Real-Time Ray Tracing on Head-Mounted-Displays for Advanced Visualization of Sheet Metal Stamping Defects
Sheet Metal Forming
Applications in Manufacturing

• Sheet metal forming summarizes a number of metal forming techniques in mass manufacturing:
  • Stamping
  • Punching
  • Blanking
  • Embossing
  • Bending
  • Coining

• Most common raw materials to form are sheet metal, other applications include materials such as polysterene
Sheet Metal Forming

Structural Defects

• Typical **structural** defects in sheet metal forming manufacturing
  • Cracking, splitting, ...
  • Springback
  • Wrinkles
  • Thinning / thickening

• Numerical solver solutions predict defects in stamped parts
  • Highly accurate simulation of the stamping process and die setup
  • Structural defects are clearly quantifiable
  • No physical prototype required
Sheet Metal Forming

Cosmetic Defects

• Aesthetics of remaining **cosmetic** defects hard to estimate from numerical analysis
• No general rule to automatically qualify based on numerics
  • Acceptance criteria vary from manufacturer to manufacturer and from model to model
  • Process builds on engineer’s expertise and experience
• Interpretation of the visual impact of a defect is highly subjective
• Further steps such as assembly, coating, paint, affect the visual impact of a defect
Visualization of Cosmetic Stamping Defects

Physical Prototypes

• Prototype parts verified using a manual review process
  • Stamped part removed from the die, trimmed, put up on a holder, and brought to a mirror-like finish
  • Use of special lighting and a combination of viewpoint and interaction with the part to evaluate visual defects

• Try-out at time where changes to die and process are costly
  • Goal: zero physical prototypes
  • Virtually produce and inspect perceived quality – before try-out
Visualization of Cosmetic Stamping Defects

Virtual Inspection

• Simulation of reflection lines
  • Reflection mapping (e.g., [Sussner et al. 2004])
  • Real-time ray tracing (e.g., [Wald et al. 2006])
    • So far limited to desktop applications

• New GPU Developments
  • RT Cores / Turing, DXR, Vulkan
  • Enable ray tracing applications in VR
    • Simulate accurate reflections at high resolutions and frame rates
    • Recreate whole physical workflows
HMD Ray Tracing Rendergraph

- Communication with HMD
  - Part of renderer pipeline
  - Implemented as render passes
  - Latency reduction
Vision Matched Rendering

Skipping invisible pixels

- Hidden Area Mesh
  - Provided by OpenVR SDK
  - Defines visible area within image
  - Depends on HMD optics (e.g., lens distortion)
  - Roughly circular on Vive Pro

- Exploit in OptiX ray generation program
  - Avoid visibility computation and shading
  - Skip computing pixels outside disc
  - Disc diameter about 80% of box width
Aliasing

- Causes strong flickering
- More objectionable than stutter
- Particularly visible in reflections on curved surfaces

AA techniques implemented:

- **Basic oversampling**
  - Render at higher resolution
  - Reduce overall flicker

- **Foveated oversampling**
  - More samples at image center
  - Reduce flicker of reflections

- **FXAA**
  - Filtering post-process
  - Smooth edges
Foveated Rendering
Experimental Variable Rate Sampling

- Foveated oversampling
  - 50% radius of visible area disc
  - Dithering to avoid sharp transition
  - 4 samples / pixel

- Still strong flicker
- Frame rate reduced to ~10 fps
- Need more than 16 for significant impact
- Not practical

Green: 1 sample / pixel
Red: 4 samples / pixel
Anti-Aliasing
Fast ApproXimate Anti-Aliasing (FXAA)

- FXAA [Lottes 2009]
  - Fullscreen post process (GLSL)
  - Edge-aware low-pass filter

- Basic algorithm
  - Detect edges based on contrast difference
  - Approximate luminance gradient
  - Filter along axis perpendicular to gradient
Anti-Aliasing

Fast ApproXimate Anti-Aliasing (FXAA)

- FXAA post-process
  - Works well for edges
  - Does not help for light reflections
  - Smooths light fragments but does not merge them

Without FXAA

With FXAA
System Setup

GPU
• NVIDIA Quadro RTX 8000
  • Turing architecture
  • 4608 CUDA cores
  • 576 Tensor cores
  • 72 RT cores
  • 48 GB device memory

HMD
• HTC Vive Pro
  • 1440 x 1600 pixels per eye
  • 90 Hz refresh rate
  • 110 degrees field of view
Performance Results

- Demo Scene
  - 2.2 Mio triangles
  - 1000 individual objects
  - 3 levels of reflection
  - 1 point light
  - Baked ambient occlusion

- Ray tracing backend provides
  - 2016 x 2240 pixels per eye
    - Super-sampled anti-aliasing
    - 100% resolution in SteamVR settings
    - Filtering done by HMD
  - 20 – 45 fps
    - HMD performs asynchronous reprojection to reduce judder
Experiences and Conclusion

• Full-frame Whitted-style ray tracing feasible on HMDs with a single GPU
  • Usable in VR inspection scenarios in virtual prototyping

• Raw frame rate not as important as you might think
  • Asynchronous reprojection works well for framerates above 20 fps
  • Significant change in stutter only perceived below 20 fps or over 90 fps

• Flicker resulting from aliasing most significant issue
  • Noise-type aliasing in reflections
  • Edge-filtering not sufficient
  • Vision matched oversampling still too costly