Progressive ray-casting volume rendering with WebGL for visual assessment of air void distribution in quality control

Ander Arbelaiz Aranzasti

Aitor Moreno Guerrero
Iñigo Barandiaran
Alejandro García-Alonso
OUTLINE

1. Introduction
   – Context
   – Goal & motivation
   – Challenges
2. State of the art
3. Contributions
4. Use case
5. Results
6. Conclusions
Context

- Volumetric data: 3D data discretized to cubic elements.
- Direct volume rendering
Goal & motivation

• Vicomtech is a research institute located in the north of Spain
  – Collaborating with local companies
    • Sariki Metrology

• Does a web based volume rendering approach complement a CT-analysis report?
  – Create a web based visualization than can be use to interactively asses the air void distribution of a plastic part
  – Easily distribute / share the volume data output between peers
  – Make a volume visualization that can work in the majority of devices
Challenges

• Ubiquitous volume rendering with WebGL
  – Achieve a high-quality rendering with large volume datasets
  – Render volumes in devices with low compute power
    • Tablets
    • Mobile phones
    • PCs with integrated GPU hardware
  – Support across multiple browser vendors
    • Improve the web compatible volume data representation structure

• Segmented volume rendering visualization
OUTLINE

1. Introduction
2. State of the art
   – Volume rendering
   – Ray-casting
   – Non-destructive CT quality control
3. Contributions
4. Segmentation & Analysis
5. Results
6. Conclusions
Direct volume rendering

- Different volume rendering techniques can be found in the literature
  - Shear-warp
  - Splatting
  - Cell projection
  - Texture slicing
  - Ray-casting
Web based volume rendering

- Moeen and Feng (2012; 2012b; 2012a)
- Congote et al. (2012; 2012)
- Movania et al. (2014)
- Noguera and Jiménez (2016)
- Lesar et al. (2018)
- Noguera et al. (2012; 2012)
- Yang et al. (2015)
- Arbelaz et al. (2016b,a, 2017a,c,b)
- Moreno et al. (2014)
- Mwalongo et al. (2018)

- Inside volume (2015a)
- MPR (2016b; 2016a)
- Volume + polygonal (2015b)
- Clip capping (2017)

X3DOM Volume Rendering development
Non-destructive CT volumetric quality inspection

• X-ray Computed Tomography (X-ray CT) is one of techniques being used to non-destructively inspect the inside of objects.
  – Inspection of parts
  – Reverse engineering
  – Analysis geomaterials
  – Additive manufacturing (AM)

• The characterization and dimensional measurement are possible with current CT technology
OUTLINE

1. Introduction
2. State of the art
3. Contributions
   - Web compatible volume data structure
   - Progressive volume rendering
4. Use case
5. Results
6. Conclusions
ImageTextureAtlas

- **OpenGL ES 2.0 for the Web**
  - Major drawback: WebGL 1.0 does not support 3D textures.
RGBAAP texture atlas

- \( k_1 = [1, 2, \ldots, n_c] \)
- \( k_2 = [0, 1, \ldots, n_c - 1] \)
- \( s_0 = \lfloor z \cdot n_s \rfloor \)
- \( c_1 = \text{mod}(s_0, n_c) \)
- \( c_2 = \text{mod}(s_1, n_c) \)
- \( s_2 = s_1 \cdot \frac{1}{n_c} \)
- \( s_1 = s_0 \cdot \frac{1}{n_c} \)

- \( d_{x1} = s_1 \cdot n_x - (\lfloor s_1 \cdot n_x \rfloor) \)
- \( d_{y1} = (\lfloor s_1 \cdot n_x \rfloor) \cdot n_y \)
- \( d_{x2} = s_2 \cdot n_x - (\lfloor s_2 \cdot n_x \rfloor) \)
- \( d_{y2} = (\lfloor s_2 \cdot n_x \rfloor) \cdot n_y \)

- \( \tilde{t}_1 = (d_{x1}, d_{y1}) + (x \cdot n_x, y \cdot n_y) \)
- \( \tilde{t}_2 = (d_{x2}, d_{y2}) + (x \cdot n_x, y \cdot n_y) \)

- \( \tilde{d}_1 = \text{texture2D}(\tilde{t}_1) \)
- \( \tilde{d}_2 = \text{texture2D}(\tilde{t}_2) \)

- \( \tilde{a}_1 = \text{step}(k_1, c_1) \)
- \( \tilde{a}_2 = \text{step}(k_1, c_2) \)
- \( \tilde{b}_1 = \text{step}(k_2, c_1) \)
- \( \tilde{b}_2 = \text{step}(k_2, c_2) \)

- \( f_1 = \tilde{b}_1 - \tilde{a}_1 \)
- \( f_2 = \tilde{b}_2 - \tilde{a}_2 \)

- \( r = \text{mix}(\tilde{d}_1 \cdot f_1, \tilde{d}_2 \cdot f_2, (z \times n_s) - s_0) \)
Single channel vs. Multi channel ImageTextureAtlas
Progressive volume rendering

• Advantages of progressive rendering
  – Prevents the application from stalling
  – Preserves user interaction
  – Distributes the rendering task over subsequent rendering frames after every user interaction
  – The rendering can be stopped at any time

• Our approach distinguishes two phases:
  – User interaction: Real-time rendering
  – No user interaction: Progressive refinement over subsequent frames
Progressive rendering (User interaction)

• When the user interacts with the virtual scene
  – Pan
  – Rotate
  – Zoom, etc.

• A real-time volume rendering algorithm is employed
  – Single-pass volume rendering algorithm
  – Fixed number of samples per ray
  – Fast, low quality rendering pass
Slabs vs. sampling steps

$T_{entry}$ $s_1$ $s_{n-1}$ $T_{exit}$

slab 0

slab $N$
Progressive rendering (No user interaction)

• An iterative high quality rendering process is initiated
  – The details of the volume data will be refined over time in the view direction
• The casted rays are discretized into smaller segments (slabs)
  – Each slab is an interaction in the progressive rendering
• For each iteration, three steps are performed
  1. **First pass (current slab):** High quality render into a texture ($T_{\text{high}}$) by sampling a slab with a fixed number of steps.
  2. **Slab buffer (copy):** Perform a copy and blending of ($T_{\text{high}}$) to accumulate the result between iterations.
  3. **Second pass (remaining slabs):** Low quality render of the remaining slabs and blending with the accumulated results.
Progressive rendering over sequential frames

- **Initialize**
  - **SLAB 0**
    - High Quality Raycasting
    - Copy
    - Low Quality Raycasting
  - **IMG 0**

- **i-th SLAB**
  - High Quality Raycasting
  - Copy
  - Low Quality Raycasting
  - **IMG i-th**

- Clear
Progressive rendering over sequential frames II

25/150 slabs
50/150 slabs
100/150 slabs
150/150 slabs
Real-time vs. Progressive

plastic part (720x720x1770)  
80 steps

plastic part (720x720x1770)  
6000 steps
OUTLINE

1. Introduction
2. State of the art
3. Contributions
4. Use case
   - Dataset
   - Air void segmentation
   - Air void mapping to texture atlas
5. Results
6. Conclusions
Dataset

• Plastic injected mold part for the medical sector
• Plastic part produced in a pre-production stage for the calibration of the injection mold machine
  – Air voids are created inside the plastic part when the process is not correctly calibrated
• Industrial Nikon X-ray microCT used for the dataset acquisition
  – 1875 slices of 836 x 939 16bit (2.9GB)
• Processed offline to adequate the dataset for its visualization by making use of ROI and windowing
  – 1770 slices of 720 x 720 8bit (918.5 MB)
Air void segmentation

- Insight Segmentation and Registration Toolkit (ITK) was used to perform the air void segmentation
  - Otsu’s thresholding for background extraction
  - Binary Fill Holes filter to identify holes in the binary image
  - Connected components filter to label each air bubble
Air void mapping to texture atlas

• The spatial volume each air void bubble (label) and a value is assigned to each label in function of its estimated volume size.
• A transfer function texture is generated to map each label value [0-255] into color.
OUTLINE

1. Introduction
2. State of the art
3. Contributions
4. Use case
5. Results
   - Performance
   - Demo
6. Conclusions
Air void quality control inspection
Performance

PC: Intel i5-6500 – 8GB RAM – Nvidia GTX 960 2GB
MiPad 1: Nvidia Tegra K1 – 2GB RAM
Mi5s: Snapdragon 821 – 3GB RAM – Adreno 530

(720x720x1770) 6000 steps – single texture
Performance

PC: Intel i5-6500 – 8GB RAM – Nvidia GTX 960 2GB
MiPad 1: Nvidia Tegra K1 – 2GB RAM
Mi5s: Snapdragon 821 – 3GB RAM – Adreno 530

(720x720x1770) 6000 steps – two textures
OUTLINE

1. Introduction
2. State of the art
3. Contributions
4. Use case
5. Results
6. Conclusions
   - Summary
   - Future work
Summary

• We have presented...
  – an ubiquitous progressive ray-casting algorithm
  – an improved data structure RGBA ImageTextureAtlas to store up to 4 times more data in a multi-channel 2D texture

• The presented contribution has been demonstrated under an industrial use case
  – Quality control of a plastic injected mold part

• This approach...
  – is applicable to other domains
  – allows the volume visualization of large datasets
  – can be adapted to the targeted device’s compute power
Future Work

• Transition to upcoming 3D graphic APIs WebGL 2.0 and WebGPU
• Out of core volume rendering for very large volume datasets
• Volume rendering pre-integration to reduce the number of samples
Thank you!

Any Questions?

aarbelai$@vicomtech.org