ICCS-404

Computer Graphics and ~~Augmented~~ Reality

EXTENDED

Pisut Wisessing, PhD

Assistant Professor

CMKL University

Revised Syllabus

ORIGINAL

- Number of quizzes and assignments
- 3 Projects

REVISED

- Flexible numbers, maintain %
- 2 Projects – the AR project is 35%

Augmented Reality (AR)
ความเป็นจริงเสริม

Augmented Reality Definition

- Combines Real and Virtual Images

  Both can be seen at the same time

- Interactive in real-time

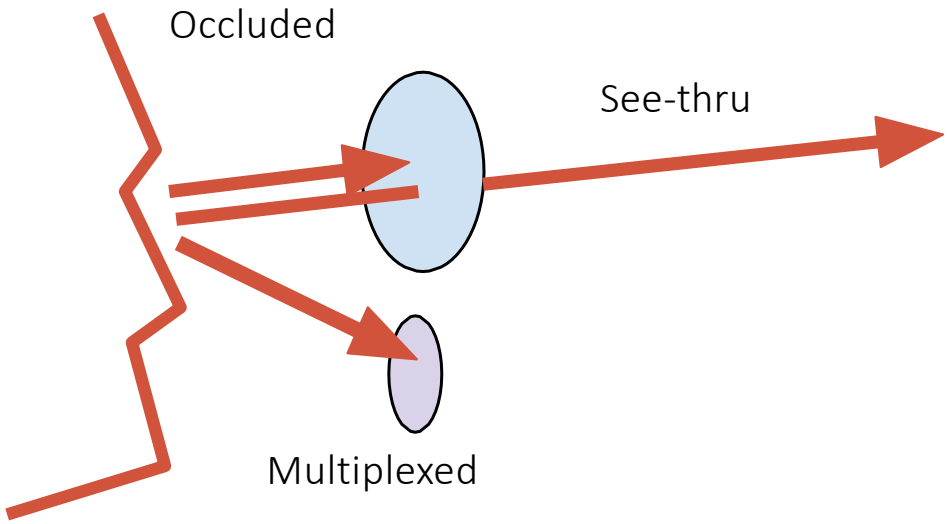
  The virtual content can be interacted with

- Registered in 3D

  Virtual objects appear fixed in space

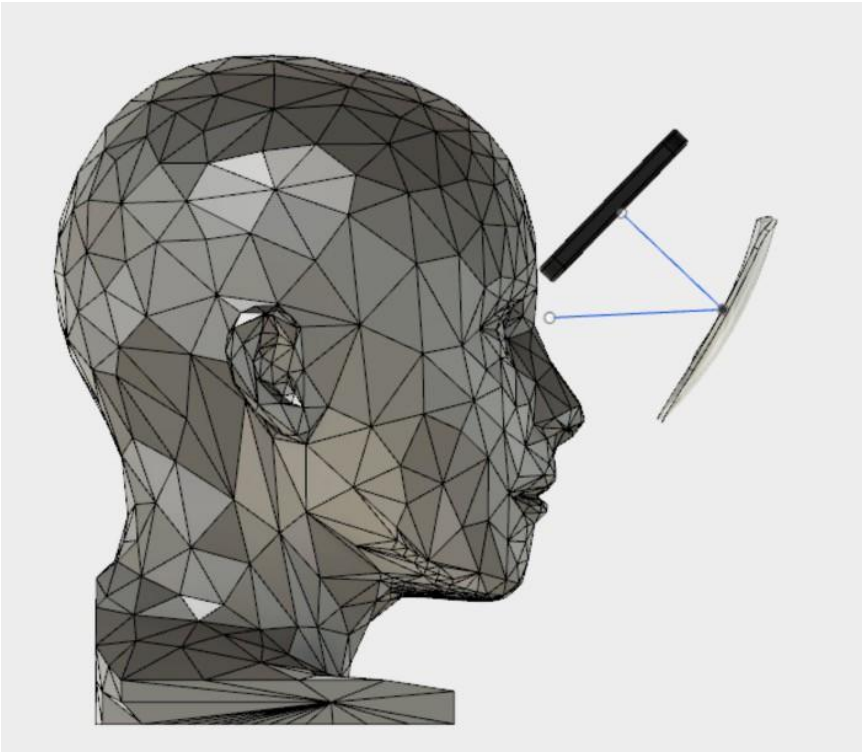
AR Display - HMDs

Head Mounted Displays (HMDs)



View Thru Optical See-Thru HMDs

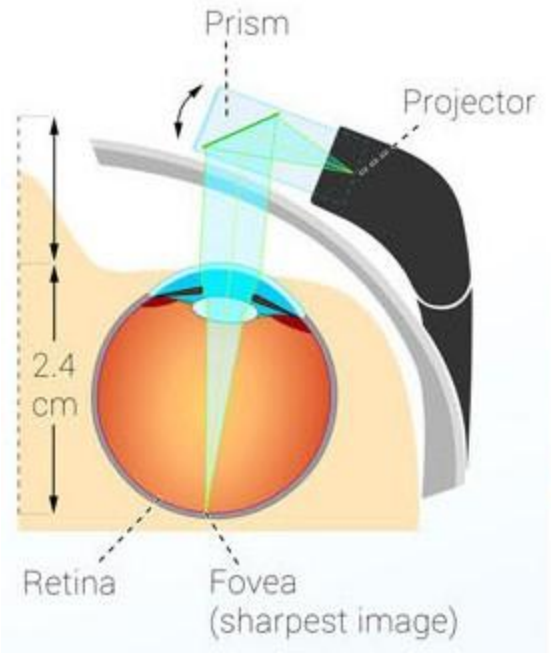
Optical Design – Curved Mirror

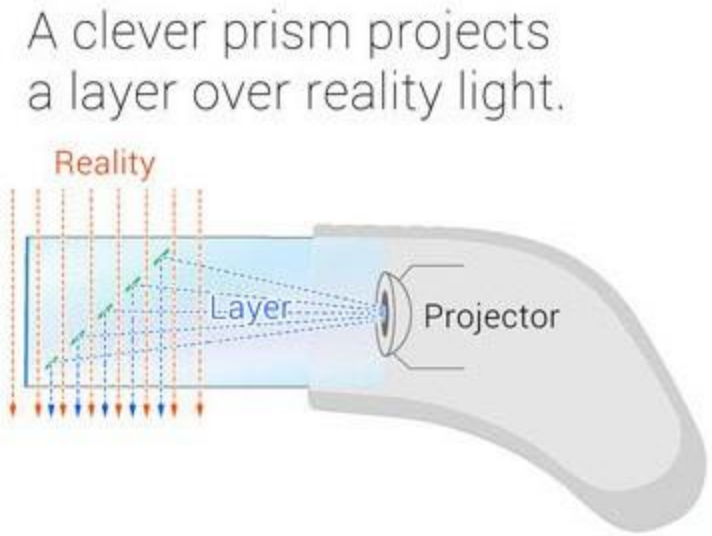


Example: Meta 2 (Not the FB Meta)

Link to Video

Optical Design - Prism



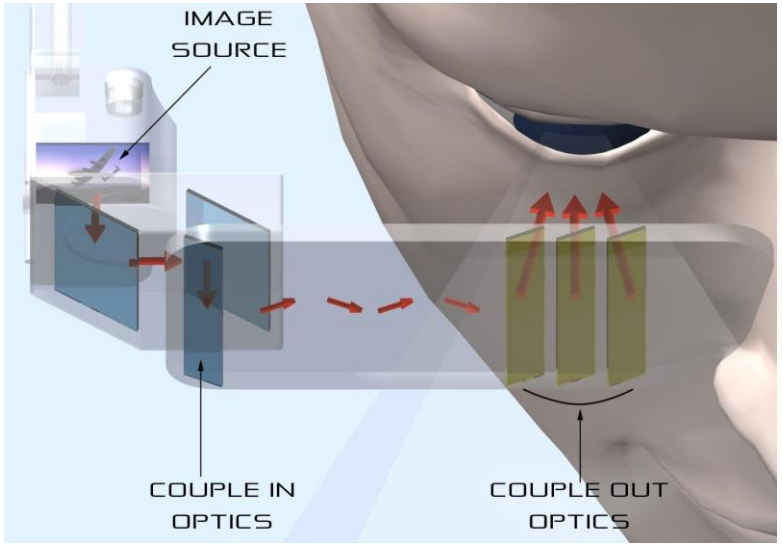


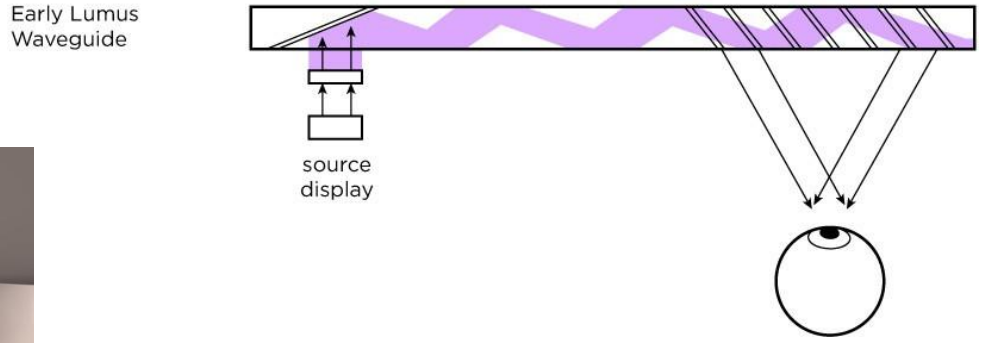
Example: Epson Moverio

Link to video

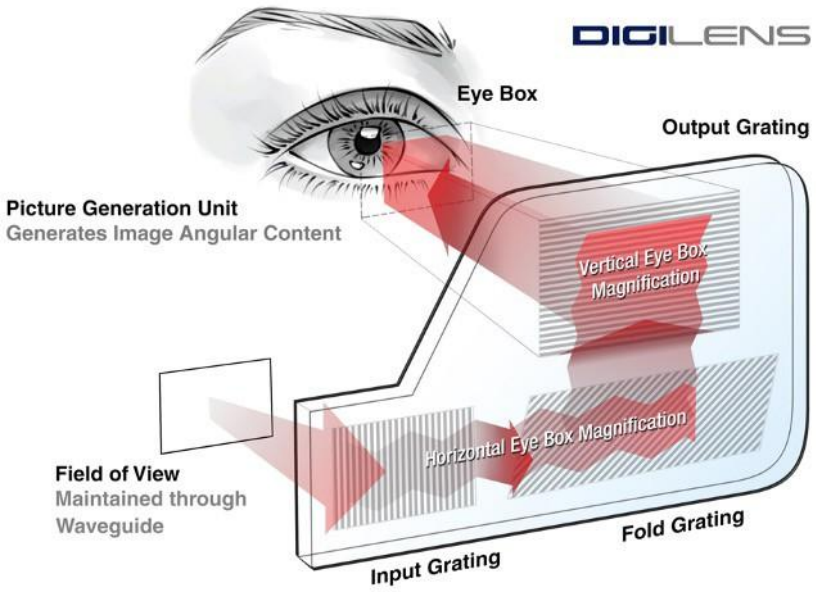
Optical Design – Waveguide

Prism + grating

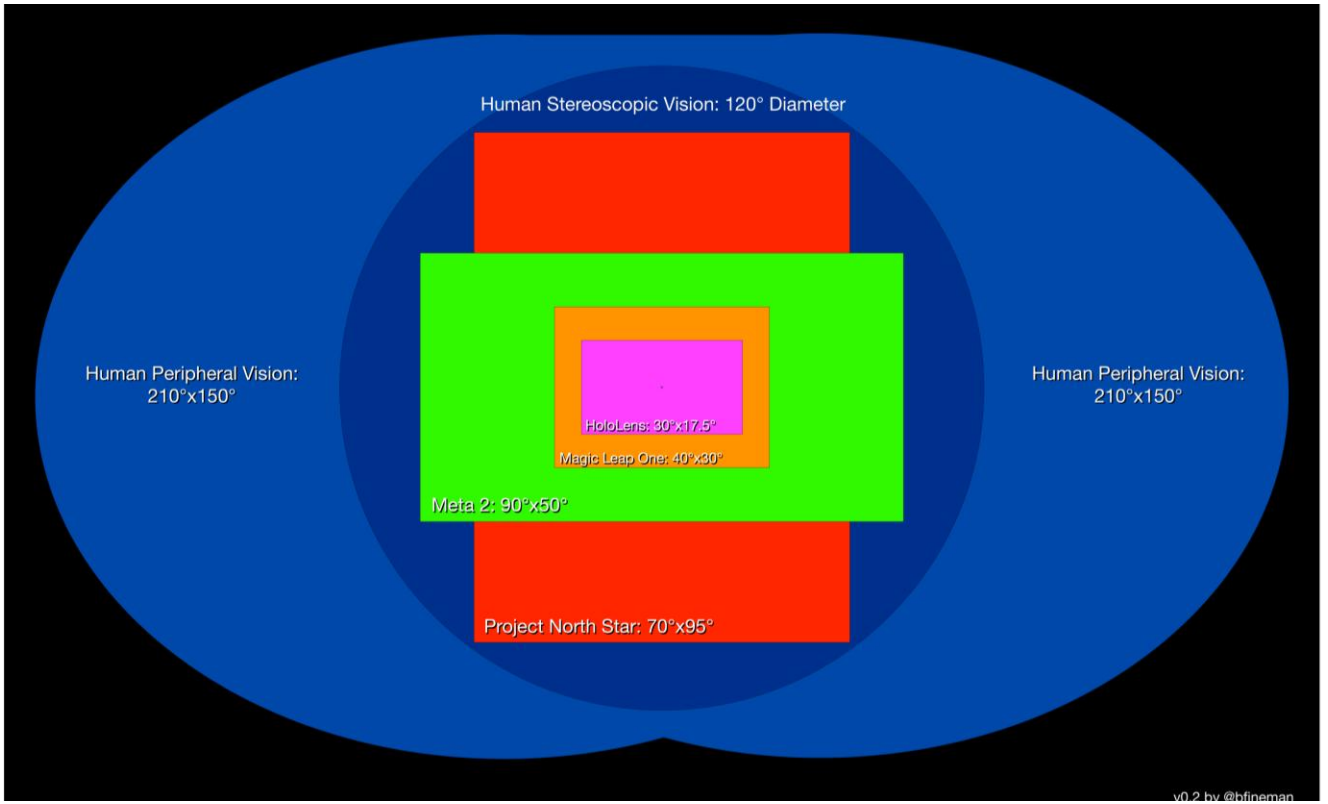




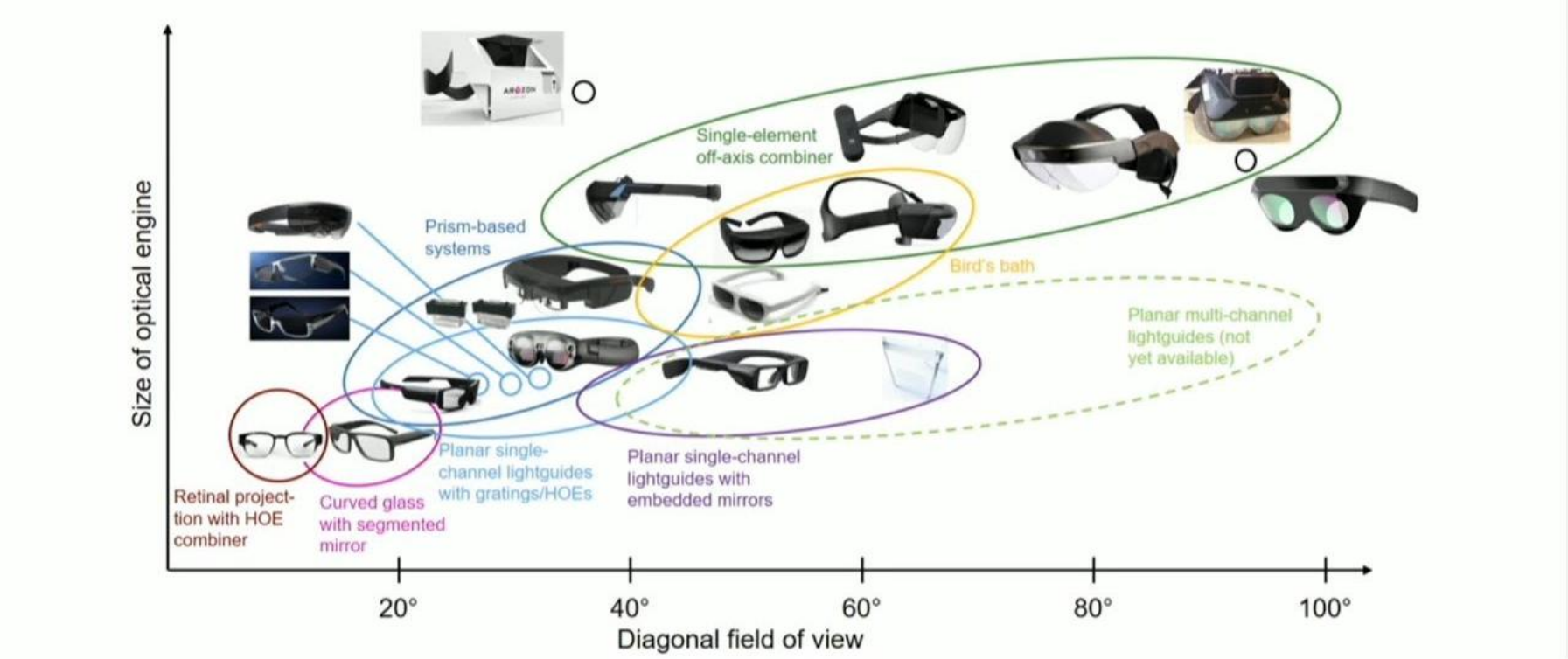
Example: Microsoft Hololens



Fields of View in AR HMDs



Quality of current AR HMDs



Pros and Cons of Optical see-thru AR

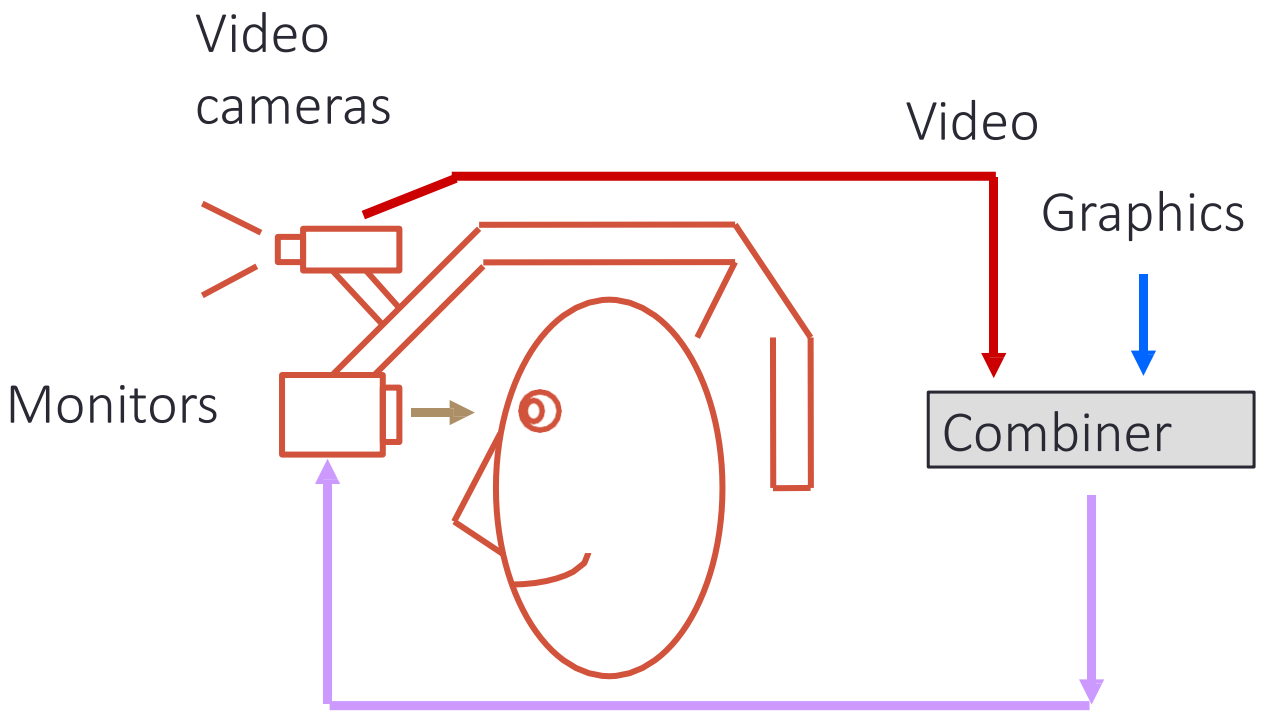
Pros

- Simpler design (cheaper)
- Direct view of real world
- No eye displacement
- Socially acceptable (glasses form factor)

Cons

- Difficult to occlude real world
- Image washout outdoors/bright lights
- Wide field of view challenging
- Can’t delay the real world

Video see-through HMD



Example: Apple Vision Pro

Pros and Cons of Video See-Thru AR

Pros

- True occlusion
- Digitized image of real world
- Registration, calibration, matchable time delay
- Wide FOV is easier to support

Cons

- Larger, bulkier hardware
- Can’t see real world with natural eyes

Multiplexed Display

Virtual Image ‘inset’ into Real World

Example: Google Glass

Link to video

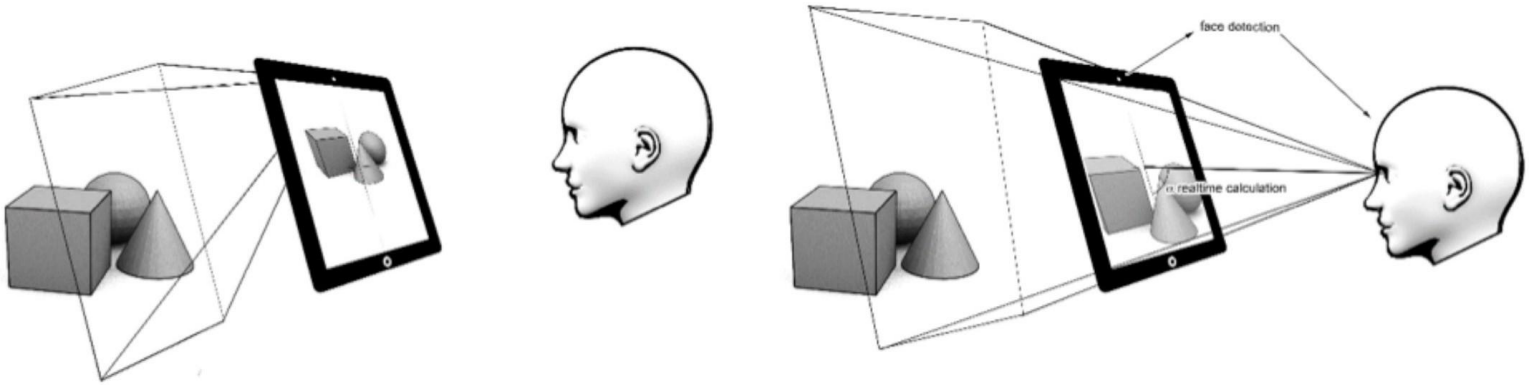
View Through Google Glass

AR Display - Handheld

Handheld AR

- Camera + display = handheld AR
- Mobile phone/Tablet display

Perspective Rendering for Handheld AR



(a) Device Perspective Rendering

(b) User Perspective Rendering

Perspective Rendering for Handheld AR

(a) Device Perspective Rendering

(b) User Perspective Rendering

AR Display - Spatial

Spatial AR

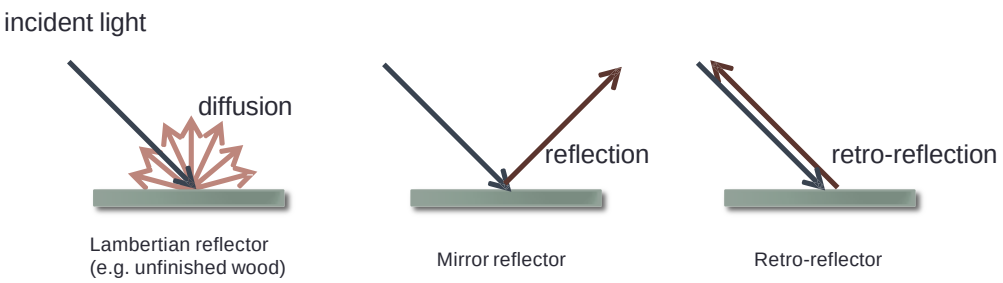
- Project onto Irregular Surfaces

Example: LED Billboard AR

Link to video

Example: Tilt Five

Retroreflective roll-able mat

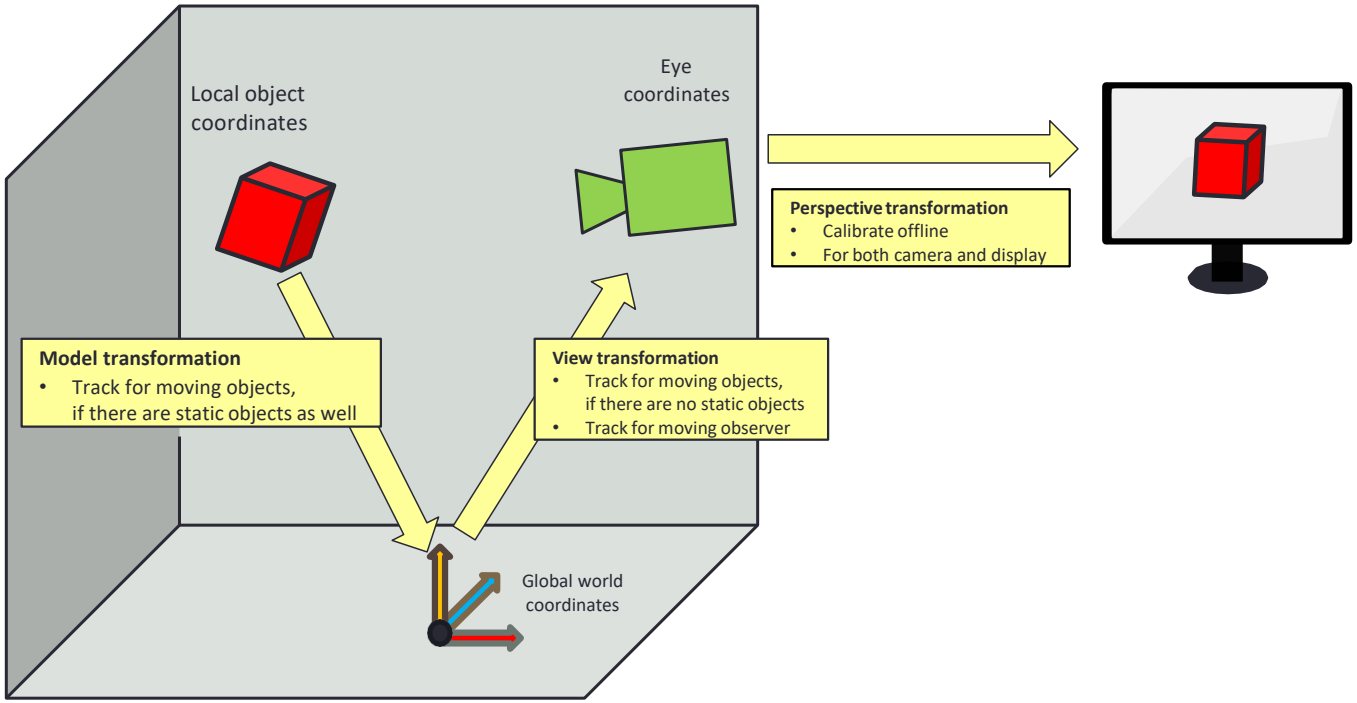


Example: Tilt Five

Link to video

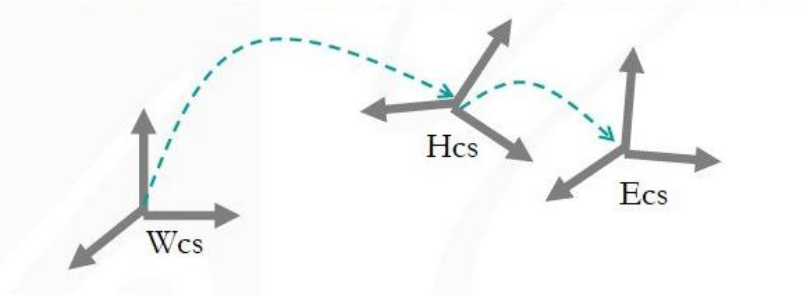
AR Tracking

Coordinate Systems



Spatial Registration

- Defining relative position of each element in the scene
- Elements are user, user's eye, environment and objects
- Transforms between coordinates
- Initial: Calibration
- Temporally: 3D/6D tracking



The Registration Problem

- Virtual and Real content must stay properly aligned
- If not:
  - Breaks the illusion that the two coexist
  - Prevents acceptance of many serious applications

Sources of Registration Errors

Static errors

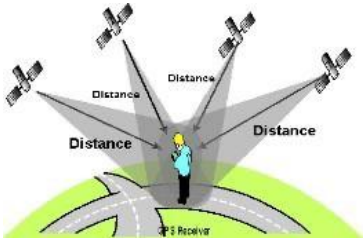
- Optical distortions (in HMD)
- Mechanical misalignments
- Tracker errors
- Incorrect viewing parameters

Dynamic errors

- System delays (largest source of error)
- 1 ms delay = 1/3 mm registration error

Tracking Technologies

- Active
  - Mechanical, Magnetic, Ultrasonic
  - GPS, Wifi, cell location



- Passive
  - Inertial sensors (compass, accelerometer, gyro)
  - Computer Vision (marker based, natural feature tracking)

- Hybrid Tracking
  - Combined sensors (e.g., vision + inertial)

AR Tracking - Optical

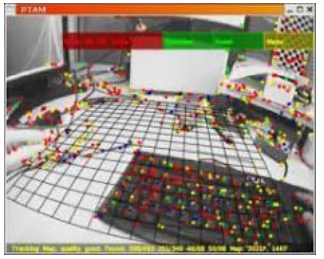
Link to video

Why Optical Tracking?

- Many AR devices have cameras
  - Cell phone/tablet, Video see-through display
- Provides precise alignment between video and AR overlay
  - Using features in video to generate pixel perfect alignment
  - Real world has many visual features that can be tracked from
- Computer Vision is a well-established discipline
  - Over 40 years of research to draw on
  - Old non-real time algorithms can be run in real time on today's devices

Common AR Optical Tracking Types

- Marker Tracking
  - Tracking known artificial markers/images
  - e.g. ARToolKit square markers
- Markerless Tracking
  - Tracking from known features in real world
  - e.g. Vuforia image tracking
- Unprepared Tracking
  - Tracking in unknown environment
  - e.g. SLAM (simultaneous localization and mapping) tracking



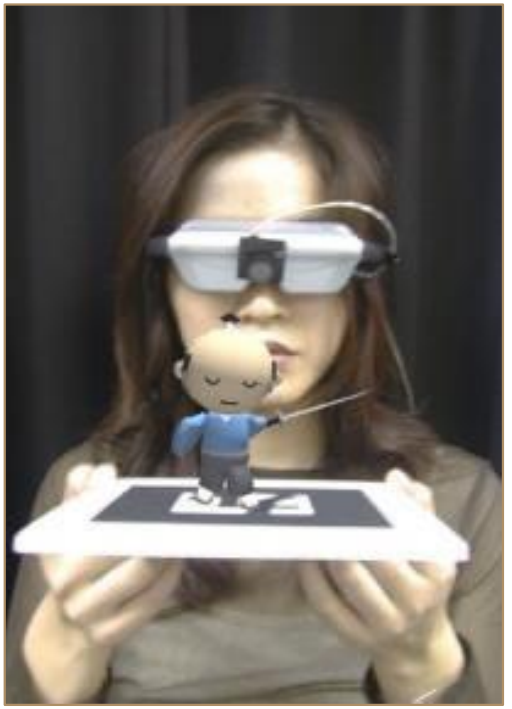
Visual Tracking Approaches

- Marker based tracking with artificial features
  - Make a model before tracking
- Model based tracking with natural features
  - Acquire a model before tracking
- Simultaneous localization and mapping
  - Build a model while tracking it

Optical Tracking - Markers

Marker Tracking

- Available for more than 20 years
- Several open-source solutions exist
  - ARToolKit, ARTag, ATK+, etc.
- Fairly simple to implement
  - Standard computer vision methods
- A rectangle provides 4 corner points
  - Enough for pose estimation!



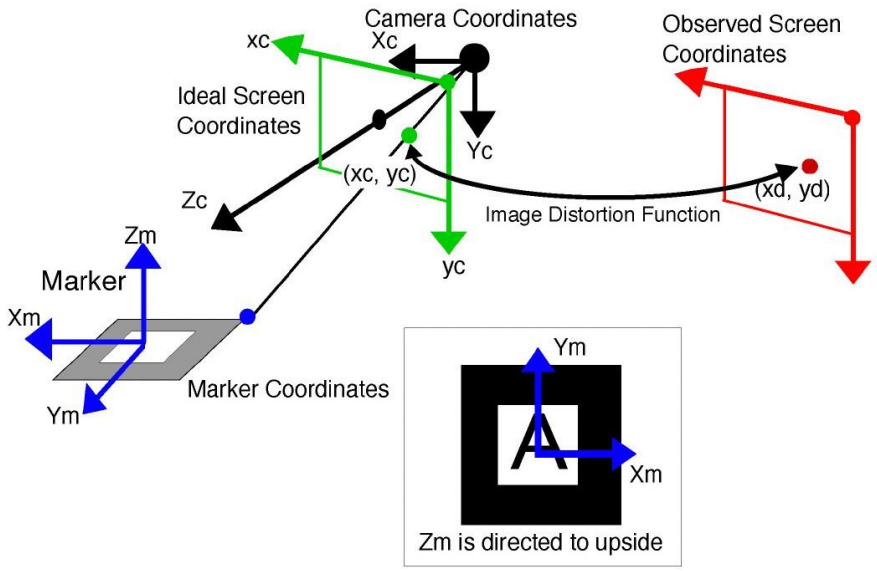
Key Problem: Finding Camera Position

Need camera pose relative to marker to render AR graphics

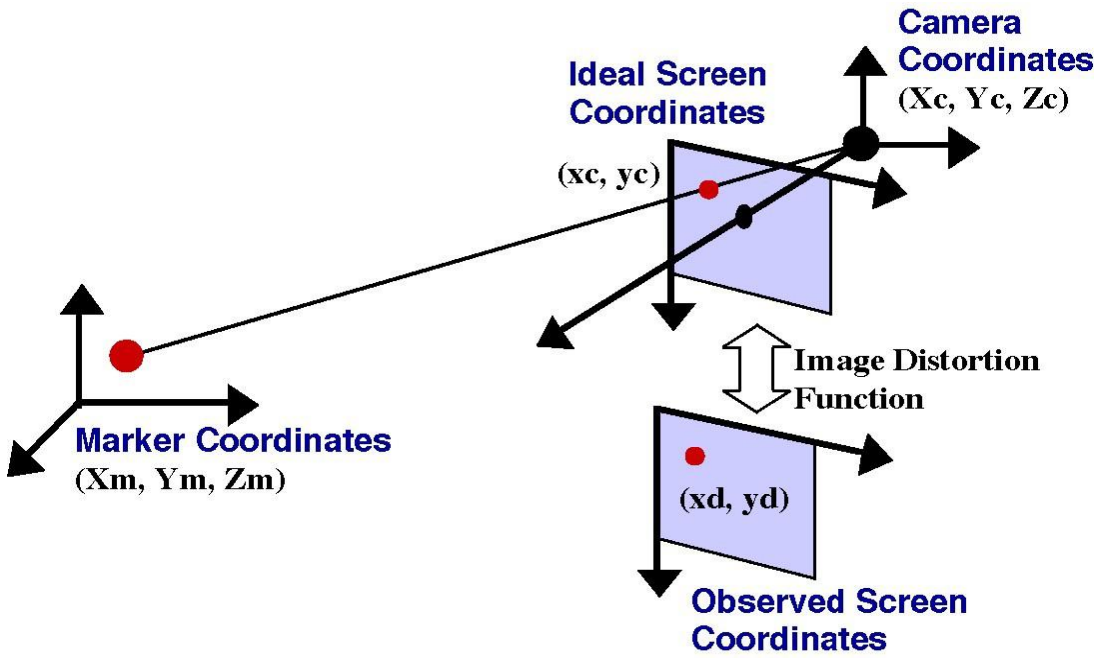
Known image → Image in Camera view → Overlay AR content

Goal: Find Camera Pose

- Knowing:
  - Position of key points in on-screen video image
  - Camera properties (focal length, image distortion)



Coordinates for Marker Tracking



Marker Tracking – General Principle

1. Capture image with known camera
2. Search for quadrilaterals
3. Pose estimation from homography
4. Pose refinement
5. Minimize nonlinear projection error

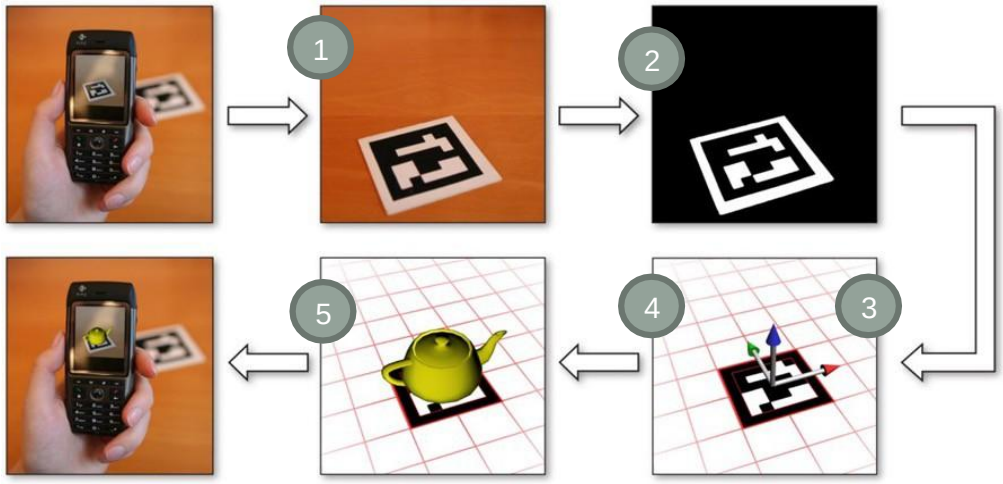
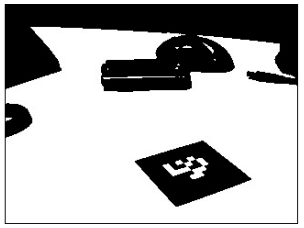
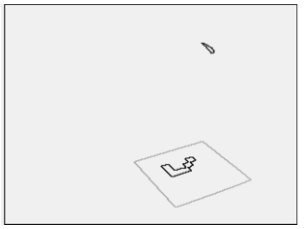


Image: Daniel Wagner

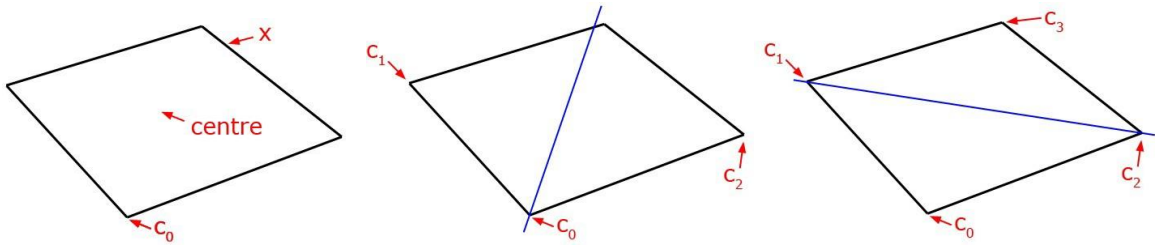
Marker Tracking – Fiducial Detection

- Threshold the whole image to black and white
- Search scanline by scanline for edges (white to black)
- Follow edge until either
  - Back to starting pixel
  - Image border
- Check for size
  - Reject fiducials early that are too small (or too large)

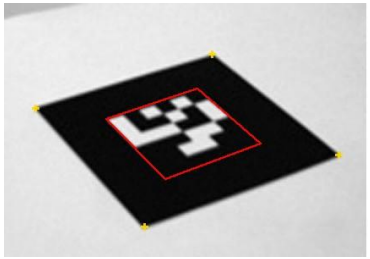
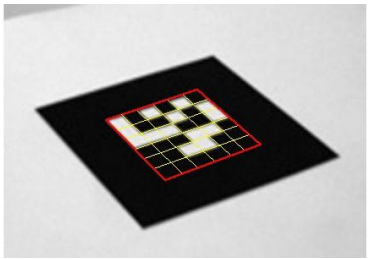
Marker Tracking – Rectangle Fitting

- Start with an arbitrary point “x” on the contour
- The point with maximum distance must be a corner c0
- Create a diagonal through the center
- Find points c1 & c2 with maximum distance left and right of diag.
- New diagonal from c1 to c2
- Find point c3 right of diagonal with maximum distance



Marker Tracking – Pattern checking

- Calculate homography using the 4 corner points
- Extract pattern by sampling and check
  - Id (implicit encoding)
  - Template (normalized cross correlation)


Marker tracking – Pose estimation

- Calculates marker pose relative to the camera
- Initial estimation directly from homography
  - Very fast, but coarse with error
  - Jitters a lot…
- Iterative Refinement using Gauss-Newton method
  - 6 parameters (3 for position, 3 for rotation) to refine
  - At each iteration we optimize on the error
- Iterate

Outcome: Camera Transform

- Transformation from Marker to Camera
- Rotation and Translation

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \mathbf{T_{CM}} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix}$$

$\mathbf{T_{CM}}$: 4x4 transformation matrix
from marker coord. to camera coord.

Questions?

Resources

- Documentation:
  - Three.js: https://threejs.org/docs/
  - A-Frame: https://aframe.io/docs/
  - AR.js: https://ar-js-org.github.io/AR.js-Docs/