Lecture 38:

Virtual and Augmented Reality III & Course Conclusion

Computer Graphics and Imaging
UC Berkeley CS184/284A, Spring 2017
Thanks to all you students for your hard work and attention.

Thanks to our terrific GSIs, Lingqi, Austin, Kevin and Ben, for their care and dedication.
Announcements (4/24/2017)

Project presentations on Wednesday May 3, 2017

- Sign ups + details coming on Piazza
- Best projects will present in showcase Thu 5/4/17

Today:

- HKN Evaluations, final course lecture
- Note: no lecture on Friday

Wednesday:

- AR / VR Demos (3-5pm)
- Thanks to Austin Le, Isabel Zhang, and VR @ Berkeley
Wednesday: Live VR and AR Demos

Thanks to Berkeley VR Club
Learn more: vr.berkeley.edu

VR: HTV Vive

AR: Microsoft Hololens
Other Cool Classes in Visual Computing

- CS194-26  Image Manipulation and Computational Photography (Efros)
- UCBUGG  DeCal on 3D Modeling and Animation
- CS194-8  Advanced Digital Animation (Garcia+)
- CS294-137  Theory and Applications of Virtual Reality & Immersive Computing (O’Brien)
- CS294-127  Computational Imaging (Ng)
- CS280  Computer Vision
- EE118  Intro to Optical Engineering (Waller)
(U)GSIs and Independent Study

CS184 will be taught in Spring 2018 (Ng)

- Email if you are interested in a U(GSI)
- Give back + help us continue improving the class

Students who did well in 184 have great preparation for independent study or research in visual computing

- Come talk to me, James O’Brien, Alyosha Efros, Jonathan Ragan-Kelley
HTC Vive Tracking System ("Lighthouse")

- Structured light transmitter
- Photodiode arrays on headset and hand-held controllers
Vive Headset & Controllers Have Array of IR Photodiodes

(Prototype) Headset and controller are covered with IR photodiodes
HTC Vive Structured Light Emitter ("Lighthouse")

Light emitter contains array of LEDs (white) and two spinning wheels with lasers. Sequence of LED flash and laser sweeps provide structured lighting throughout room.

Credit: Gizmodo: http://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768
HTC Vive Tracking System

For each frame, lighthouse does the following:

- LED pulse, followed by horizontal laser sweep
- LED pulse, followed by vertical laser sweep

Each photodiode on headset measures time offset between pulse and laser arrival

- Determines the x and y offset in the lighthouse’s field of view
- In effect, obtain an image containing the 2D location of each photodiode in the world
  - (Can think of the lighthouse as a virtual “camera”)

CS184/284A  Ren Ng
HTC Vive Tracking System ("Lighthouse")

Credit: rvdm88 / youtube. https://www.youtube.com/watch?v=J54dotTt7k0
Tracking Summary

Looked at three tracking methods

- Camera on headset + computer vision + gyro
- External camera + marker array on headset
- External structured light + sensor array on headset

3D tracking + depth sensing an active research area

- SLAM, PTAM, DTAM...
- Microsoft Hololens, Google Tango, Intel Realsense, ...
Rendering Challenges in VR
The goal of a VR graphics system is to achieve “presence”, tricking the brain into thinking what it is seeing is real.

Achieving presence requires an exceptional low-latency system:

- What you see must change when you move your head!
- End-to-end latency: time from moving your head to the time new photons hit your eyes
  - Measure user’s head movement
  - Update scene/camera position
  - Render new image
  - Transfer image to headset, then transfer to display in headset
  - Actually emit light from display (photons hit user’s eyes)
- Latency goal of VR: 10-25 ms
  - Requires exceptionally low-latency head tracking
  - Requires exceptionally low-latency rendering and display
Thought Experiment: Effect of Latency

Consider a 1,000 x 1,000 display spanning 100° field of view
- 10 pixels per degree

Assume:
- You move your head 90° in 1 second (only modest speed)
- End-to-end latency of system is a slow 50 ms (1/20 sec)

Result:
- Displayed pixels are off by 4.5° ~ 45 pixels from where they would be in an ideal system with 0 latency

Example credit: Michael Abrash
Name of the Game, Part 2: High Resolution

iPhone 6: 4.7 in "retina" display:
- 1.3 MPixel
- 326 ppi → 57 ppd

Future “retina” VR display:
- 57 ppd covering 200°
- = 11K x 11K display per eye
- = 220 MPixel

Human: ~160° view of field per eye (~200° overall)
(Note: does not account for eye’s ability to rotate in socket)

Strongly suggests need for eye tracking and foveated rendering (eye can only perceive detail in 5° region about gaze point)

Eyes designed by SuperAtic LABS from the thenounproject.com
Foveated Rendering

Idea: track user’s gaze, render with increasingly lower resolution farther away from gaze point.

Three images blended into one for display.
Requirement: Wide Field of View

View of checkerboard through Oculus Rift (DK2) lens

Lens introduces distortion

- Pincushion distortion
- Chromatic aberration (different wavelengths of light refract by different amount)

Icon credit: Eyes designed by SuperAtic LABS from the thenounproject.com
Image credit: Cass Everitt
5 Getting Started

Your developer kit is unpacked and plugged in. You have installed the SDK, and you are ready to go. Where is the best place to begin?

If you haven’t already, take a moment to adjust the Rift headset so that it’s comfortable for your head and eyes. More detailed information about configuring the Rift can be found in the Oculus Rift Hardware Setup section of this document.

After your hardware is fully configured, the next step is to test the development kit. The SDK comes with a set of full-source C++ samples designed to help developers get started quickly. These include:

- **OculusWorldDemo** - A visually appealing Tuscany scene with on-screen text and controls.
- **OculusRoomTiny** - A minimal C++ sample showing sensor integration and rendering on the Rift (only available for D3DX platforms as of 0.4. Support for GL platforms will be added in a future release).

We recommend running the pre-built OculusWorldDemo as a first-step in exploring the SDK. You can find a link to the executable file in the root of the Oculus SDK installation.

5.1 OculusWorldDemo

![Software Compensation for Lens Distortion](image)

Step 1: Render scene using traditional graphics pipeline at full resolution for each eye

Step 2: Warp images in manner that scene appears correct after physical lens distortion

(Can use separate distortions to R, G, B to approximately correct chromatic aberration)

Image credit: Oculus VR developer guide
Challenge: Rendering via Planar Projection

Recall: rasterization-based graphics is based on perspective projection to plane
- Distorts image under high FOV, as needed in VR rendering
- Recall: VR rendering spans wide FOV

Pixels span larger angle in center of image
(lowest angular resolution in center)

Future investigations may consider: curved displays, ray casting to achieve uniform angular resolution, rendering with piecewise linear projection plane (different plane per tile of screen)

Image credit: Cass Everitt
Consider Object Position Relative to Eye

**Case 1: object stationary relative to eye:**
- (eye still and red object still)
- OR
- red object moving left-to-right and eye moving to track object
- OR
- red object stationary in world but head moving and eye moving to track object

**Case 2: object moving relative to eye:**
- (red object moving from left to right but eye stationary, i.e., it’s looking at a different stationary point in world)

Spacetime diagrams adopted from presentations by Michael Abrash
Eyes designed by SuperAtic LABS from the thenounproject.com
Effect of Finite Frame Rate and Latency: Judder

**Case 2: object moving from left to right, eye stationary**
*(eye stationary with respect to display)*

Continuous representation.

**Case 2: object moving from left to right, eye stationary**
*(eye stationary with respect to display)*

Light from display
(image is updated each frame)

**Case 1: object moving from left to right, eye moving continuously to track object**
*(eye moving relative to display!)*

Light from display
(image is updated each frame)

Explanation: since eye is moving, object’s position is relatively constant relative to eye (as it should be, eye is tracking it). But due discrete frame rate, object falls behind eye, causing a smearing/strobing effect (“choppy” motion blur). Recall from earlier slide: 90 degree motion, with 50 ms latency results in 4.5 degree smear.

Spacetime diagrams adopted from presentations by Michael Abrash
Reducing Judder: Increase Frame Rate

Case 1: continuous ground truth
red object moving left-to-right and
eye moving to track object
OR
red object stationary but head moving
and eye moving to track object

Light from display
(image is updated each frame)

Higher frame rate results in closer
approximation to ground truth

Spacetime diagrams adopted from presentations by Michael Abrash
Reducing Judder: Low Persistence Display

Case 1: continuous ground truth
- red object moving left-to-right and eye moving to track object
  OR
- red object stationary but head moving and eye moving to track object

Light from full-persistence display
- frame 0
- frame 1
- frame 2
- frame 3

Light from low-persistence display
- frame 0
- frame 1
- frame 2
- frame 3

Full-persistence display: pixels emit light for entire frame
Low-persistence display: pixels emit light for small fraction of frame
Oculus DK2 OLED low-persistence display
- 75 Hz frame rate (~13 ms per frame)
- Pixel persistence = 2-3ms

Spacetime diagrams adopted from presentations by Michael Abrash
Artifacts Due to Rolling OLED Backlight

Image rendered based on scene state at time $t_0$

Image sent to display, ready for output at time $t_0 + \Delta t$

“Rolling backlight“ OLED display lights up rows of pixels in sequence

- Let $r$ be amount of time to “scan out” a row
- Row 0 photons hit eye at $t_0 + \Delta t$
- Row 1 photos hit eye at $t_0 + \Delta t + r$
- Row 2 photos hit eye at $t_0 + \Delta t + 2r$

Implication: photons emitted from bottom rows of display are “more stale” than photos from the top!

Consider eye moving horizontally relative to display (e.g., due to head movement while tracking square object that is stationary in world)

Result: perceived shear!
Recall rolling electronic shutter effects on digital cameras.
Compensating for Rolling Backlight

Perform post-process shear on rendered image

- Similar to previously discussed barrel distortion and chromatic warps
- Predict head motion, assume fixation on static object in scene
  - Only compensates for shear due to head motion, not object motion

Render each row of image at a different time (the predicted time photons will hit eye)

- Suggests exploration of different rendering algorithms that are more amenable to fine-grained temporal sampling, e.g., ray caster? (each row of camera rays samples scene at a different time)
Increasing Frame Rate Using Re-Projection

Goal: maintain as high a frame rate as possible under challenging rendering conditions:

- Stereo rendering: both left and right eye views
- High-resolution outputs
- Must render extra pixels due to barrel distortion warp
- Many “rendering hacks” (bump mapping, etc.) are less effective in VR so rendering must use more expensive techniques

Researchers experimenting with reprojection-based approaches to improve frame rate (e.g., Oculus’ “Time Warp”)

- Render using conventional techniques at 30 fps, reproject (warp) image to synthesize new frames based on predicted head movement at 75 fps
- Potential for image processing hardware on future VR headsets to perform high frame-rate reprojection based on gyro/accelerometer
Near-Future VR Rendering System Components

- Low-latency image processing for subject tracking
- Massive parallel computation for high-resolution rendering
- Exceptionally high bandwidth connection between renderer and display: e.g., 4K x 4K per eye at 90 fps!
- High-resolution, high-frame rate, wide-field of view display
- In headset motion/accel sensors + eye tracker
- On headset graphics processor for sensor processing and re-projection
Activity in Image Capture of VR Content

Google’s JumpVR video: 16 4K GoPro cameras

Consider challenge of:
Registering/3D align video stream (on site)
Broadcast encoded video stream across the country to 50 million viewers

Lytro Immerge
A “dense light field camera array” pursuing 6 degree-of-freedom video for VR

Many, many others: Jaunt ONE, Vuze, Samsung Gear 360, Nokia Ozo, …
Summary

VR presents many new graphics challenges!

Tracking

- Head-pose tracking with high accuracy and low latency

Rendering

- Low-latency, high resolution & frame-rate, wide field of view, …

Displays

- Going beyond 2D panel displays: HMDs, curved displays, …

Capture

- How to capture video for VR displays?
Acknowledgments

Thanks to Kayvon Fatahalian, James O’Brien, Brian Wandell and Gordon Wetzstein for lecture slides and resources!