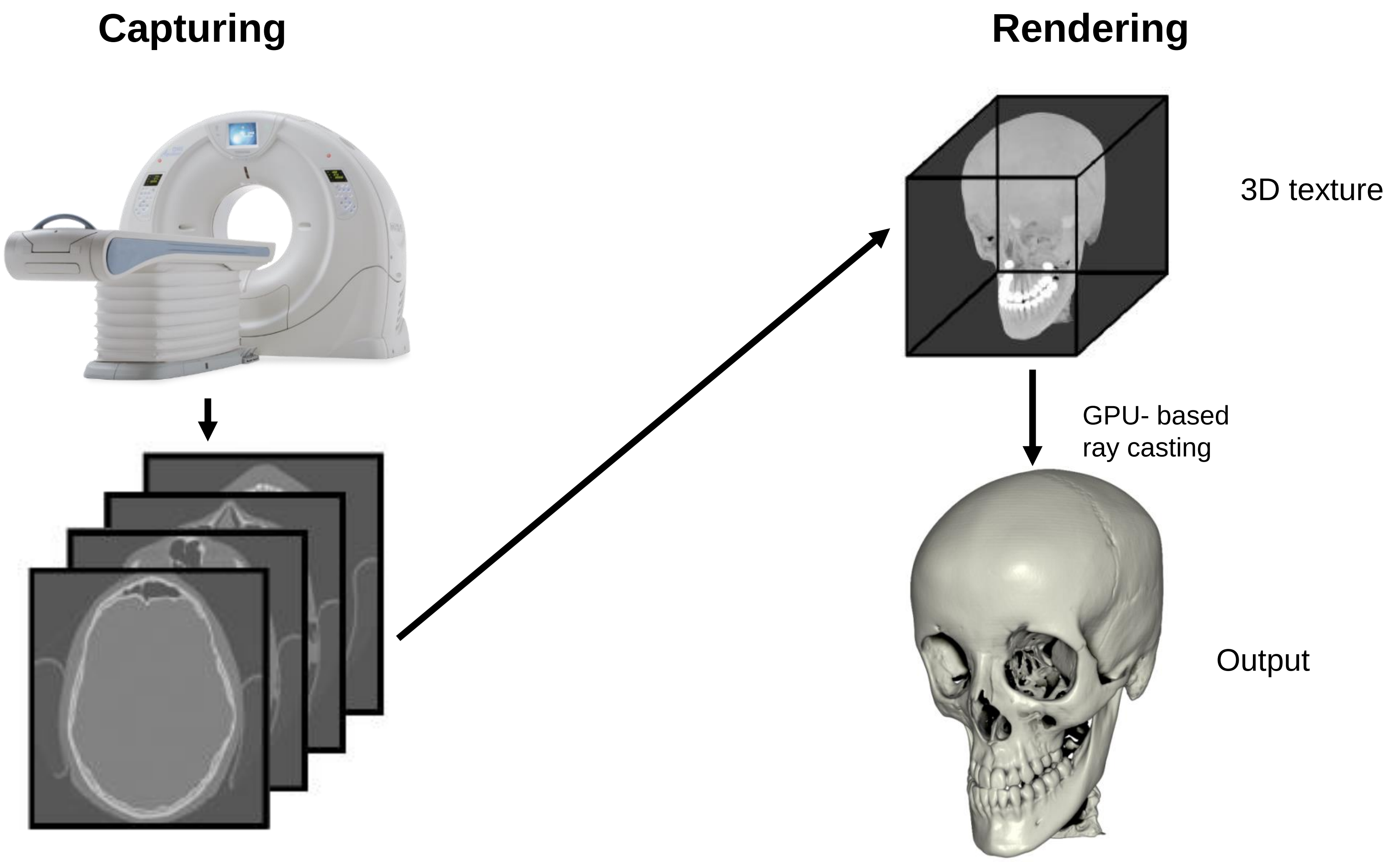
Scalable Mobile Visualization: Volumetric Data

Pere-Pau Vázquez, UPC

Rendering Volumetric Datasets

- Introduction
- Challenges
- Architectures
- GPU-based ray casting on mobile
- Conclusions

Rendering Volumetric Datasets



Rendering Volumetric Datasets

- **Introduction**
  - Volume datasets
    - Sizes continuously growing (e.g. $>1024^3$)
      - Complex data (e.g. 4D)
  - Rendering algorithms
    - GPU intensive
    - State-of-the-art is ray casting on the fragment shader
  - Interaction
    - Edition, inspection, analysis, require a set of complex manipulation techniques

Rendering Volumetric Datasets

- **Desktop vs mobile**
  - Desktop rendering
    - Large models on the fly
    - Huge models with the aid of compression/multiresolution schemes
  - Mobile rendering
    - Standard sizes (e.g. 512^3) still too much for the mobile GPUs
    - Rendering algorithms GPU intensive
      - State-of-the-art is GPU-based ray casting
    - Interaction is difficult on a small screen
      - Changing TF, inspecting the model…

Rendering Volumetric Datasets

- **Challenges on mobile:**
  - Memory:
    - Model does not fit into memory
      - Use client server approach / compress data
  - GPU capabilities:
    - Cannot use state of the art algorithm (e.g. no 3D textures)
      - Texture arrays
  - GPU horsepower:
    - GPU unable to perform interactively
      - Progressive rendering methods
  - Small screen
    - Not enough details, difficult interaction

Rendering Volumetric Datasets

- **Mobile architectures**
  - Server-based rendering
  - Hybrid approaches
  - Pure mobile rendering

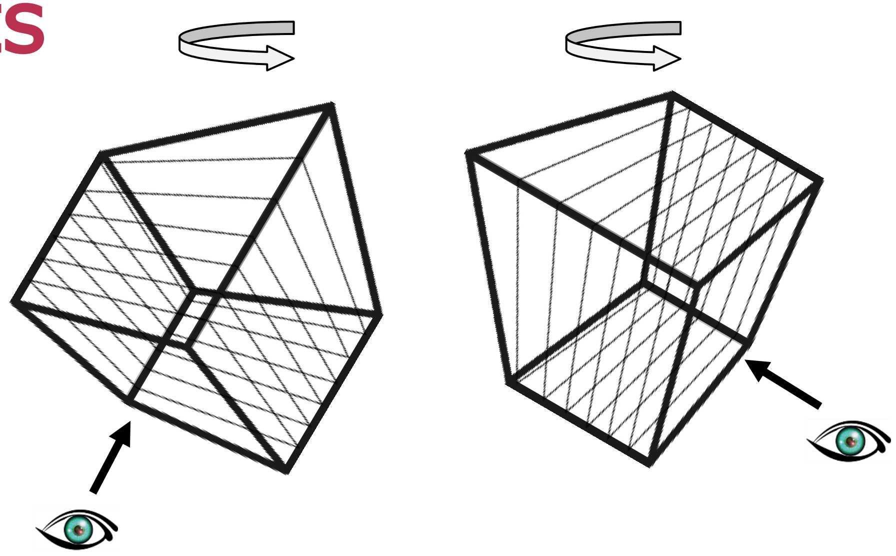
  - Server-based and hybrid rely on high bandwidth communication

Rendering Volumetric Datasets

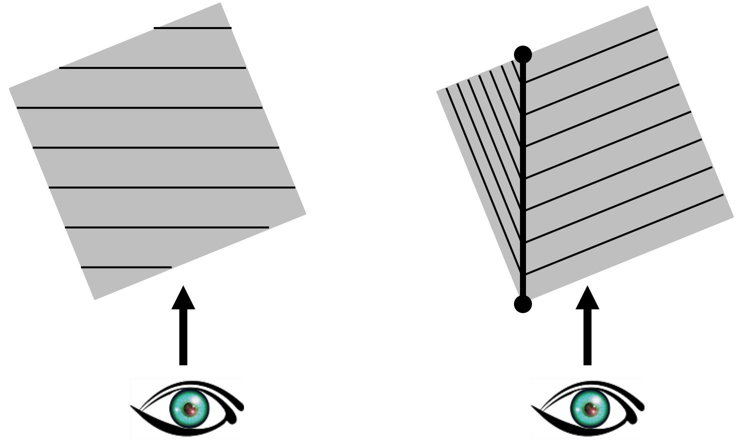
- **Pure mobile rendering**
  - Move all the work to the mobile
  - Nowadays feasible

- **Direct Volume Rendering on mobile. Algorithms**
  - Slices
  - 2D texture arrays
  - 3D textures

Rendering Volumetric Datasets



- **Slices**
  - Typical old days volume rendering
    - Several quality limitations
    - Subsampling & view change
  - Improvement: Oblique slices [Kruger 2010]

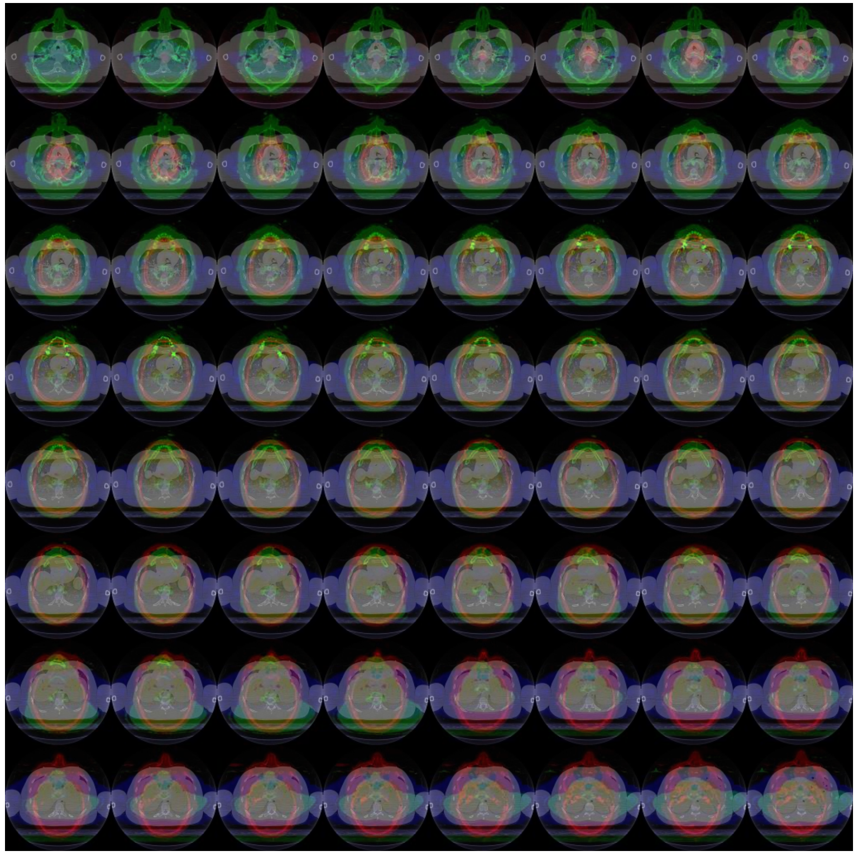


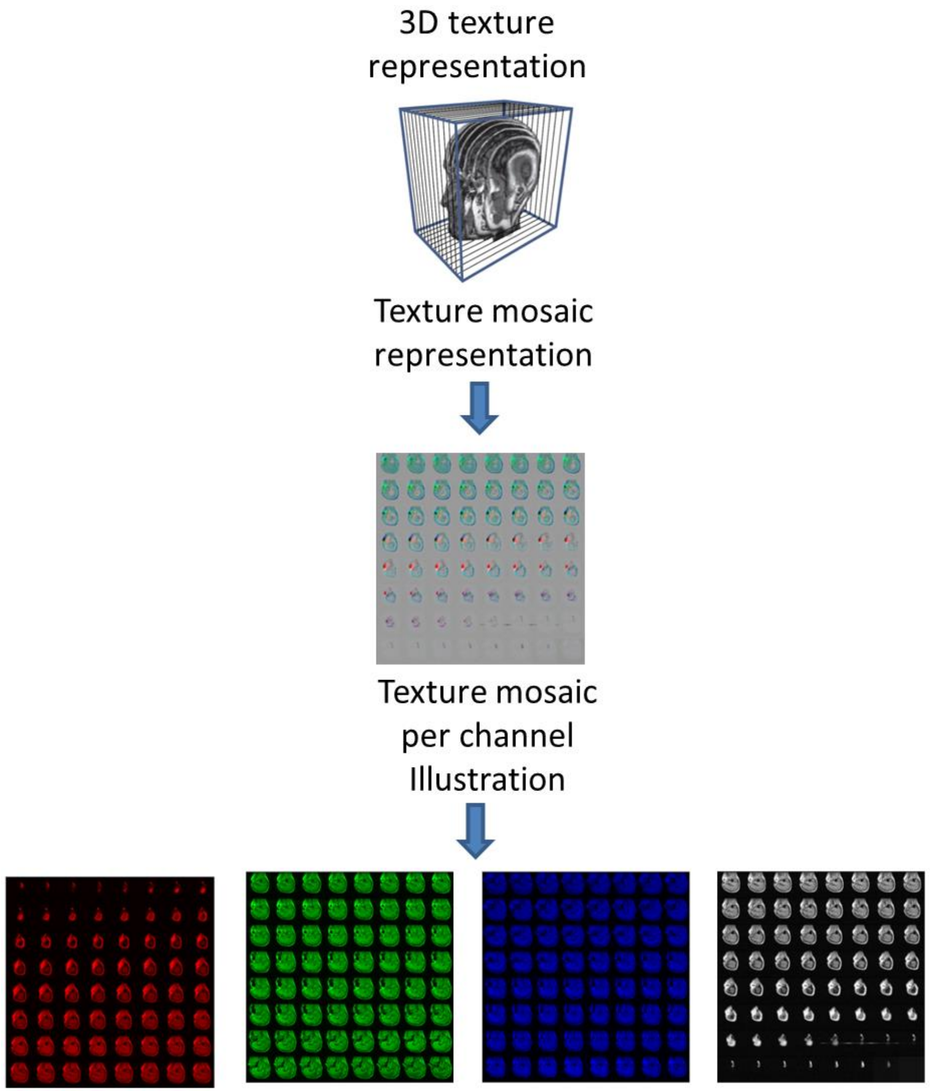
Rendering Volumetric Datasets

- **2D texture arrays + texture atlas [Noguera et al. 2012]**
  - Simulate a 3D texture using an array of 2D textures
  - Implement GPU-based ray casting
    - High quality
    - Relatively large models
    - Costly
    - Cannot use hardware trilinear interpolation

Rendering Volumetric Datasets

- **2D texture arrays + texture atlas**





Rendering Volumetric Datasets

- **2D texture arrays + compression [Valencia & Vázquez, 2013]**
  - Increase the supported sizes
  - Increase framerates

Compression format	Compression ratio	RBA format	RGBA format	GPU support	Overall performance	Overall quality
ETC1	4:1	Yes	No	All GPUs	Good (RC)	Good
PVRTC	8:1 and 16:1	Yes	Yes	PowerVR	Not so good	Bad
ATITC	4:1	Yes	Yes	Adreno	Good (RC)	Good

Rendering Volumetric Datasets

- **2D texture arrays + compression**
  - ATITC: improves performance from 6% to 19%. With an average of 13.1% and a low variance of performance.
  - ETC1(-P): improves performance from 6.3% to 69.5%. With an average of 32.6% and the highest variance of performance.
  - PVRTC-4BPP: improves performance from 4.7% and 36.% and PVRTC-2BPP: from 9,5% to 36,5%. The average performance of both methods is ~15% with high variance.

Rendering Volumetric Datasets

- **2D texture arrays + compression**
  - Ray-casting: gain performance in average of 33%.
  - Slice-based: gain performance in average o f 8%.
  - Ray-casting frame rates are better in all cases compared to slice-based.

Rendering Volumetric Datasets

- **2D texture arrays + compression**

Uncompressed

Compressed with ATI-I

Compressed with ETC1-P

Rendering Volumetric Datasets

- **2D texture arrays + compression**

Uncompressed

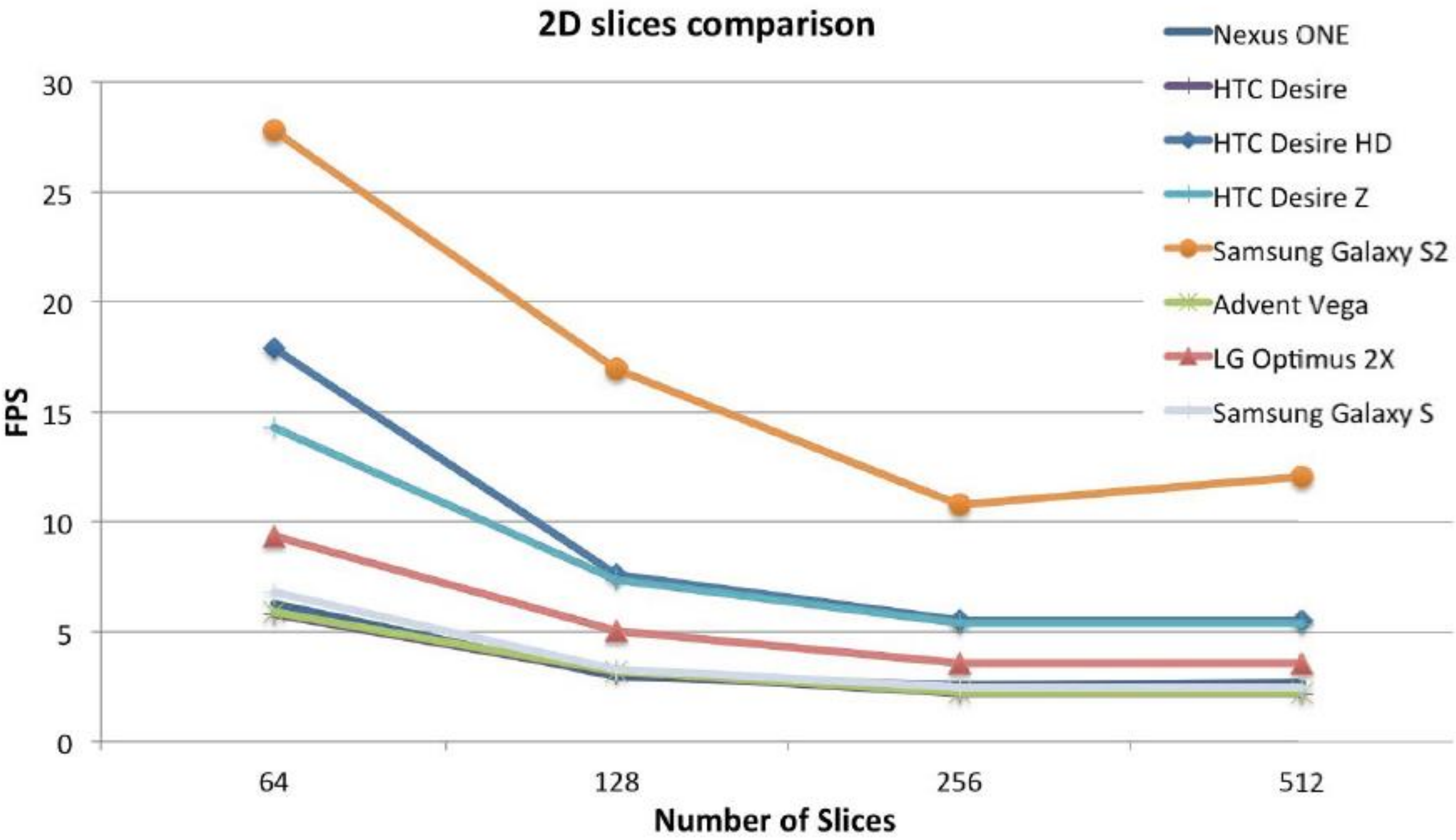
Compressed with PVRTC-4BPP

Compressed with PVRTC-2BPP

Rendering Volumetric Datasets

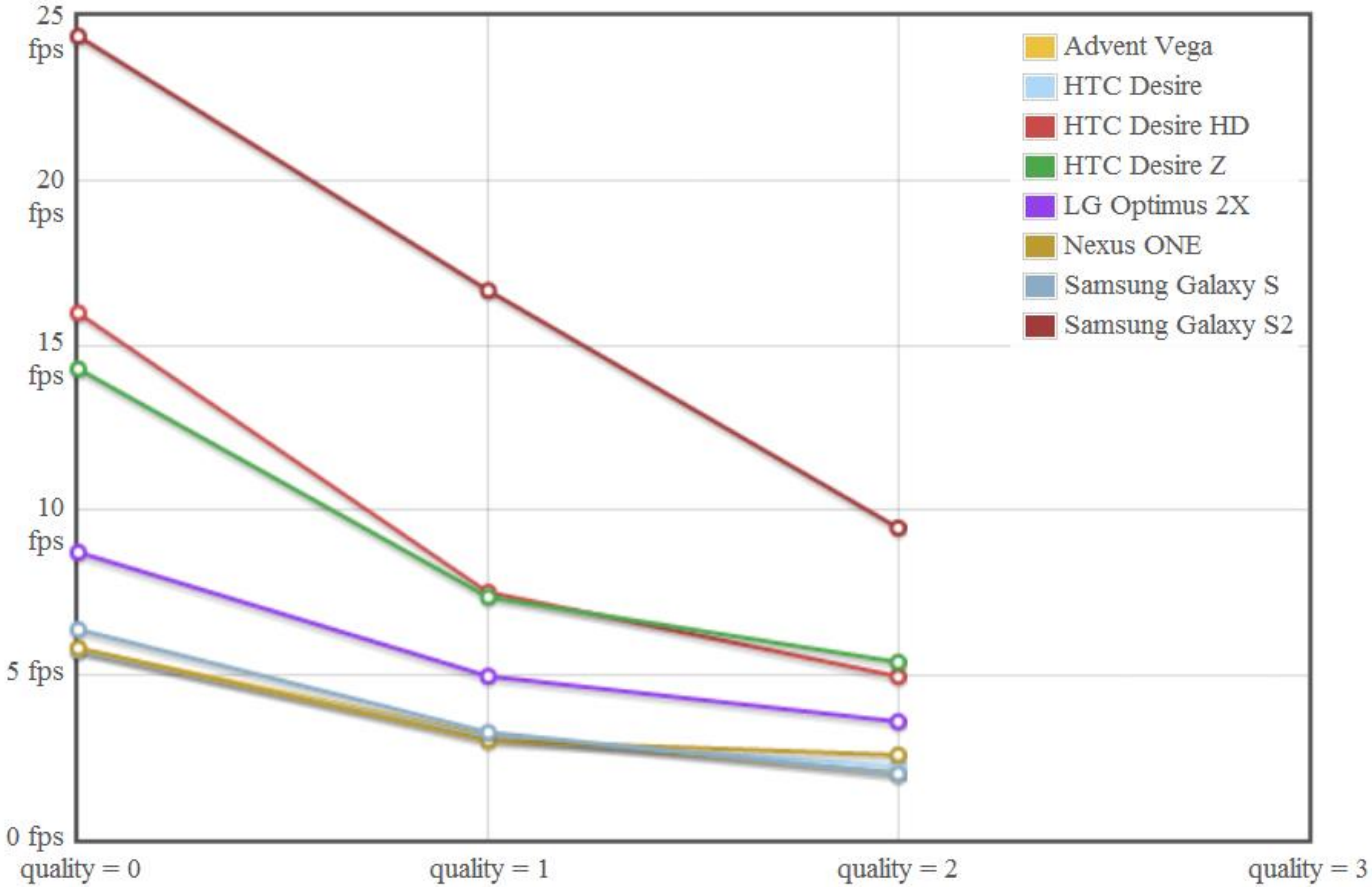
- **3D textures [Balsa & Vázquez, 2012]**
  - Allow either 3D slices or GPU-based ray casting
  - Initially, only a bunch of GPUs sporting 3D textures (Qualcomm's Adreno series >= 200)
  - Performance limitations (data: 256^3 – screen resol. 480x800)
    - 1.63 for 3D slices
    - 0.77 fps for ray casting

Rendering Volumetric Datasets



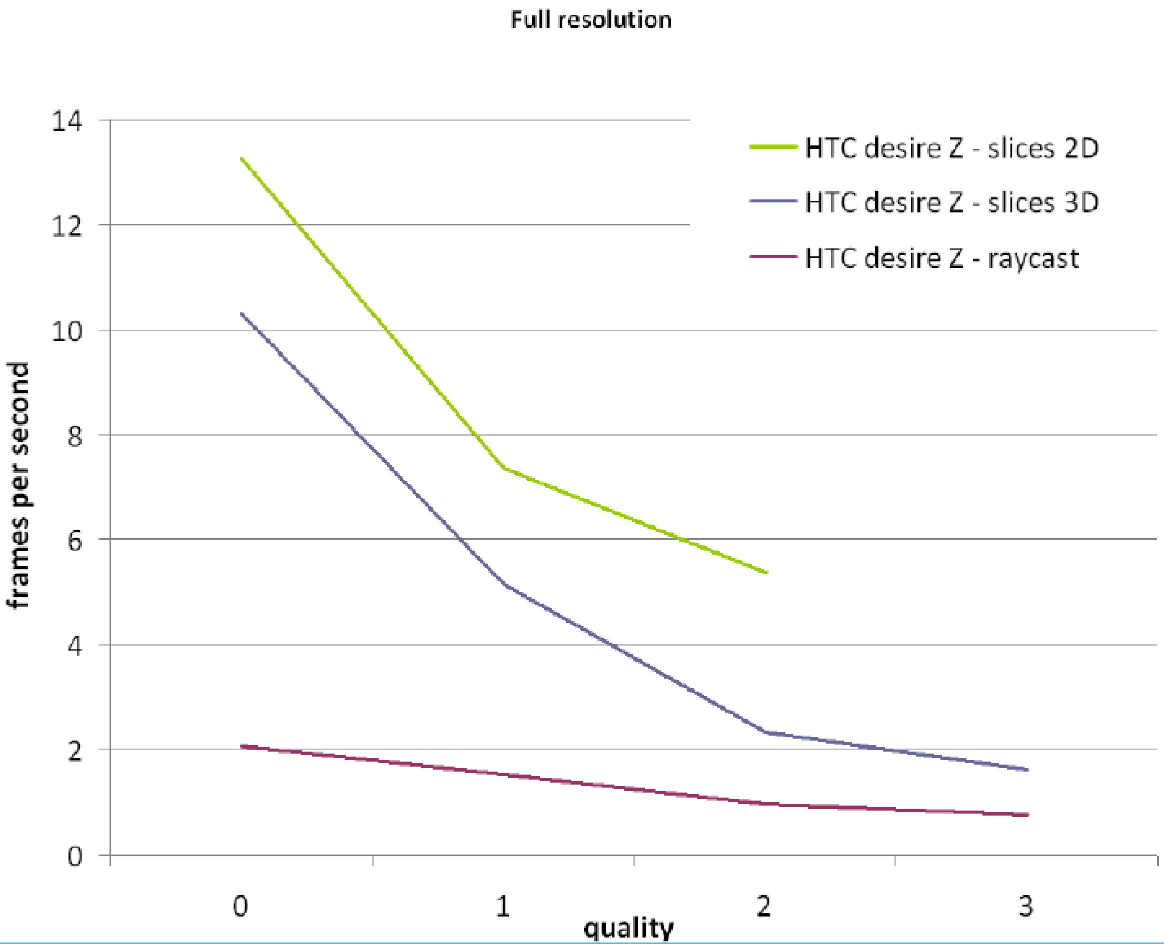
Rendering Volumetric Datasets

- **2D slices**



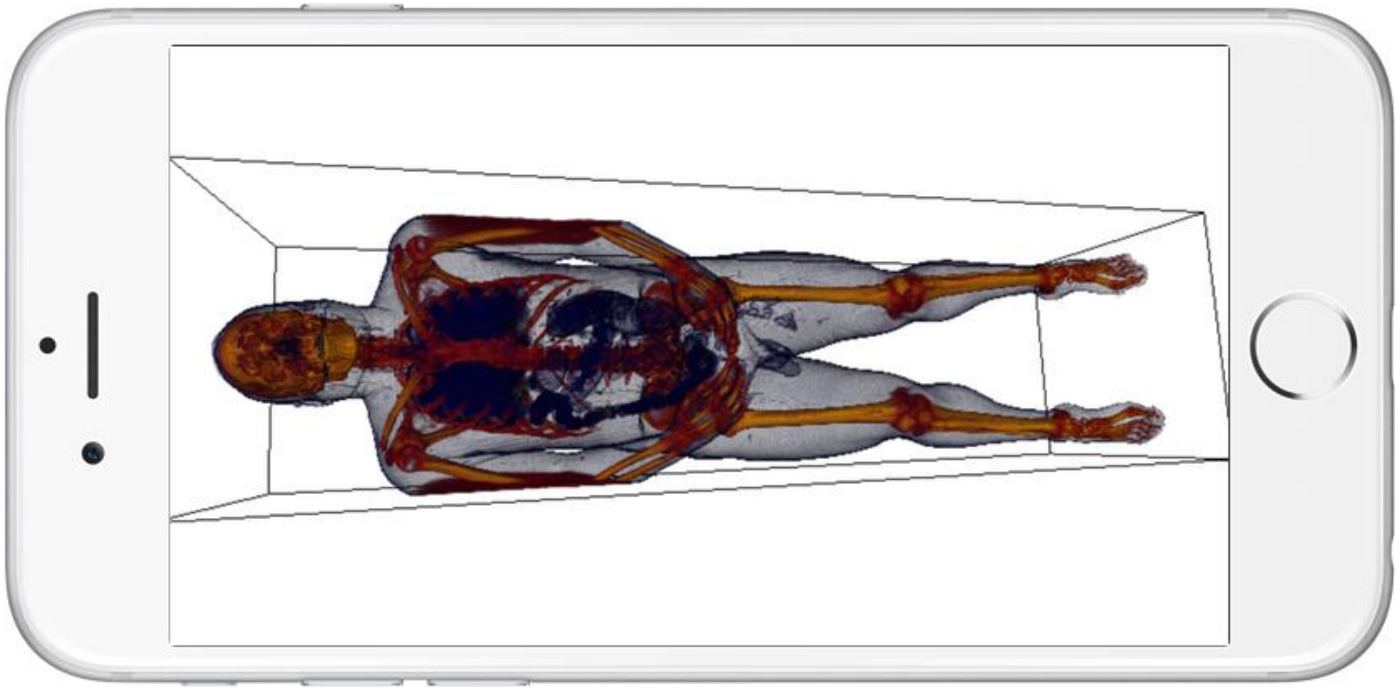
Rendering Volumetric Datasets

- **2D slices vs 3D slices vs raycasting**



Rendering Volumetric Datasets

- **Using Metal on an iOS device [Schiewe et al., 2015]**



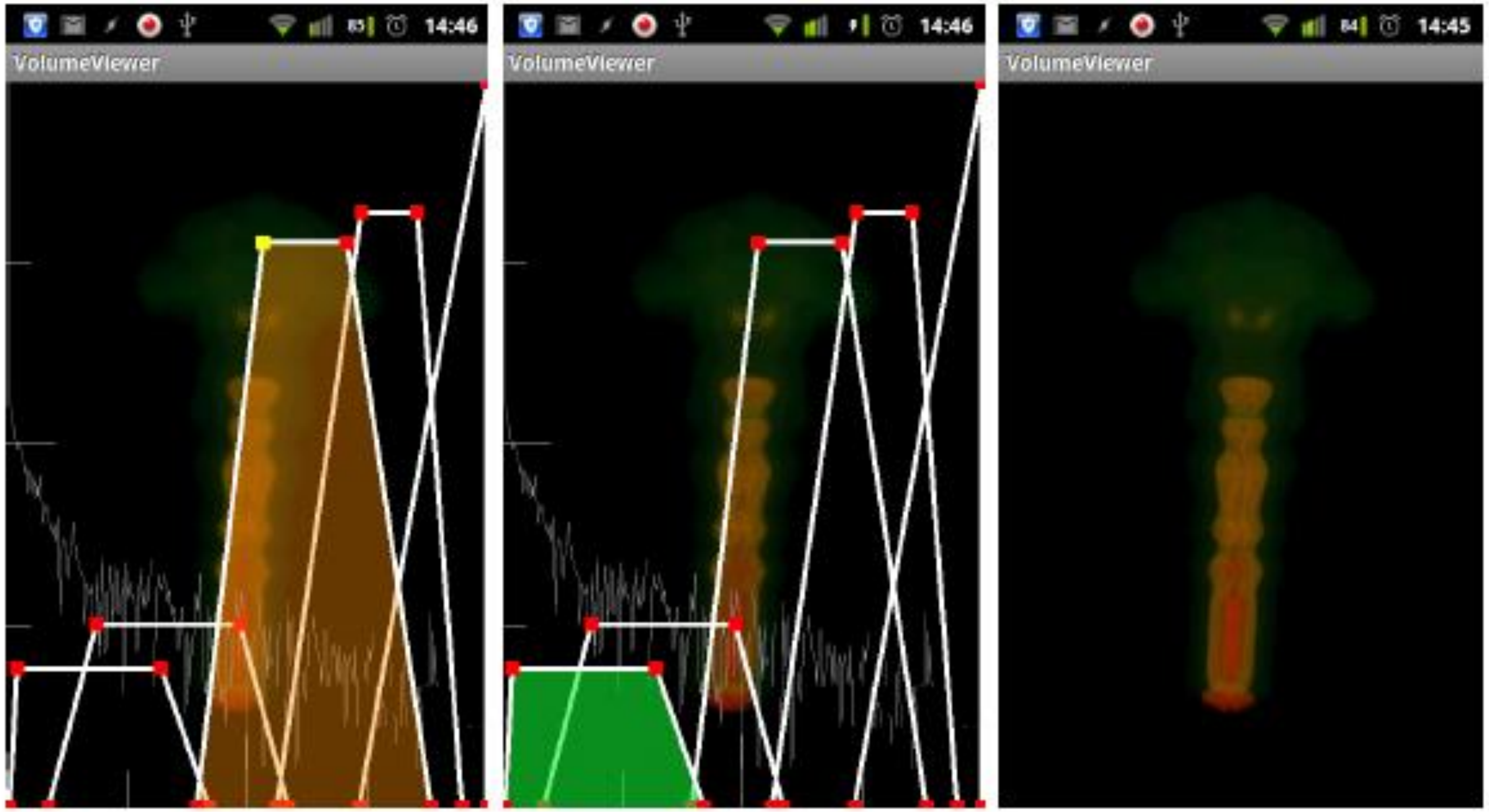
Taken from [Schiewe et al., 2015]

Volume data. GPU ray casting on mobile

- **Using Metal on an iOS device [Schiewe et al., 2015]**
  - Standard GPU-based ray casting
  - Provides low level control
  - Improved framerate (2x, to a maximum of 5-7 fps) over slice-based rendering
  - Models noticeably smaller than available memory (max. size was 256^2x942)

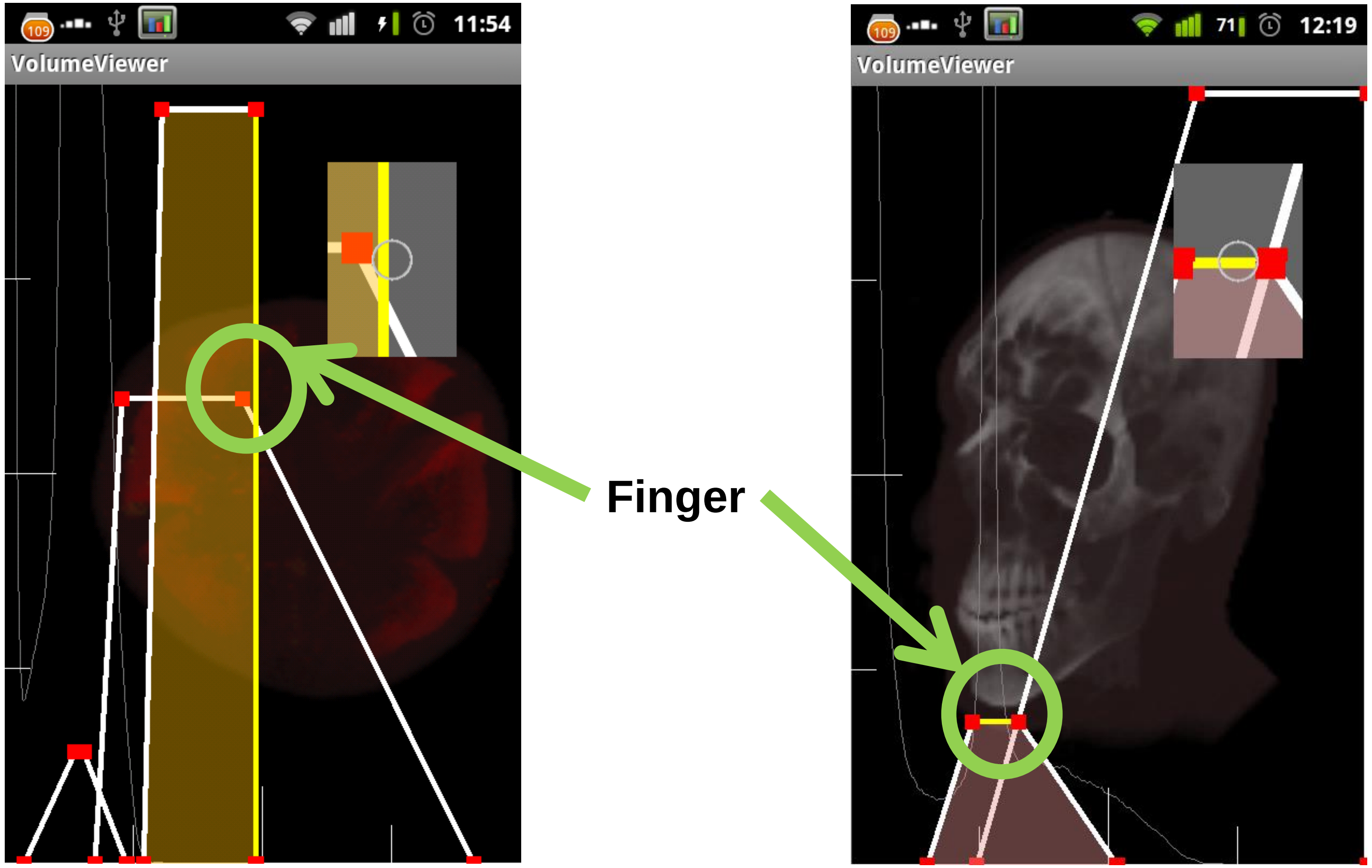
Rendering Volumetric Datasets

- **Challenges: Transfer Function edition**



Rendering Volumetric Datasets

- **Challenges: Transfer Function edition**



Rendering Volumetric Datasets

- **Conclusion**
  - Volume rendering on mobile devices possible but limited
    - Can use daptive rendering (half resolution when interacting)
  - 3D textures in core GLES 3.0
    - Still limited performance (~7fps…)
  - Interaction still difficult
  - Client-server architecture still alive
    - Can overcome data privacy/safety & storage issues
    - Better 4G-5G connections
    - …

Next Session

MOBILE METRIC CAPTURE AND RECONSTRUCTION