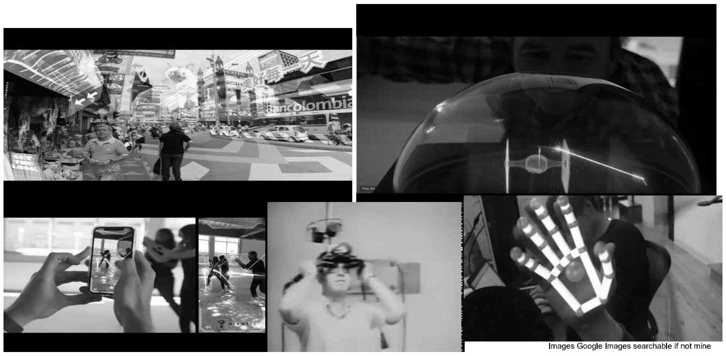
Intro to Augmented Reality



Images Google Images searchable if not mine

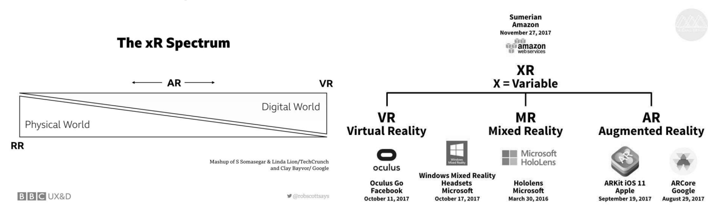
Survey & A4

- www.tinyurl.com/xrprojectssurvey
- Need to know by Wednesday if you want food on Thursday
- A4 will be easy; assign today and due Monday after spring break

Topics

- Displays & Lenses
- Optics & Distortion
- Tracking
- Localization
- 3D Reconstruction

Recall: XR spectrum



- Citations in images. Can also reverse image search
- Can argue that devices like Vives and some Oculuses are MR b/c they use real-world tracking constraints, muddying the waters
- Programming more or less the same for entire spectrum

Pure AR vs. MR

- Pure AR has no real 3D understanding of physical environment (PE)
  - Relies almost entirely on **markers** & calibration
  - Almost entirely **computer vision (CV)** with **single camera views**
  - Since user doesn't have a 3D view of VE, minor errors in calibration usually unnoticeable
  - Usually **no interaction between VE & PE** beyond markers
  - APIs like ARCore, ARKit, etc.
  - Almost always video-passthrough w/ single camera (like mobile phones)
- MR
  - Mix of understanding of VE and PE
  - 3D understanding of PE (**spatial understanding**) used to place virtual objects
    - or PE is used to create entire VE to allow virtual mechanics
  - Relies heavily on **3D reconstruction** methods and **depth sensors/structure from motion**
    - Tracking methods we talk about later apply: active/passive markers & markerless
- Where line is blurred between modern VR & MR:
  - **Inside-out tracking**: tracking with cameras on HMD instead of external sensors
  - Oculus Quest & inside-out tracked devices which use PE to limit VE & can track real objects like hands. How much needs to be reconstructed vs simple 2D CV?

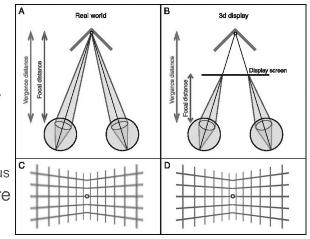
For OUR purposes (& simplicity)

- "AR" will just be any situation where the real world is **VISIBLE** (tracked or not) for the purpose of this discussion
- Helps avoid the ambiguity of AR/MR (e.g. HoloLens and Quest are technically both MR…)

Sutherland '68 was technically an MR display

Fundamental problem in XR displays: Vergence-accommodation conflict

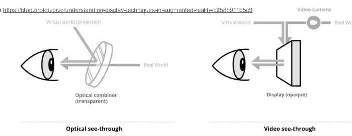
- We have a screen in front of the eyes that they want to focus on… but we want to convince the eyes that they're looking at something far away
  - **Vergence**: how far the object is we want them to focus on
  - **Accommodation**: the actual real object the eyes stop their focus at (the screen) is not at the same position as the thing being converged on
- Implications:
  - **Focusing**: if stereo display is not designed correctly, the eyes will try to converge on the screen itself
  - **Defocusing**: Depth-of-field effect won't be accurate because there is no "field" on a 2D display; the physical screen doesn't vary in depth so everything will be in focus
- In AR especially: since the real & virtual world are mixed, these distortions become very obvious



Types of AR Displays

Main Categories

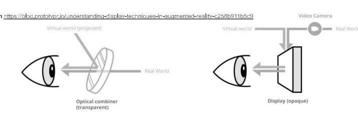
From https://blog.prototypr.io/understanding-display-techniques-in-augmented-reality-c258b911b5c9



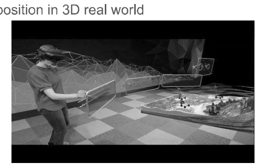
- Video Pass-Through/See-Through
  - You're looking at an opaque screen which shows you the view of a camera on the other side
  - E.g. mobile AR; you're looking at the phone screen which shows you the camera's view
  - E.g. Oculus Quest Guardian calibration; you see black & white camera image of the real world

Main Categories

From https://blog.prototypr.io/understanding-display-techniques-in-augmented-reality-c258b911b5c9



- Optical Pass-Through/See-Through
  - Virtual rendering done by:
    - Seeing an image that is reflected off of a **combiner** (usually mirror-like lens) (e.g. the image is translucent and appears on some display in front of the eye)
    - Seeing the image because it's reflected directly into the eye (**virtual retina displays**)
  - Describes most MR headsets that we care about
  - Image may or may not be 3D or related to the real world environment
    - E.g. Google Glass was not MR; only added some AR UI elements
    - E.g. Hololens is MR because all elements have 6dof position in 3D real world



Spherical/Semi-Spherical Large Combiner

- A screen/projector near forehead has image reflected by combiner
- The combiner/mirror is very large but supports high FOV & optics easier to get right
- Some of the cheapest AR HMDs you can find
  - E.g. this $60 AR headset on Amazon (has no tracking though…. But potentially good for UI): https://www.amazon.com/Headset-Glasses-Augmented-Reality-Android/dp/B07N2PXK8W/ref=sr_1_3?dchild=1&keywords=ar+headset&qid=1586803446&sr=8-3

Good followup resource: https://www.kguttag.com/2016/10/21/armr-optics-for-combining-light-for-a-see-through-display-part-1/

Spherical/Semi-Spherical Large Combiner

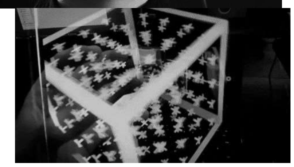
- Higher-tech 6DoF examples
  - Meta 2:
    - Had very high FOV (90 deg) but questionable tracking.
    - Built-in hand tracking.
    - Wired to PC.
    - Went out of business but is being revived
    - Headset is gigantic

FM: https://www.roadtovr.com/wp-content/uploads/2016/03/meta-2-development-kit-hands-on-augmented-reality-headset-AR-3.jpg

Good followup resource: https://www.kguttag.com/2016/10/21/armr-optics-for-combining-light-for-a-see-through-display-part-1/

Spherical/Semi-Spherical Large Combiner

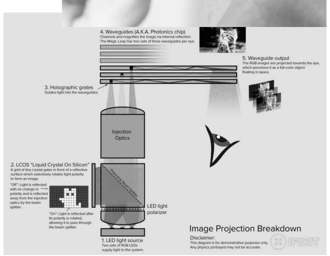
- Higher-tech examples
  - Leap Northstar:
    - Can be built for <$100
    - Has hand tracking with Leap Motion
    - Open source
    - Form factor not great
    - https://developer.leapmotion.com/northstar
    - No built-in tracking



Good followup resource: https://www.kguttag.com/2016/10/21/armr-optics-for-combining-light-for-a-see-through-display-part-1/

Near-Eye Displays

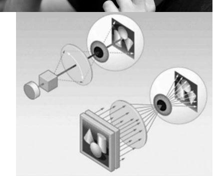
- Very similar to VR HMD design, except screens should be smaller
- Can be done in a few ways
  - Light fields
  - Microdisplays + waveguides + smaller combiner + some other stuff (complex Hololens "light engine")
    - Basically, shoot some light rays & use the waveguides/lenses to guide them to the combiner
  - Shoot light directly into eyes
- Basic goal is to make form factor as close to typical glasses as possible
- Very complicated & rely on deep understanding of how **optics** work (how eyes process light)



Good followup resource: https://www.kguttag.com/2016/10/21/armr-optics-for-combining-light-for-a-see-through-display-part-1/

Virtual Retina Displays (VRDs)

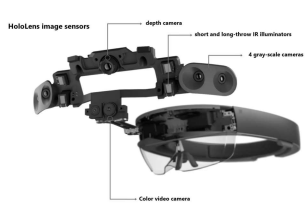
- Project image directly onto eye instead of mirror
- Very emerging; optics still hard to get right



Virtual retinal displays create an image directly on the retina without producing it first on a screen or in real space (left). This technique has several advantages over displays on tiny LCD screens (right) and has the potential to produce realistic 3-D full-color videos. The only drawback would be the need for helmets or goggles.

Near-Eye Display Examples

- Hololens by Microsoft (2016)
  - Current de facto standard for good AR headsets
  - Hololens 1: FOV of 30 degrees horizontally
    - 1 front-facing camera, 4 grayscale cams for tracking/reconstruction, 1 depth cam, some IR illuminators
    - Oculus Quest-style speakers
    - $3000 minimum!
    - Hardware similar to old smartphones (e.g. Galaxy S4)
  - Hololens 2: FOV of 43 deg horizontally
    - Same as H1 except better form factor/ weight distribution, hand-tracking, and much stronger hardware
    - Hardware similar to Galaxy S8
  - Both use a version of Windows RT so API familiar to many AR devs
  - Hardware all self-contained in HMD



Near-Eye Display Examples

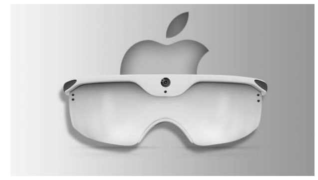
- MagicLeap One: FOV 40 deg horizontal (2018)
  - Hololens competitor funded by Google
  - Hardware slightly stronger than Hololens 1
  - Has eye-tracking which is a big deal
  - Has tracked remote & hardware stored on "lightpack" (little disk thing held in pocket)
  - Runs on Android-like OS called Lumin
  - Not much good content to expand from
  - Can't be worn with glasses; they wanted people to buy special lenses (as if MR isn't inaccessible enough….)

Near-Eye Display Examples

- Google Glass (13 degree FOV) (2013)
  - Fall under “smart glasses” which are really just meant for UI elements (no understanding of PE)
  - Tiny FOV makes it borderline useless
  - Flopped because of safety/privacy reasons (and the API was horrible and poorly supported)
  - They’re trying again…. Focusing on industrial applications like instructions & uses Android (2019)

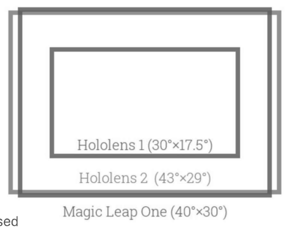

Near-Eye Display Examples

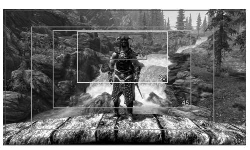
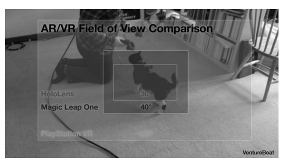
- Apple Glasses (unreleased)
  - Another version of smart glasses. Basically has same purpose as Apple Watch did
  - FOV seems pretty big though



Challenges of near-eye displays

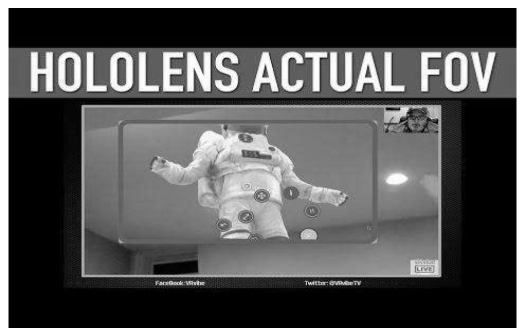
- Brightness vs opacity
- FOV is tiny (compare to Quest FOV which is 90 degrees)
- Hardware is weak
- Vergence-accommodation conflict more of a problem the closer combiner is to eyes
  - Focus is usually off…. We’ll see in a sec how it’s addressed



FOV problem

- Fundamentally an optics/display design problem



Addressing Focus: Varifocal Displays

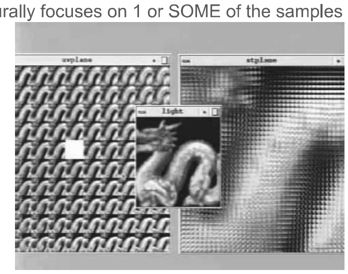
- Adjustable combiners/membranes that reflect different parts of image at different depths
  - So when the user looks at a given part of the image, membrane distorts to account for eye focus
- Usually requires moving hardware… susceptible to breaking or losing calibration


Addressing Focus: Multifocal Displays

- Multiple combiners/membranes that reflect different parts of image at different depths
  - So when the user looks at a given part of the image, they are naturally focusing on the right membrane
- Usually requires lot of calibrated hardware that’s hard to fit into ergonomic HMD
  - More hardware for more focal planes; e.g. 40-plane display: https://arxiv.org/abs/1805.10664
- MagicLeap One has multiple focal planes which Hololens doesn’t

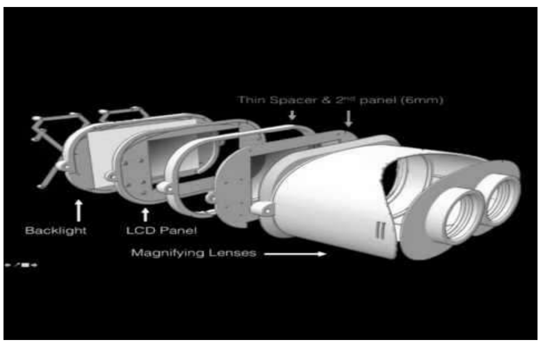
Another solution: Light Field Rendering (Levoy '96)

- Samples images from multiple viewpoints, interpolates between them, and disperses parts of the samples based on camera/eye params.
- Goal is to model how light actually bounces into the eye (similar to HRTFs in audio)
- Your eye naturally focuses on 1 or SOME of the samples and naturally defocuses others

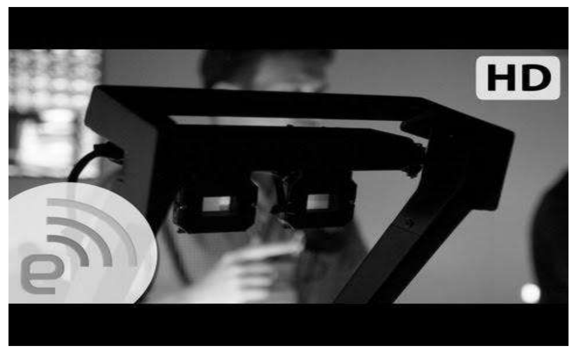


Great light field tutorial for independent use

Another good talk on light fields



Even more



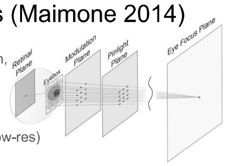
Lanman '13: Near-eye light field displays

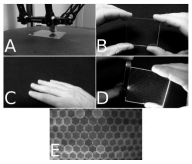
(not AR but work is being done to make it work better with MR displays)

Other AR/VR Displays/Holograms

Another AR Display Method: Pinlight Displays (Maimone 2014)

- Also highly recommend Maimone 2017 for overview of near-eye displays
- First combiner produces defocused image, 2nd combiner focuses it again, naturally don't focus on other parts of the image
  - (almost the exact reverse of how older light field rendering worked)
  - allows huge FOV (110 deg)
- Hard to do color accurately & hard to make sharp edges (makes it look low-res)





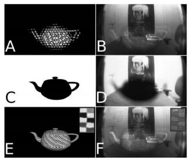
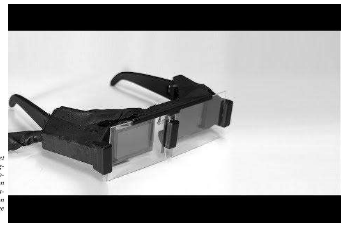


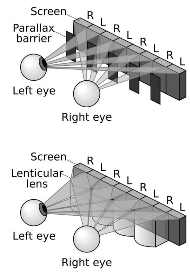
Figure 11: *Sample display inputs and results. Blue outlined inset images show magnified regions. A) Modulation pattern for augmented image sent to display for blue color channel. B) Photograph of display while blue color channel displayed. C) Modulation pattern sent to display as occlusion mask. D) Photograph of display while occlusion mask is shown. E) Simulated reconstruction of augmented image. F) Photograph of actual augmented image and background, taken over complete color and occlusion cycle.*



Autostereoscopy

- Ability to see "3D" object without requiring HMD

From YOCOMA



3DS

Autostereoscopic Displays/**Holography**

- Basically "holograms," one way or another (mostly illusions)
- A few ways of doing it
  - 3D glasses: not autostereoscopic since they require glasses but ideas are similar
  - Parallax barrier displays or Lenticular Lenses, aka "3D" TVs
    - Create "hologram" through visual trickery much like light field rendering
    - Older "3D" screens, older 3DS
  - With reflections
    - Light fields
    - Intersecting images/Wave Fields

Parallax Barrier Displays

- Cheapest & simplest type of autostereoscopic display
- Multiple images rendered and criss-crossed/multiplexed so barrier blocks part of image that viewer/eye shouldn't be able to see from that viewpoint
  - 3D glasses do this except w/ color channels (aka **anaglyph**); colors end up looking bad though
- Makes strong assumptions about IPD (interpupillary distance) and distance from screen
- Generally only looks ok from certain range of perspectives
- Resolution effectively cut in half
- 3DS and most of the 3D TVs popular in early 2000s

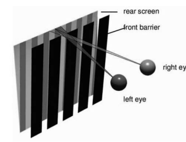
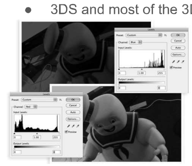
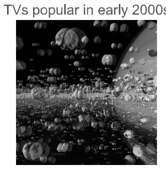
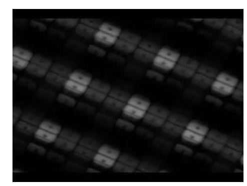


Fig. 1. A parallax barrier is a series of transparent and opaque strips that permit each eye to see different regions of the display screen behind the barrier.
From "Advances in the Dynallax Solid-State Dynamic Parallax Barrier Autostereoscopic Visualization Display System" (Peterka 2008), great resource on these displays

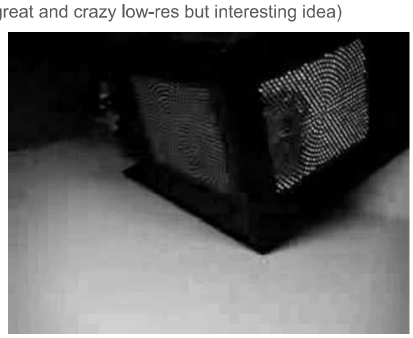






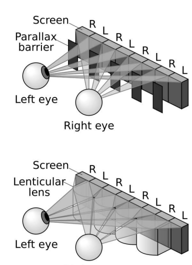
Cube version of parallax barrier display

(result not great and crazy low-res but interesting idea)



Lenticular Lens Displays

- Similar to parallax barrier except uses lenticular lens to cause eye to converge on right image
- Handles wider range of viewer positions
- Resolution still effectively cut, but maybe not in half
- Matusik 2004 ("3D TV: A Scalable System for Real-Time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes") good for understanding how the images get rendered
- Miyazaki 2014 uses slanted ones because math
  - "Multi-user autostereoscopic display based on direction-controlled illumination using a slanted cylindrical lens array"
  - Don't recall the effect looking significantly better than non-slanted



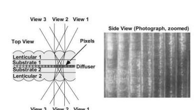
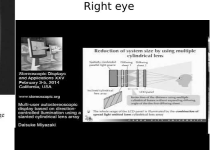


Figure 6: *Formation of vertical stripes on the diffuser of the rear-projection display. The closeup photograph (right) shows the lenticules and stripes from one viewpoint.*

From Matusik 2014

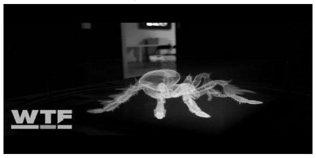
Figure 8 Pictures of images displayed by the experimental system taken at corresponding position of exit pupils; (a) Image for left eye and (b) for right eye.

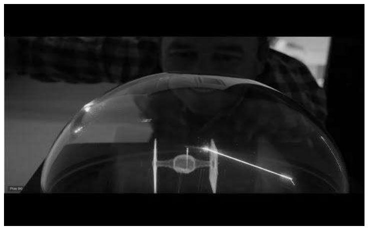
From Miyazaki 2014



Volumetric displays

- Basically the only existing "real" holograms
- Hardware bends light to cause beams to intersect/scatter & create 3D pixels
- Some other implementations have mirrors which rotate very fast to handle all viewpoints (more later)
- Limitations are mostly in resolution
  - Can only shoot so many light rays at once
- "Autostereoscopic"





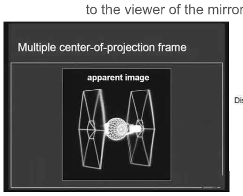
Autostereoscopic displays

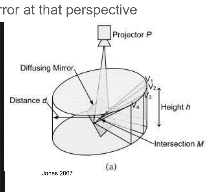
- Google Starline

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/424ee26722e5863f1ce17890d9499ba9a964d84f.pdf

Light Field & Volumetric Displays

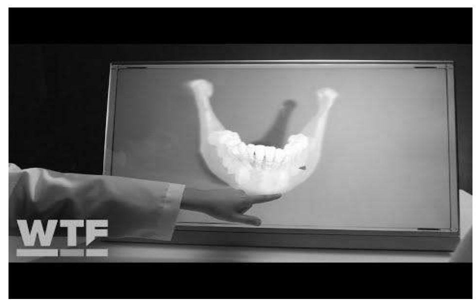
- Render bunch of viewpoints and interpolated perspective-correct image gets reflected back to viewer
- 3D effect depends on both motion parallax & stereo vision
- Expensive & requires a lot of hardware
- E.g. HoloVizio (Balogh 2006); basically a much better 3D TV
  - Wide range of 3D vision but still must be in front of screen
- E.g. Jones 2007 "Rendering for an Interactive 360° Light Field Display"
  - Spinning mirror which reflects appropriate part of light field to the viewer of the mirror at that perspective





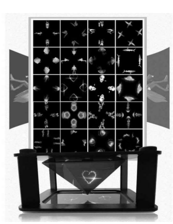
Cool modern light field display by Looking Glass (2020)

Basically much higher-res version of HoloVizio

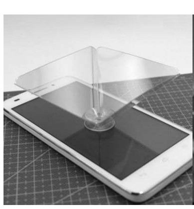


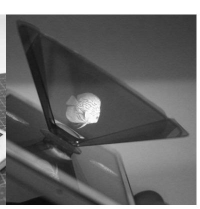
Intersecting Images/Wave Fields

- Probably oldest type of hologram (proposed in Lohmann 1978, expanded in papers like Berkhout 1988 & Dorsch 1994)
- Reflect multiple images onto a converging area such that intersecting pixels look 3D
- (can be reasonably built by anyone using screen + glass panels)
- Idea used in 3D audio to make good virtual audio with real speakers
  - (called **wave field synthesis**)



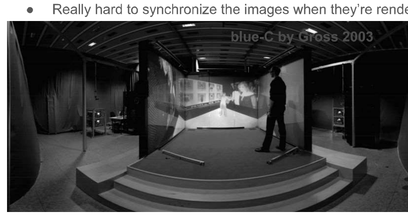
By Zerosky





CAVEs

- Older VR displays that used projectors & head-tracking to render images on some "screens"
  - (more like translucent walls)
  - Based on cave allegory from Plato's *Republic* (our reality see is a reflection/projection of the truth)
- Were arguably cheaper, more ergonomic, & easier to get working than old HMDs
  - Old HMDs were very low-latency & it was hard to generate the needed images on the same PC b/c GPUs were weak; CAVEs allow each side to be rendered on a different GPU very easily
  - Old HMDs also very heavy & wires get annoying
- Required ton of hardware & space; not easy to walk around without hitting wall
- Really hard to synchronize the images when they're rendered on different PCs



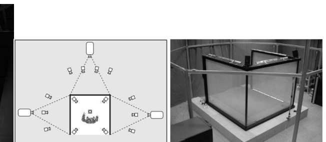


Figure 3: Picture illustrating the projector-camera arrangement of blue-c. a) Schematic view. b) Real system with scaffold to attach the cameras.

blue-C by Gross 2003

- Helped build foundation of CAVE systems, networking, etc.
- Image multiplexed & rendered to each eye with shutter glasses (basically horizontal parallax barrier displays)
  - Also protected eyes from extremely bright projector lights

Human Tails by Steptoe 2013 used a CAVE

- CAVEs were used for reconstruction/skeletal tracking a lot b/c it couldn't be done well in HMDs back then (this is still the case for reconstruction)

Raskar 1998 "Office of the Future" similar to CAVE

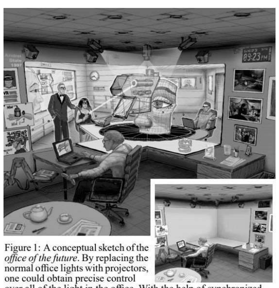


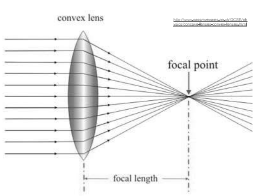
Figure 1: A conceptual sketch of the *office of the future*. By replacing the normal office lights with projectors, one could obtain precise control over all of the light in the office. With the help of synchronized cameras, the geometry and reflectance information can be captured for all of the visible surfaces in the office so that one can project images *on* the surfaces, render images *of* the surfaces, or interpret changes *in* the surfaces. The inset image is intended to help differentiate between the projected images and the real objects in the sketch.

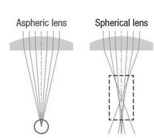
Display internals are very complicated

- Won't cover as it's not very important for the high-level developer
- Has tons of tiny hardware components like LCOS chips which reflect light
- Waveguides also pretty complicated...
- Resources for those interested:
  - https://www.kguttag.com/2016/10/27/armr-combiners-part-2-hololens/
  - https://en.wikipedia.org/wiki/Optical_head-mounted_display
  - https://medium.com/@RDelly/how-augmented-reality-headsets-work-diving-into-different-optical-designs-50d6240933ef
  - "Liquid-Crystal-on-Silicon for Augmented Reality Displays" by Yuge Huang
  - "Holographic Three-Dimensional Virtual Reality and Augmented Reality Display Based on 4K-Spatial Light Modulators" by Hongyue Gao
  - https://virtualrealitypop.com/understanding-waveguide-the-key-technology-for-augmented-reality-near-eye-display-part-i-2b16b61f4bae
  - https://www.youtube.com/watch?v=p8VfLwCxJ-4

Optical Distortion/Aberration

- Why aren't lenses flat?
  - Light physics! Lenses must be convex to guide light into eye
  - Lens shape affects convergence. **Aspheric** is common & combats distortions usually resulting in doubled image
- **Distortion**: normally caused by shape of lens not being flat
  - Resulting image doesn't align with what the "eye" saw
    - Lines not straight, scales messed up, etc.
  - Implications in most graphics fields, including computer vision & optics (AR/VR lens design)



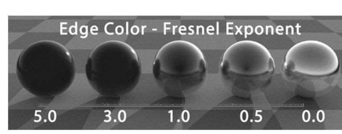


A spherrical lens with spherical aberration shows differences in focal positions of light rays depending on their distances from the optical axis.

undistorted pincushion distortion barrel distortion

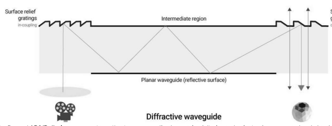
Fresnel

- Distortions to light being reflected/ refracted. Light at extreme angles more distorted (TLDR: lots of physics)
- Important consideration for AR/VR lenses, esp. Since eyes & screen/combiner (light sources) are so close to each other
- In VR optics, for screen image to transfer to eye correctly, we normally use **Fresnel lenses**
- AR is much harder....



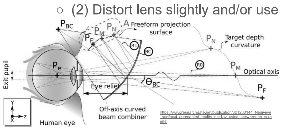
Distortion in AR

- AR is harder…. Need to somehow undistort virtual environment without distorting real view.
- Some ways to do it (very high-level):
  - (1) Design the waveguide & light sources to guide light onto a lens with little lens distortion (e.g. try to handle on the optics hardware side)
    - Problem here is hardware quickly starts becoming too hard to fit compactly

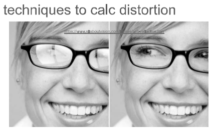


Really good AR/VR display resource: https://hackernoon.com/fundamentals-of-display-technologies-for-augmented-and-virtual-reality-c88e4d0b0895

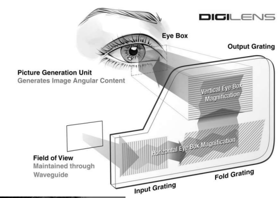
  - (2) Distort lens slightly and/or use reflective materials & computer vision techniques to calc distortion

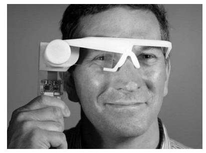


(3) **Mix both ideas!**



Common in Near-Eye Displays: Waveguides



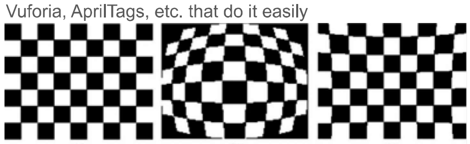


https://phys.org/news/2014-07-optical-component-revolutionise-augmented-reality.html

Good resource: https://medium.com/@RDelly/how-augmented-reality-headsets-work-diving-into-different-optical-designs-50d6240932ef

Calculating Distortion

- Typically done with easily-recognizable **fiducial markers**
  - Images/objects for which we know exact real-world shape/size/features without distortion
  - Similar in concept to diffuse/albedo/base texture map for materials in graphics
- Find marker in distorted image, figure out **equation/matrix** needed to undistort it, apply equation to all future images generated for that lens
  - Most cameras only require matrix since lenses usually uniform (convex shapes)
  - Many AR lenses complex enough to require equation(s) due to non-uniformity
- Details found in computer vision classes…
- Lots of APIs like OpenCV, Vuforia, AprilTags, etc. that do it easily

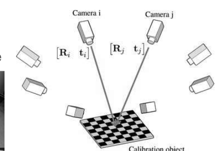


(a) (b) (c)

Figure 2. Examples of different types of lens distortion. (a) original (b) barrel distortion (c) pincushion distortion

Most Common Calibration Technique: Checkerboards

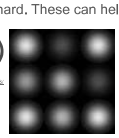
- Hold checkerboard with accurately-measured sizes in front of camera(s) outside of lens & calculate distortion/camera params
  - Aka **intrinsic calibration**
- Do this for plenty of checkerboard poses to best estimate distortion across lens
  - (esp. If shape of lens is not uniform)
- For many-camera setups, find & calibrate poses visible to both cameras to get **extrinsic calibration**
  - (where cameras are relative to each other)
- Extrinsic calibration for cameras that aren't the same is tedious….
  - Requires intrinsic calibration of each type of cam THEN extrinsic
  - When all cameras are the same, you can often do both parts at once



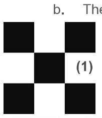
Good resource w/ code

Common Fiducial Markers

1. **Checkerboard**: Lines are trivial, high contrast (edge finding is easy), possibly hard to orient
2. **ArUco**: like simplified QR codes. Checkerboard benefits + orientation. Can change density
3. **ChArUco**: Checkerboard+ArUco. Checkerboard part great for straight lines, ArUco good for orientation
4. **Dense features/image targets**: Use feature correspondences. Good for cluttered images/reducing assumptions
5. **3D objects**: good for handling lots of marker orientations; harder to define unique features
6. **Defocused**: most cameras can't focus too close so small/close markers are hard. These can help address this
   a. (or use extremely close FOV cams like I do…. E.g. endoscopes)
   b. These can also help with auto-focus-related problems

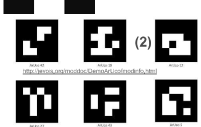
(6) 

(5) 

(1) 

Pretty good paper on making good markers: https://www.researchgate.net/publication/260251570_Automatic_generation_and_detection_of_highly_reliable_fiducial_markers_under_occlusion

https://ieeexplore.ieee.org/document/6080113/

(2) 

http://jevois.org/moddoc/DemoArUco/modinfo.html

(3) 

(4) 

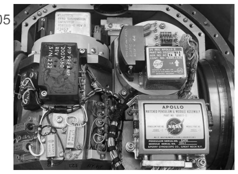
Common Marker-Tracking APIs for AR

- **OpenCV**: open-source, highly compatible, lots of example code, usually hardware-intensive
- **Vuforia**: proprietary :(, works great w/ weaker devices like Hololens, seemingly by far the best with occlusion, very hard to use for general purpose CV, doesn't natively work with UE4 :(
  - By far easiest to get working & best performance IMO. Doesn't require manual camera calibration which is a HUGE plus
- **AprilTags**: similar to OpenCV but meant to work with smaller markers & better performance. Common in robotics
- **MAXST**: similar to Vuforia that's harder to use & seems to perform worse but has SLAM (we'll see SLAM in a bit), doesn't natively work with UE4 :(
- **Wikitude**: marker tracking intended for mobile devices & location-based experiences (e.g. ads or tourism)
- **ARToolkit**: Basically OpenCV except specifically made to work on AR devices (better real-time performance)
- For video passthrough: **ARKit** (iOS) **& ARCore** (Android) (mobile only). Great performance on their intended OSes…. Not really for anything else. Pretty easy to use. Seem to be replacing ARToolkit
- Many of these rely on marker taking decent % of the image space (e.g. marker should be at least 10% of image)
  - Means that camera FOV, focal range, etc. are important. Very small/large markers are hard

Somewhat dated comparison here: https://thinkmobiles.com/blog/best-ar-sdk-review/

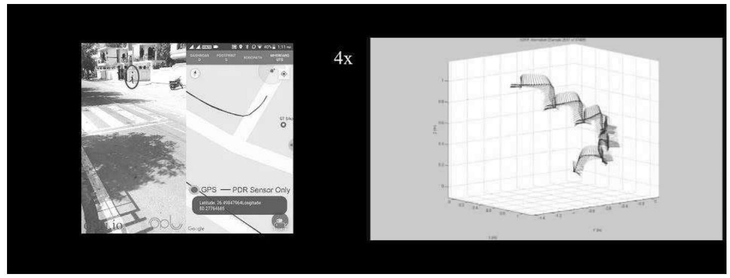
Tracking AR devices

- Like with VR, 6dof AR devices need to know position
- AR devices are assumed to have NO external tracking ability whatsoever…
  - (e.g. the Quest is arguably the first consumer VR device to also make this assumption in 6dof)
- 3dof rotation can be handled mostly by internals (recall Oculus Tracking paper)
  - As we'll see, we still need a lot of correction in AR b/c AR devices have weak hardware & can't correct rotation often enough to avoid drift
- 3dof translation & rotational correction is harder…. Very hard to do internally
  - Some cool implementations of internal translation tracking with IMUs, which sense a few types of deltas. Usually not accurate enough to work for AR/VR but still cool
    - e.g. "Pedestrian Tracking with Shoe-Mounted Inertial Sensors" Foxlin '05
    - Measuring only deltas means error can accumulate
  - Typically use cameras & reconstruction



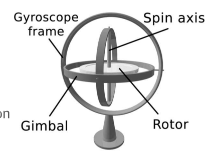
Apollo 11 IMU

Examples of Translational Tracking w/ IMUs



LaValle '14: "Head Tracking for the Oculus Rift"

- Great read if you're interested in tracking
- Key points:
  - VR tracking is not done with 1 tracking method
  - Gyroscope gives rough 3dof rotation
  - Tilt correction is done with accelerometer which detects gravity direction
  - Drift correction is doing with filters
  - Yaw correction is done with magnetometer, sort of like a compass
  - Predictions are done for further corrections



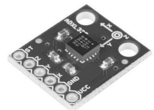
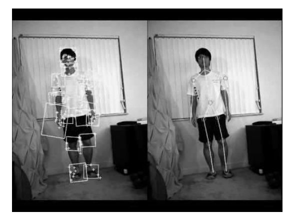
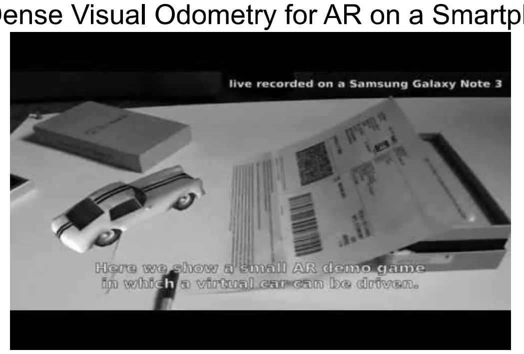


Image-Based Tracking

- Tracking by figuring out where feature moved between frames of video
- **Feature tracking:** methods to find where 1 feature is inside of another image/frame
  - another topic for CV classes
- **Optical flow**: set of techniques finding displacement of features between frames
  - Usually make continuity assumptions to assist with **confidence** (how sure you are that it's the right feature
  - Usually done with KLT (Kanade–Lucas–Tomasi) method
- Can be done with or without markers; markers have much higher confidence
- Great for most cases & easy to implement



Recall: Markerless/Reconstruction Examples: "Semi-Dense Visual Odometry for AR on a Smartphone"



Why Isn't Image-Based Tracking Enough?

- Iterative feature-matching methods with only 2D images converge slowly
- Feature detection often requires high resolution/strong CPU
  - Not an option for most modern AR/MR devices…especially as we move to mobile. Lots of latency
- Motion blur & other distortion (e.g. auto-exposure, auto-focus) make it hard to have high confidence in feature correspondences
- Depth is ambiguous! Small object close to camera or giant object from camera?
  - Where pure AR & MR diverge: using real world to guide mechanics in virtual world…. Need to know where real world objects are in 6dof

Intro to 3D Reconstruction

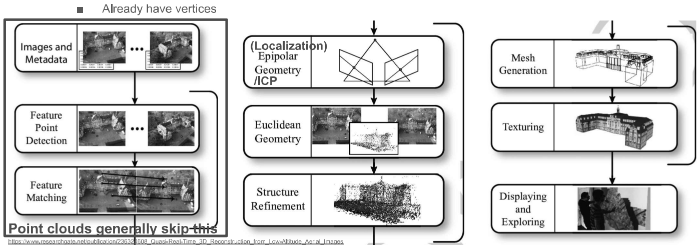
(low-level details found in classes on computational geometry, numerical analysis, etc.)

Reconstruction & AR

- Depth sensors and/or Structure from Motion (SfM)
- 2 fundamental methods in reconstruction
  - **SLAM**: Simultaneous localization and mapping
    - Used to figure out structure of real world
  - **ICP**: Iterative Closest Point
    - Used to figure out which part of partial 3D points

Reconstruction Overview

- Trying to create a full 3D mesh from non-meshed data
  - Recall that a mesh needs vertices, edges, AND polygons for rendering
- 2 typical types of non-meshed data
  - 2D images (taken from regular camera)
    - Requires **photogrammetry** to get vertices
  - Point clouds (taken with depth sensor)
    - Already have vertices

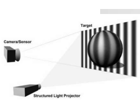


https://www.researchgate.net/publication/236324608_Quasi-Real-Time_3D_Reconstruction_from_Low-Altitude_Aerial_Images

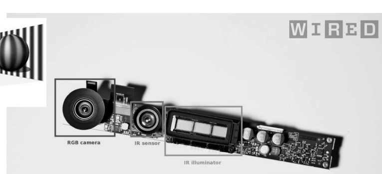
Point Clouds

- **In general**: bunch of 3D vertices, possibly with vertex colors
  - (a regular 3D mesh can be trivially converted to a point cloud by removing edges & polygons)
- **In reconstruction**: 3D feature correspondences from depth sensor
  - (both cameras in stereo sensor realize they see the same point)
- Standard file format: **.ply**
- Almost always incredibly noisy if gathered from sensors
  - (feature correspondence often random; pyramidal KLT can sometimes track them)
- If they have vertex colors, point cloud density usually affects quality of resulting texture
  - Density affected by things like interference, reflections, depth sensor cam quality, latency, etc.
- **The challenge**: how do we create the mesh?
  - Mesh is important for geometric algorithms, rendering, collision detection, etc.

Depth Sensors



- Figure out depth of pixel
- Create "**RGBD**" image
  - (often D is put into alpha so RGBA)
- Done a few ways:
  - **Structured light**
    - shoot pattern of light, see how object deforms it
    - Highly susceptible to interference
  - **Calibrated stereo cameras**
    - Synchronized & calibrated cams which see
  - **Lidar sensors** for incredibly high-end apps like space
    - Send patterns out at long distances & sense it
    - Basically high-end structured light sensors



Good info: https://www.intelrealsense.com/beginners-guide-to-depth/

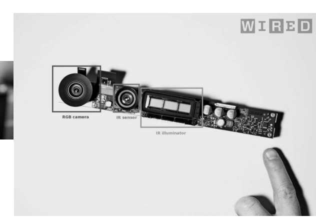
Popular Depth Sensors

- Kinect

- Realsense

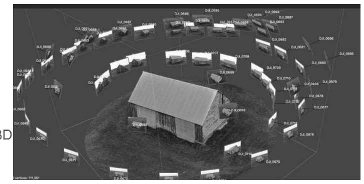
- Leap Motion

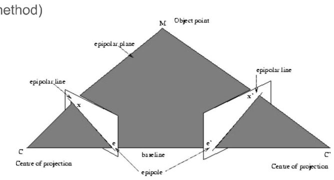
- iPhone X
- Structure



Photogrammetry

- Using CV techniques on tons of images of the same object to reconstruct
  - We can find consistent features across a few camera views & estimate camera pose, then convert features along object of interest into 3D point cloud using epipolar lines & use reconstruction methods for mesh itself
- Deep learning methods help figure out object of interest (e.g. YOLO method)





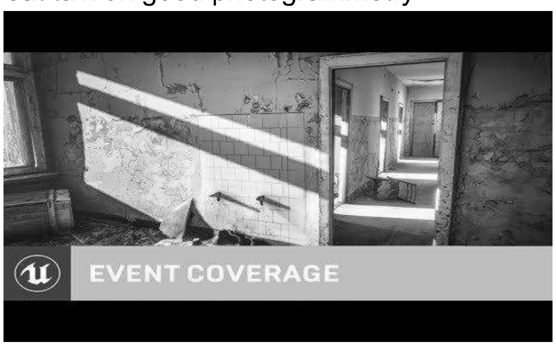
Resources for those interested:
http://paulbourke.net/reconstruction/

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT10/node3.html

Cool example

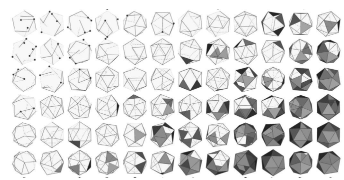
- Photogrammetry is normally how really complex 3D objects based on real things are made
  - E.g. things in nature often close to impossible to model well by hand

Great talk on good photogrammetry



Reconstruction Methods

- Robust methods necessary for things like dentistry, brain scans, etc.
- Common methods
  - **Computational geometry** (Voronoi methods & Delaunay triangulation)
  - **Fitting implicit functions** (3D version of spline-fitting + Labelling/Clustering)
- Robust methods traditionally not meant to be realtime
  - AR reconstruction needs to be realtime & work on weaker hardware
  - Often use these methods at lower resolution, try stochastically, local predictions, etc.

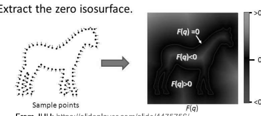


Implicit Function Fitting

Given point samples:

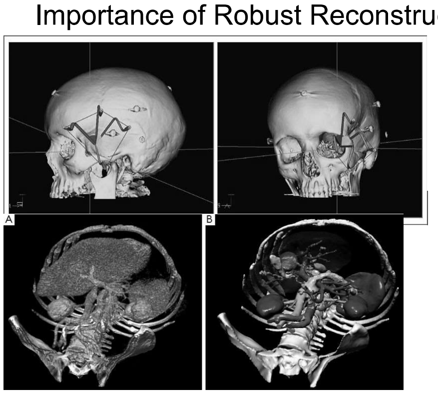
– Define a function with value zero at the points.

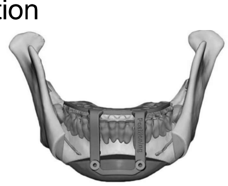
– Extract the zero isosurface.

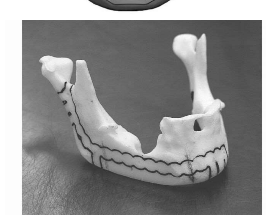


From JHU: https://slideplayer.com/slide/4475756/

Importance of Robust Reconstruction



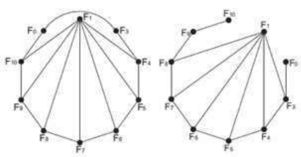




Computational Geometry Methods

Vertex Switching (Stanley 1985)

- Graph theory-based
- Basically tries to create closed graph of vertices without intersections
- Still one of the best in terms of reconstruction quality in dense clouds
- Works really well on sparse clouds
  - (others usually don't; modern research working on this)
- Incredibly slow & memory intensive. Basically the bubble sort of reconstruction



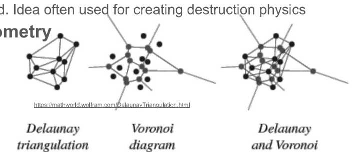
(better resource on the method than the original paper itself: https://www.site.uottawa.ca/~lucia/grads/patrick/niesink_msc.pdf)
Proofs: https://www.sciencedirect.com/science/article/pii/S0095895684710057?via%3Dihub

Delaunay Triangulation (Delaunay 1934)

- Set of algorithms which create mesh with union of triangles
  - Creates triangle with 3 verts if the circumcircle doesn't encompass another triangle
  - Tries to prevent triangles from intersecting
  - Usually many possible solutions
  - Typically creates **convex hull**: mesh encompassing points meeting convexity rules (used to generate your MeshColliders in Unity. Known as "simple collision" in UE4)
- **Voronoi diagrams:** construct mesh by drawing edges perpendicular to line generated by vertex pairs which intersect at center of points (simplest method)
  - These methods are **duals**: can be converted between each other fairly trivially (https://en.wikipedia.org/wiki/Dual_graph)
  - Each possible configuration is a different seed. Idea often used for creating destruction physics
- Foundation of most **computational geometry**

Resources for those interested:
https://tanmaybinaykiya.github.io/tetrahedralization-of-3d-meshes
https://en.wikipedia.org/wiki/Voronoi_diagram
https://kogs-www.informatik.uni-hamburg.de/~tchernia/slides/surface_reconstruction_WS0809.pdf
https://www.semanticscholar.org/paper/Optimization-of-Mesh-Reconstruction-using-Delaunay-Sood-Kaur/37f6a549177d2d38ae96aa7ddbcd306350725672
https://demonstrations.wolfram.com/ConvexHullAndDelaunayTriangulation/

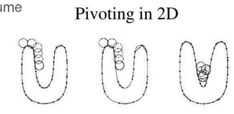
https://mathworld.wolfram.com/DelaunayTriangulation.html



Ball Pivoting reconstruction (Bernardini 1999)

- Basically rolls a ball of uniform size between vertices; edge created if ball can reach another vertex
  - (note that the goal is to create the outer hull; we don't care much about the inner verts which will be invisible to the mesh viewers)
- Fast & common method. Works best on uniform-density clouds
- Good for general-purpose reconstruction and doesn't make assumptions the the mesh is closed
  - If mesh is open, ball will roll back around & create flat volume

Pivoting in 2D



(a) Circle of radius ρ pivots from point to point, connecting them with edges.
(b) When sampling density is low, some of the edges will not be created, leaving holes.
(c) When the curvature of the manifold is larger than 1/ρ, some of the points will not be reached by the pivoting ball, and features will be missed.

Resources for those interested:
https://doc.cgal.org/latest/Advancing_front_surface_reconstruction/index.html
https://www.cs.jhu.edu/~misha/Fall13b/Papers/Cazals06.pdf

https://www.slideserve.com/malaya/the-ball-pivoting-algorithm

Power Crust (Amenta 2001)

- Watertight mesh
  - (rarely needed for AR since there are a lot of "holes", e.g. mirrors)
- Voronoi diagram creation->make spheres as big as can fit without touching points along edges of voronoi diagram
  - (use of spheres comes in handy like in ball pivot)
- Idea is that there will be clear separation of inner-mesh spheres and outer spheres (which become gigantic as they're unconstrained on multiple sides)
- Simpler mesh generated from centers of inner spheres called **medial axis**
  - can be used for skeleton/shape recognition
- Can be incredibly slow but robust. Techniques in vein of stochastic gradient descent (choose random # of points) help a bit

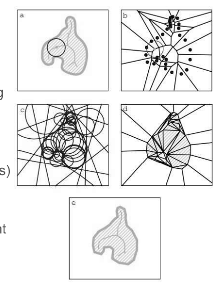


Figure 2. Two-dimensional example of power crust construction. a) An object with its medial axis; one maximal interior ball is shown. b) The Voronoi diagram of S, with the Voronoi ball surrounding one pole shown. In 2D, we can select all Voronoi vertices as poles, but not in 3D. c) The inner and outer polar balls. Outer polar balls with centers at infinity degenerate to halfspaces on the convex hull. d) The power diagram cells of the poles, labeled inner and outer. e) The power crust and the power shape of its interior solid.

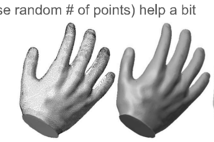


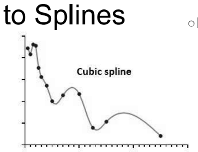
Figure 1. Laser range data, the reconstructed watertight polygonal model, and its simplified medial axis.

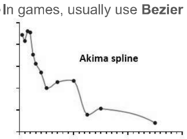
Resources for those interested:
http://www.cs.jhu.edu/~misha/Fall05/09.26.05.pdf

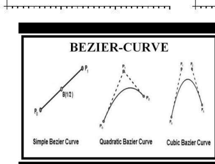
Implicit Functions

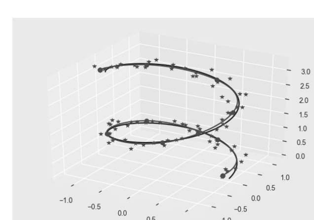
Recall: Quick Intro to Splines

- **Spline**: interpolant with certain amount of differentiability
  - **Interpolant**: Gives us info between data points, eg. between waypoints
  - Not important for our purposes, but why is differentiability important?
    - Allows us to easily merge interpolants without creating holes or sharp edges
  - In games, usually use **Bezier splines**, which use control points to describe tangents



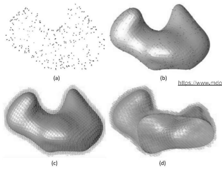






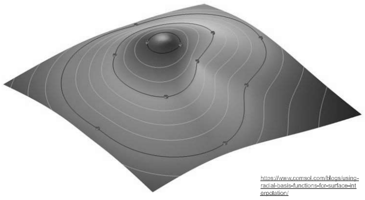
Radial Basis Function (RBF) & Moving Least Squares (MLS)

- Surface interpolants… very similar to splines except for surfaces
  - Find (usually continuous) function that wraps around points
- Activation neurons measuring how close a point is to the implicit function.
- **Implicit methods** iteratively try to learn the implicit function which the points are closest to up to a certain order
  - **Order**: how complicated the function is
  - Not very fast yet (but getting there) (e.g. more recent, robust deep learning methods)



https://www.mdpi.com/2076-3417/9/24/5335

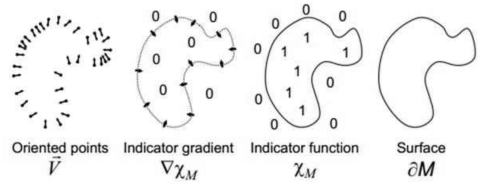
Resources for those interested:
https://kogs-www.informatik.uni-hamburg.de/~tchernia/slides/surface_reconstruction_WS0809.pdf



https://www.comsol.com/blogs/using-radial-basis-functions-for-surface-interpolation/

Poisson Surface Reconstruction (Kazhdan 2006)

- Tries to create implicit surface based solely on normals
  - **Normal**: which way point is facing/oriented, generated by point cloud during the capture
  - In graphics, used to determine which way the "outside" of the surface is
  - Can sometimes preprocess normals
- Fails if normals aren't well-oriented
- Fairly popular for closed surfaces
- Works well with depth sensors b/c normals usually point towards the camera (outwards relative to the surface



Resources for those interested:
https://stackoverflow.com/questions/838761/robust-algorithm-for-surface-reconstruction-from-3d-point-cloud

https://www.researchgate.net/publication/269324165_3D_Surface_Reconstruction_Based_on_Image_Stitching_from_Gastric_Endoscopic_Video_Sequence

Texturing

- Use vertex colors directly (match the point to the color in image space)--looks bad unless high-res point cloud
- Find camera with best view of a triangle, and project the part of the image belonging to that triangle onto it, building up a texture/UV map (Catmull 1974, Furukawa 2015, other multi-view stereo papers)

https://www.andreasjakl.com/capturing-3d-point-cloud-intel-realsense-converting-mesh-meshlab/

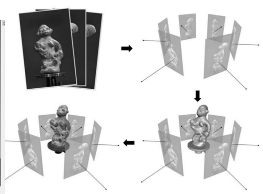


Figure 1.2: Example of a multi-view stereo pipeline. Clockwise: input imagery, posed imagery, reconstructed 3D geometry, textured 3D geometry.
https://carlos-hernandez.org/papers/fnt_mvs_2015.pdf

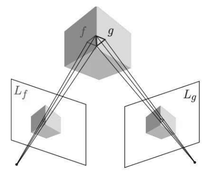


Figure 1. Textures are generated by back-projecting views to corresponding triangle faces.

https://www.semanticscholar.org/paper/High-Quality-Texture-Mapping-for-Multi-view-Dou-Nagahashi/2ce9bd171dd182ac9e3c9ddf44e271da4dea9a04

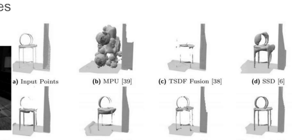
Resource on map for projecting textures to 3D: https://syncedreview.com/2020/04/25/turing-award-a-closer-look-into-edwin-catmulls-phd-thesis/, https://ia801307.us.archive.org/2/items/DTIC_ADA004968/DTIC_ADA004968.pdf (Catmull 1974 talks about texture mapping for reconstruction to give appearance of complex surfaces; Ed Catmull worked for Pixar/Disney for many years & is a big part of what their movies look so good)

Hard Reconstruction Cases

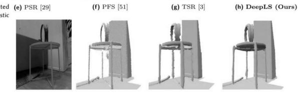
- **Thin surfaces**: cause sparsity in point cloud, require high-res RGB or RGBD cam
  - Only recent papers can reconstruct well, e.g. DeepLS (Chabra 2020)
- **Reflections**: cause IR rays to reflect; basically impossible to reconstruct
  - Some recent papers take advantage of reflections to reconstruct rest of environment (e.g. Whelan '18)
- **Light interference**: can obscure texture in image-based tracking
  - Visible spectrum lights don't really affect IR (why depth sensors usually use IR)
- **IR interference**: can obscure IR pattern or give false positives
  - Huge problem if using multiple depth sensors at once

Fig. 1. Reconstructing a scene with mirrors. *From left to right:* Input color image showing the scanner with attached AprilTag in a mirror, reconstructed geometry without taking the mirrors into account, reconstruction taking the detected mirrors (rendered as cross-hatched area) into account and a photorealistic rendering of the scene including the mirrors. Detecting the mirrors is crucial for accurate geometry reconstruction and realistic rendering.

Resources for those interested:
https://research.fb.com/publications/reconstructing-scenes-with-mirror-and-glass-surfaces/



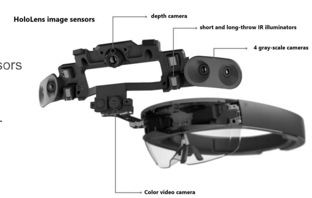
(a) Input Points (b) MPU [39] (c) TSDF Fusion [38] (d) SSD [6]
(e) PSR [29] (f) PFS [51] (g) TSR [3] (h) DeepLS (Ours)

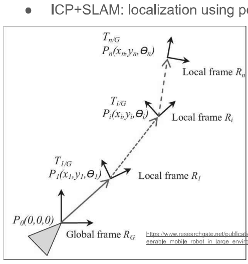


(i) Close-up view. From left to right: RGB, Input Points, TSR and DeepLS reconstructions.

Pose Over Time

- AR devices like Hololens are basically moving depth sensors
- How to know where 6DoF inside-out tracked devices are between frames?
- How to know that the real world 3D object in 1 frame after reconstruction is same as in another frame?
- ICP+SLAM: localization using point clouds

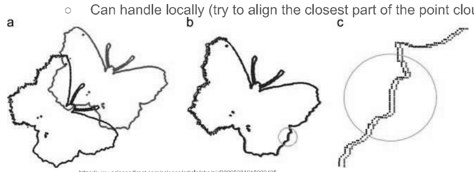




https://www.researchgate.net/publication/308865849_FAST_ICP-SLAM_for_a_bi-steerable_mobile_robot_in_large_environments

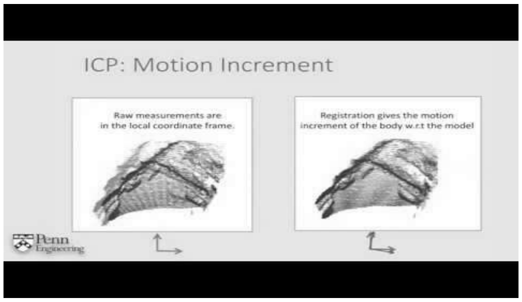
Iterative Closest Point (ICP) (Yang 1991, Besl 1992)

- Assumes at least some points in point cloud b/w frames are pretty close
  - (usually a fair assumption in dense point clouds)
- Iteratively makes guesses to try to align as many points as possible EM/gradient descent style
  - Basically tries to minimize MSE between 2 point clouds; some methods use curvature as well
- Important for any dynamic reconstruction actor (depth sensor itself, the subject, etc.)
- Foundational concept in dynamic 3D tracking systems
- Converges pretty quickly but # of points becomes a problem
  - Can handle stochastic gradient descent style
    - (not a good idea for point clouds b/c we don't know which features correspond b/w frames
  - Can handle locally (try to align the closest part of the point cloud, e.g. your desk)



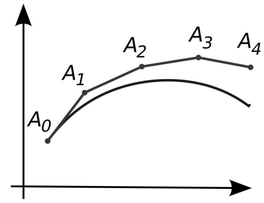
https://www.sciencedirect.com/science/article/abs/pii/S0925231215000405

Good video on ICP



Recall: Euler Method for Differential Equations

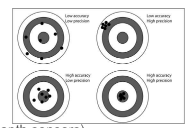
- Current position+(derivative at point (speed)*time)
- Predicted function strays from true function but preserves approximate shape
- Similar to how IMU error works without image/depth tracking
- Aka "dead reckoning"



Simultaneous Localization and Mapping (SLAM)

- General set of methods trying to figure out where a device moved between frames (localization) & saves reference point it used to figure that out for future reference (mapping)
- Works great by itself, but ICP necessary to figure out if, after multiple instances of the simulation (e.g. Hololens turned on & off), the new instance's point cloud is the same as before (e.g. you're in the same room)
  - Also necessary to recognize if a 3D object is the same as before/where you placed virtual objects
- Usually uses dead reckoning for initial predictions, then RGB/RGBD for corrections
- ICP (RGBD) for corrections & building global map
- Used by pretty much any standard 2D/3D tracked device

Kalman Filters for Better XR SLAM



- Kalman filters
  - Assume mix of sensors:
    - high-freq, inaccurate sensor using **dead reckoning** (e.g. IMU)
    - low-freq, accurate sensor using **global estimate** (e.g. GPS, images, depth sensors)
  - Computes covariance, error, corrections, etc. based on drift vs. global estimate
  - IMUs don't have a global estimate without inside-out or external tracking thus cannot be filtered alone
  - Allows us to have high tracking frequency with good accuracy (sub-millimeter)
- Limitations of Kalman filter:
  - Many sensors (esp. cheap ones) can't estimate their own error well or are finicky or low quality (e.g. low-res cam is not a great global estimator)
  - Assumes a Gaussian covariance which isn't always true (particle filters better for other types of distributions, esp. When there's a lot of interference, e.g. IR)
  - Require a decent # of timesteps to make accurate corrections (some delay)
- Oculus Quest/Rift S: global estimate is a few 3D features in PE (https://ai.facebook.com/blog/powered-by-ai-oculus-insight/)
- Hololens/MagicLeap: global estimate is low-poly reconstructed room mesh & anchor

Analytical and Iterative Methods

- Notice a trend in CS fields…
- Reconstruction:
  - Physically-based methods & computational geometry: Voronoi decomposition/Delaunay
  - Implicit function prediction: iteratively find parameters of nth-order function describing surface
- Tracking:
  - Analytical: dead reckoning & Euler methods
  - Iterative: ICP+SLAM, Kalman filters
- **Analytical/geometric/physically-based/closed-form** methods are great for handling knowns/constraints
- **Iterative** methods are great for handling noise + free/bound variables
- In 3D graphics, we almost always use a combination of them
  - Lots of sources of noise & unknowns, but constraints are easier to handle b/c we have dimensionality limitations, constraint dependencies, closed surfaces, etc.

Reconstruction still has a place in VR (besides tracking)

- Reconstruction of avatar, face tracking, real environment tracking, etc.
- Sra 2016 "Procedurally Generated Virtual Reality from 3D Reconstructed Physical Space" uses real world to make VE
  - Definitely MR

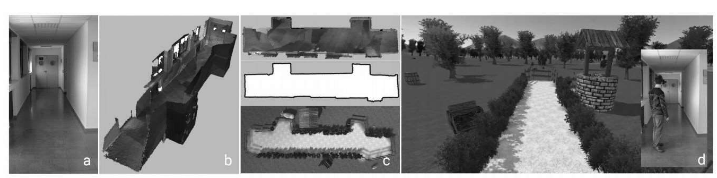


Figure 1: *Different steps of the proposed system. (a-b) We start with creating a 3D map of the real environment. (c) We detect the walkable area in the input 3D map to determine where the user can move freely. The generated virtual world is created according to the estimated walkable area in the point cloud. (d) Inset shows a user navigating the generated virtual environment by walking in the real environment, while visually experiencing it through a Tango HMD.*

Spatial Understanding/Awareness/Anchors

- **Spatial Understanding/Awareness/Mapping**: library for alignment of physical/virtual world
  - (basically a wrapper for SLAM/ICP)
  - Nice features like producing meshes of the REAL room in realtime for collision detection!
  - Has other built-in functions for raycasting onto real objects/meshes
- **Spatial Anchors/World Anchors/Persistent Coordinate Frames**
  - Define the 3D features used to most quickly do 6DoF tracking
    - E.g. this is done automatically by default to try to stabilize “holograms”
  - Also allow multiple of the same AR device to share these and see same virtual object in same physical place for multi-user apps
  - Sometimes store key frames or key 360 images to recognize quickly (e.g. Oculus Quest)