ICCS-404

Computer Graphics and ~~Augmented~~ Reality

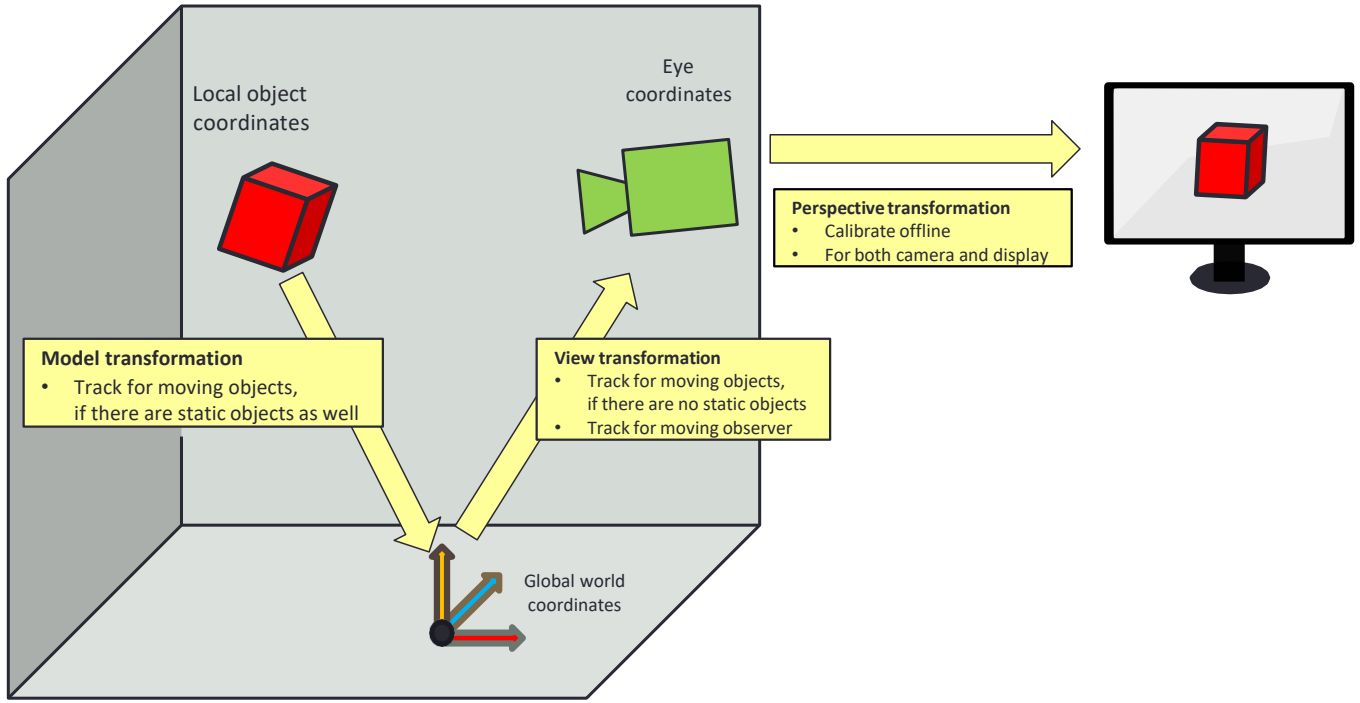
EXTENDED

Pisut Wisessing, PhD
pisut@cmkl.ac.th

Assistant Professor
CMKL University

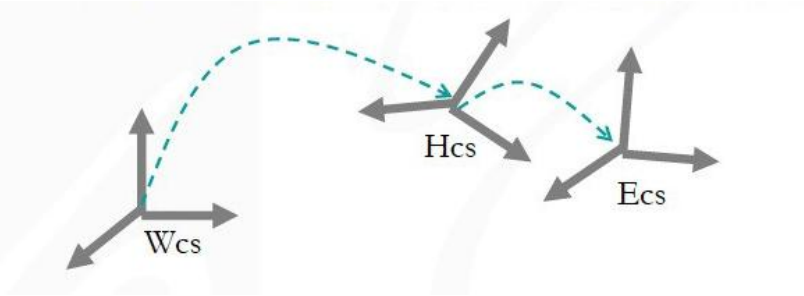
AR Tracking

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

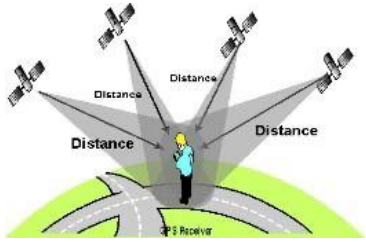
- Optical distortions (in HMD)
- Mechanical misalignments
- Tracker errors
- Incorrect viewing parameters

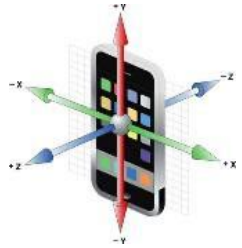
Dynamic errors

- 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

Example: Optical Tracking

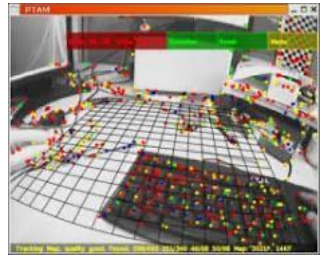
Link to video

Why Optical Tracking for AR?

- 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

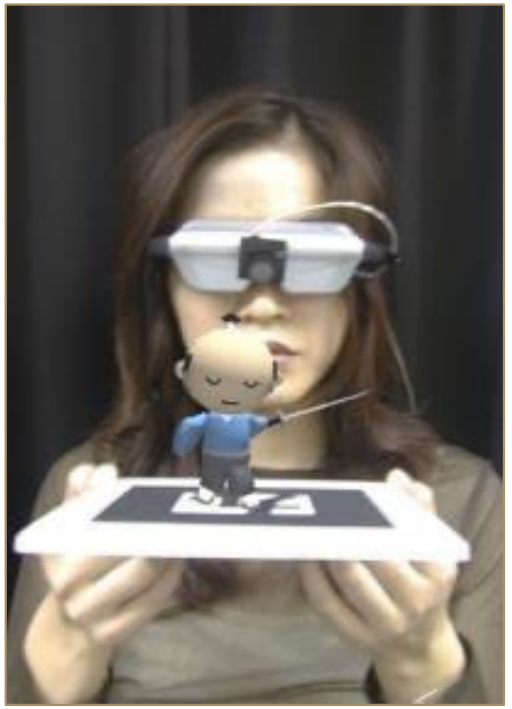
- Marker Tracking
  - Tracking known artificial markers/images
  - e.g. ARToolKit square markers
- Markerless Tracking
  - Tracking from known features in real world
  - e.g. Vuforia image tracking
- Unprepared Tracking
  - Tracking in unknown environment
  - e.g. SLAM (simultaneous localization and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



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

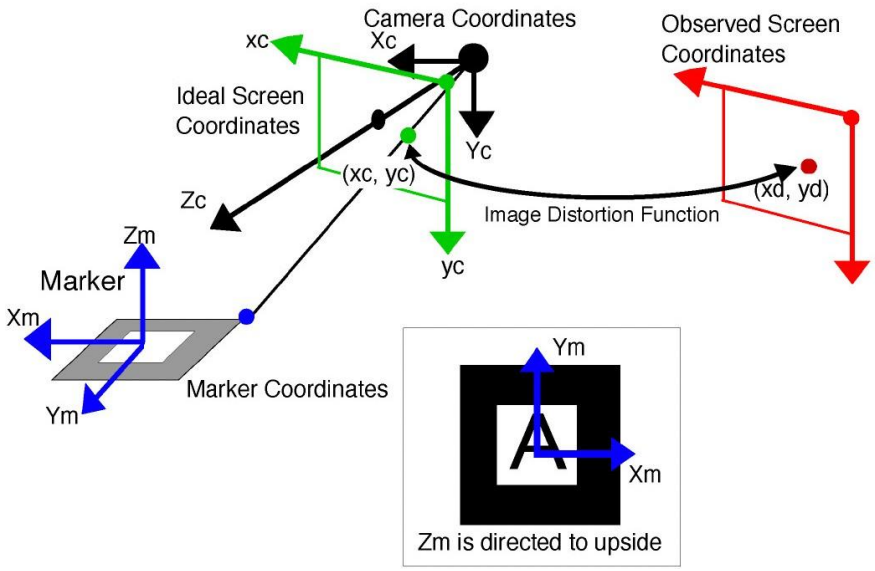
Known image

Image in Camera view

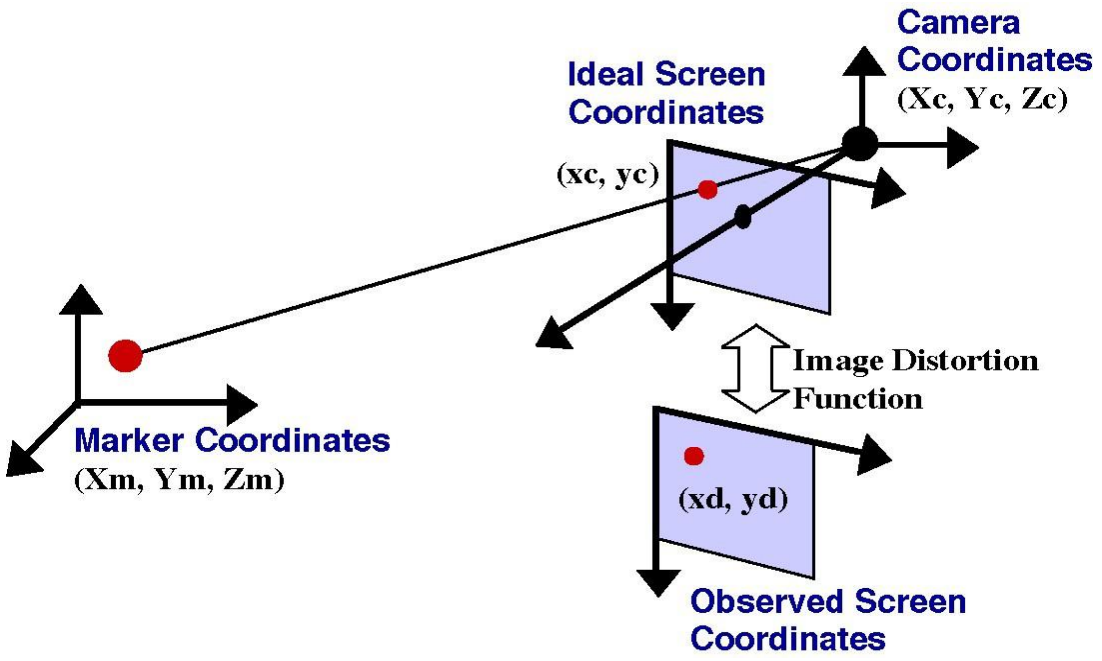
Overlay AR content

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

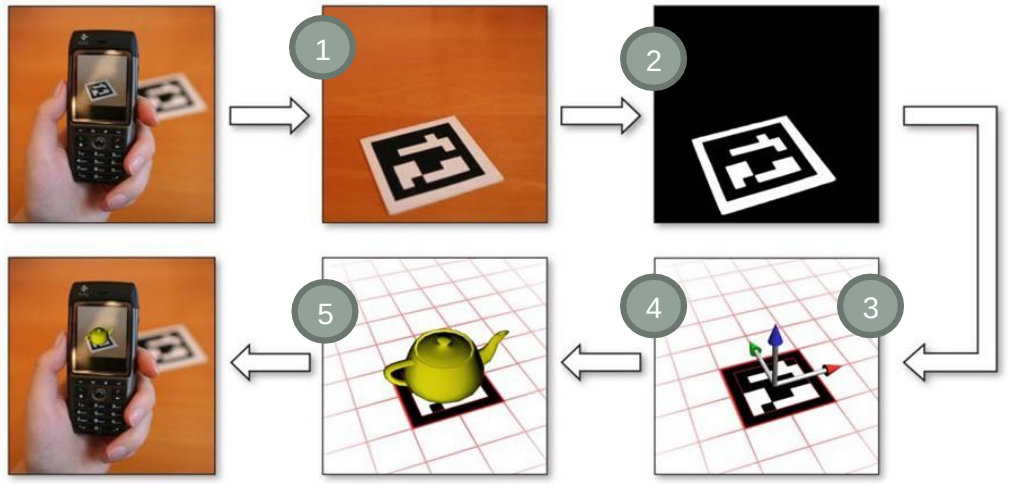
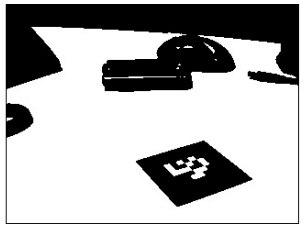
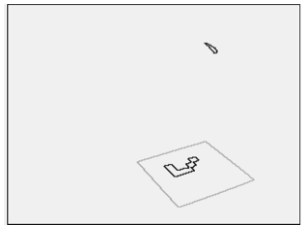


Image: Daniel Wagner

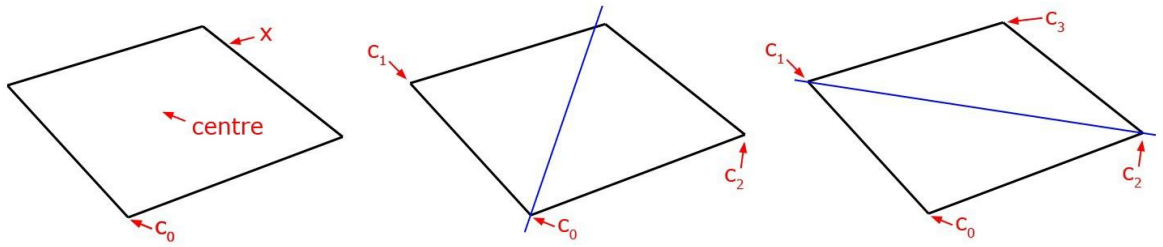
Marker Tracking – Fiducial Detection

- 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)

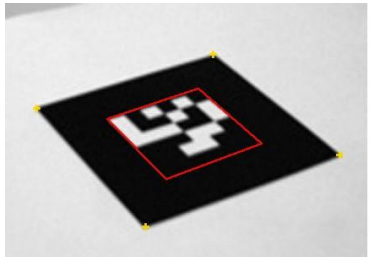
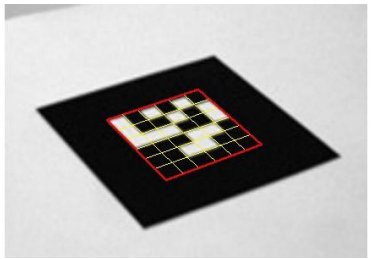
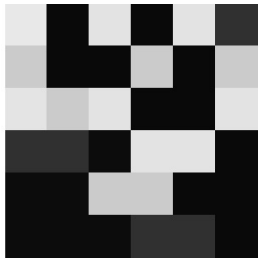
Marker Tracking – Rectangle Fitting

- 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

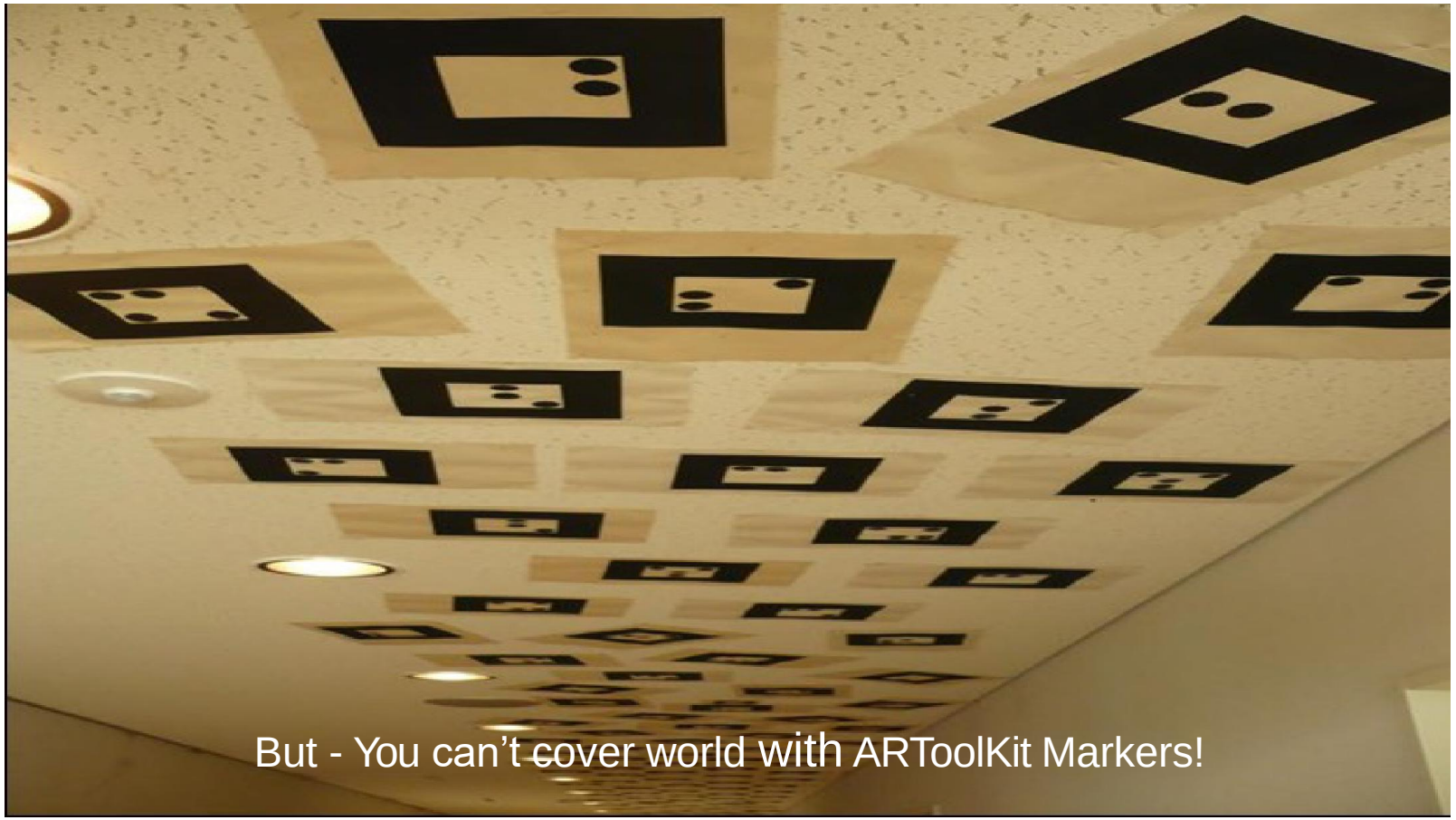
- Calculates marker pose relative to the camera
- Initial estimation directly from homography
  - Very fast, but coarse with error
  - Jitters a lot…
- Iterative Refinement using Gauss-Newton method
  - 6 parameters (3 for position, 3 for rotation) to refine
  - At each iteration we optimize on the error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

Outcome: Camera Transform

- Transformation from Marker to Camera
- Rotation and Translation

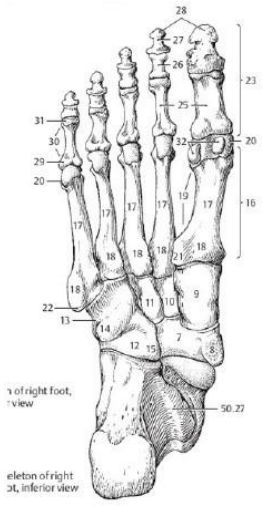
$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \mathbf{T_{CM}} \begin{bmatrix} X_m \\ Y_m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

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


But - You can’t cover world with ARToolKit Markers!

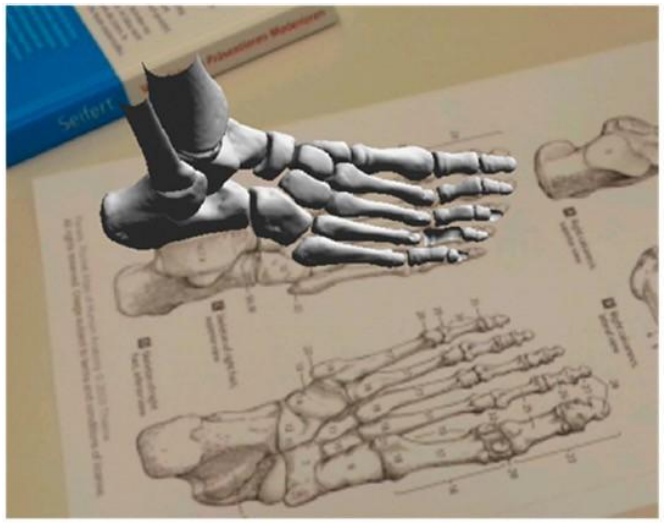
Optical Tracking - Markerless

Markerless Track



Target Image

Feature Detection



AR Overlay

Example: Sayduck/Vuforia

Link to video

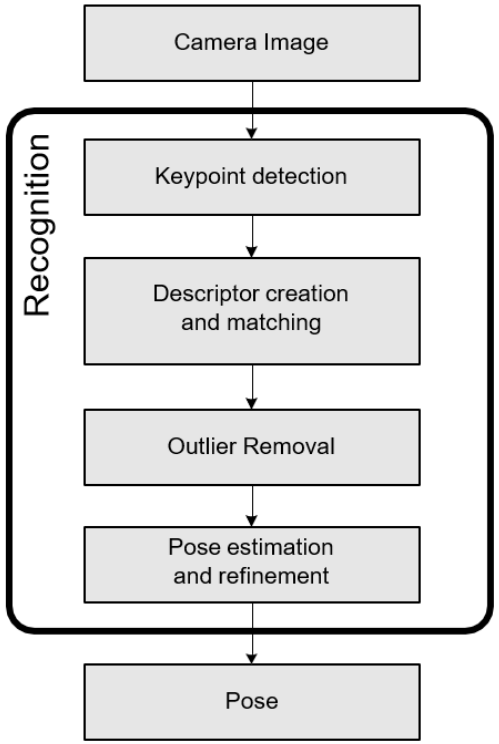
Texture Tracking

Example: Vuforia Texture Tracking

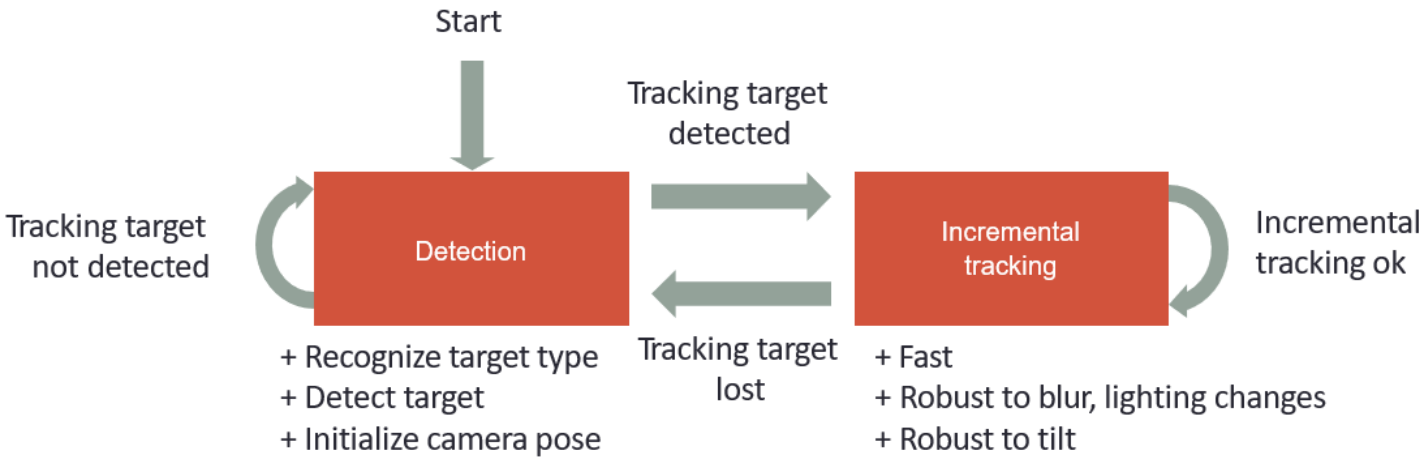
Link to video

Tracking by Keypoint Detection

- This is what most trackers do…
- Targets are detected every frame
- Popular because tracking and detection are solved simultaneously



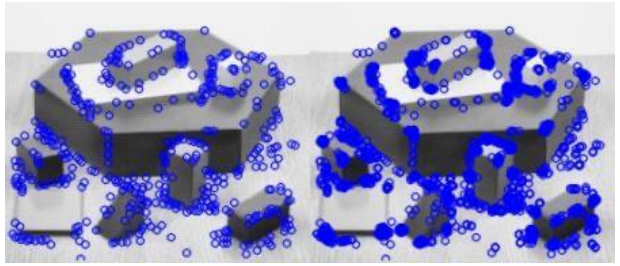
Detection and Tracking



- Tracking and detection are complementary approaches.
- After successful detection, the target is tracked incrementally.
- If the target is lost, the detection is activated again

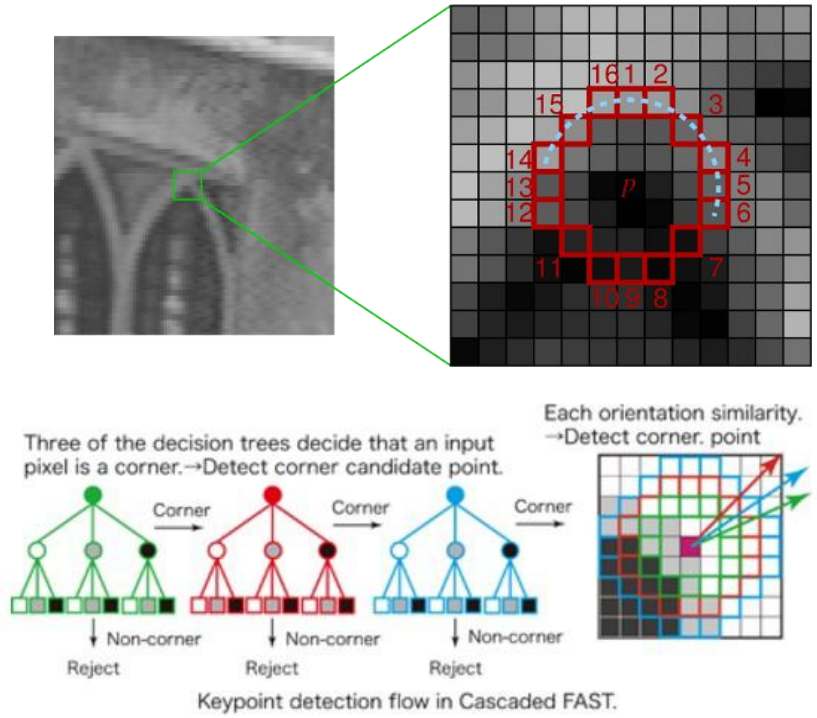
What is a Keypoint?

- It depends on the detector you use!
- For high performance use the FAST corner detector
  - Apply FAST to all pixels of your image
  - Obtain a set of keypoints for your image
  - Describe the keypoints



Rosten, E., & Drummond, T. (2006, May). Machine learning for high-speed corner detection. In *European conference on computer vision* (pp. 430-443). Springer Berlin Heidelberg.

FAST Corner Keypoint Detection



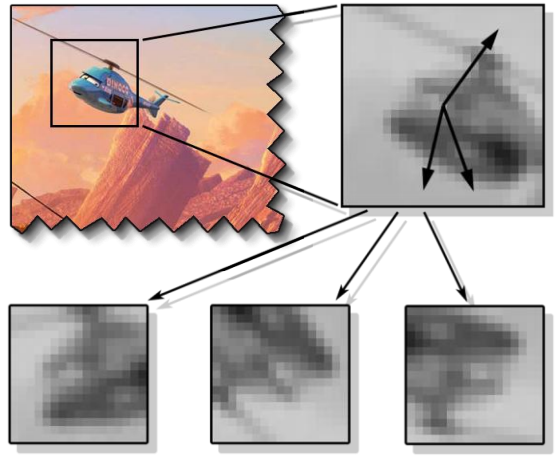
Keypoint detection flow in Cascaded FAST.

Example: FAST Corner Detection

Link to video

Descriptors

- Describe the Keypoint features
  - Estimate the dominant keypoint orientation using gradients
  - Compensate for detected orientation
  - Describe the keypoints in terms of the gradients surrounding it



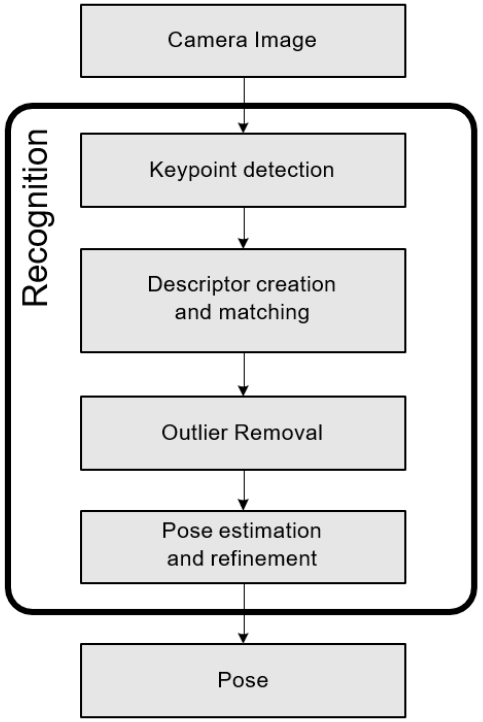
Wagner D., Reitmayr G., Mulloni A., Drummond T., Schmalstieg D., Real-Time Detection and Tracking for Augmented Reality on Mobile Phones. IEEE Transactions on Visualization and Computer Graphics, May/June, 2010

Database Creation

- Offline step – create database of known features
- Searching for corners in a static image
- For robustness look at corners on multiple scales
  - Some corners are more descriptive at larger or smaller scales
  - We don't know how far users will be from our image
- Build a database file with all descriptors and their position on the original image

Real-time Tracking

- Search for known keypoints in the video
- Create the descriptors
- Match the descriptors from the
- live video against those in the database
  - Brute force is not an option
  - Need the speed-up of special data structures

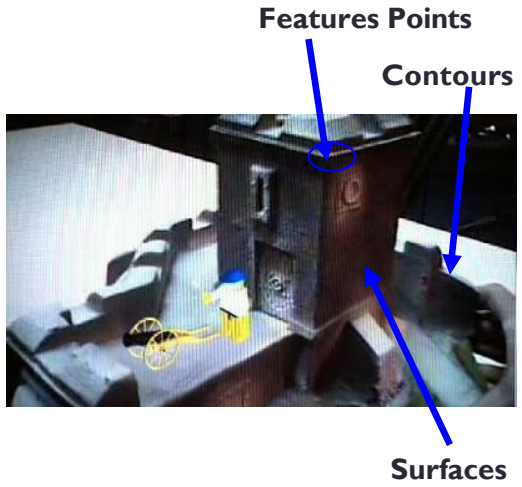


Example

Link to video

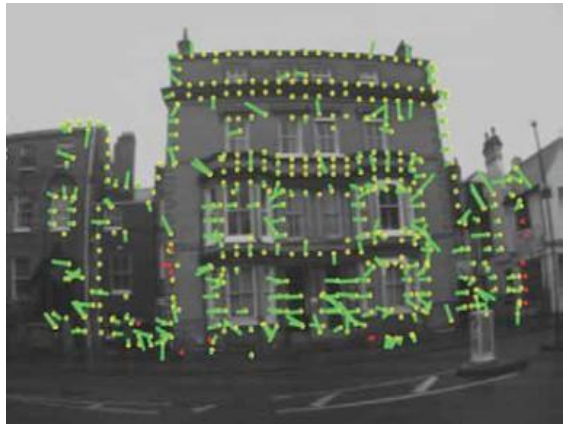
Natural Feature Tracking (NFT)

- Use Natural Cues of Real Elements
  - Edges
  - Surface Texture
  - Interest Points
- Model or Model-Free
- No visual pollution



Natural Features

- Detect salient interest points in image
  - Must be easily found
  - Location in image should remain stable when viewpoint changes
  - Requires textured surfaces



- Match interest points to tracking model database
  - Database filled with results of 3D reconstruction
  - Matching entire (sub-)images is too costly
  - Typically interest points are compiled into “descriptors”

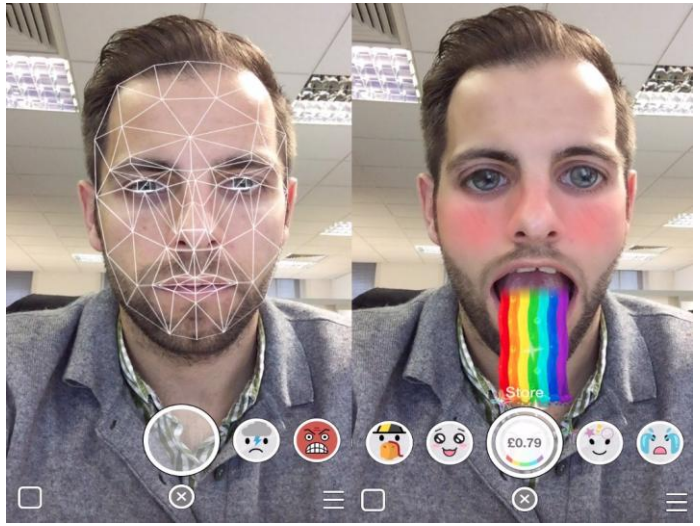
3D Model Based Tracking

Tracking from 3D object shape

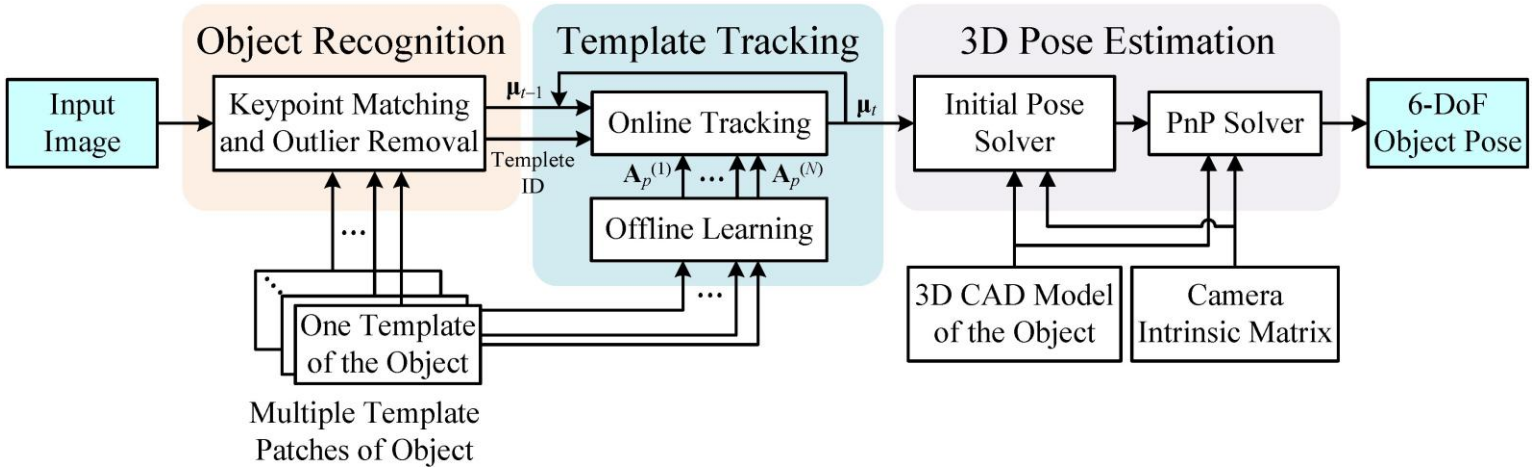
- Align detected features to 3D object model

Examples

- SnapChat Face tracking
- Mechanical part tracking
- Vehicle tracking
- Etc...

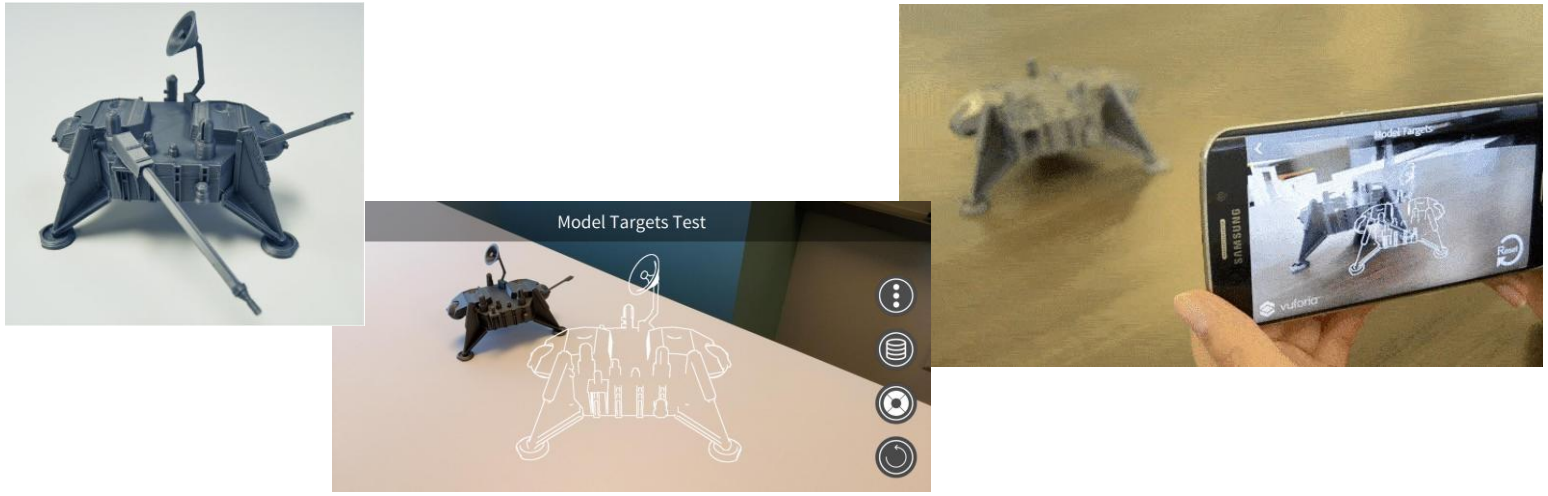


Typical Model Based Tracking Algorithm



Example: Vuforia Model Tracker

- Uses pre-captured 3D model for tracking
- On-screen guide to line up model



Model Tracking Demo

Link to video

Optical Tracking - Unprepared

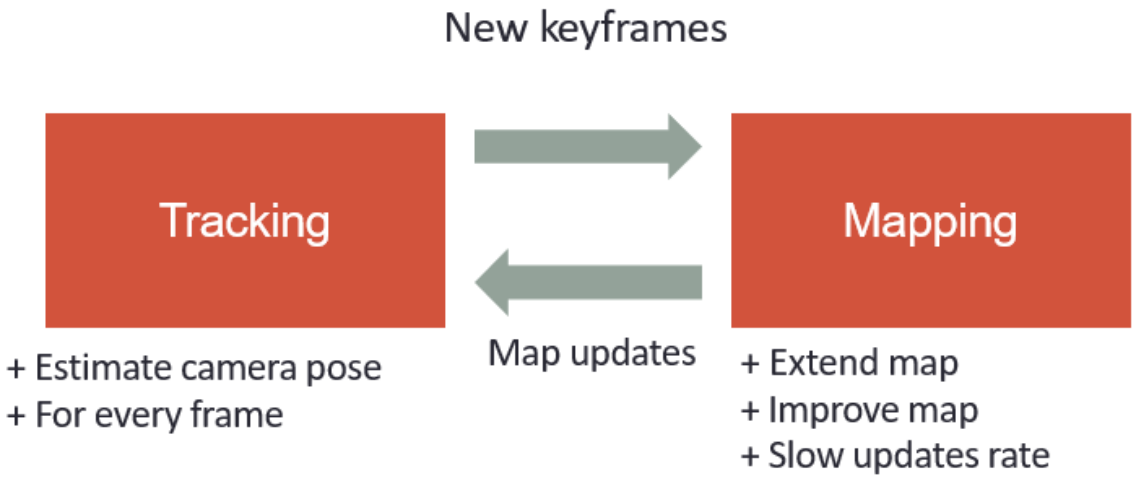
Example: HoloLens

Link to video

Tracking from an Unknown Environment

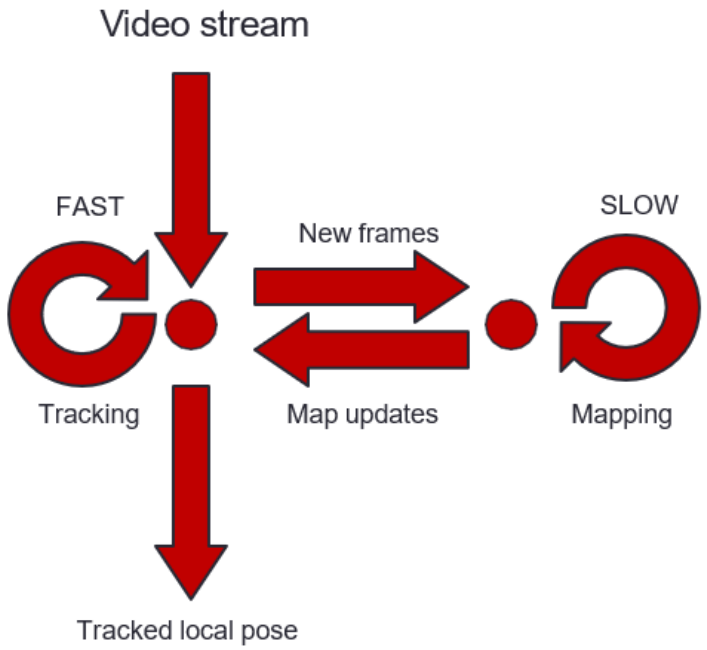
- What to do when you don't know any features?
  - Very important problem in mobile robotics - Where am I?
- SLAM (Simultaneously Localize And Map the environment)
  - *Goal:* to recover both camera pose and map structure while initially knowing neither.
  - *Mapping:* Building a map of the environment which the robot is in
  - *Localization:* Navigating this environment using the map while keeping track of the robot's relative position and orientation

Parallel Tracking and Mapping



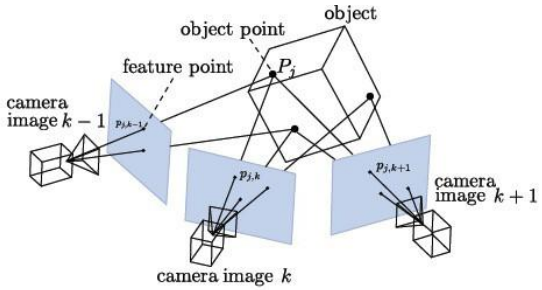
Parallel tracking and mapping uses two concurrent threads, one for tracking and one for mapping, which run at different speeds

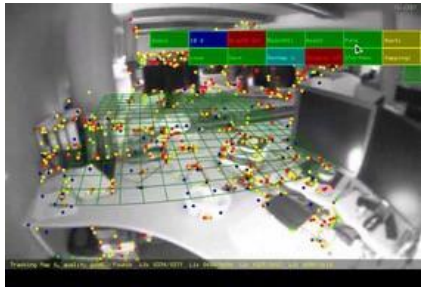
Parallel Tracking and Mapping



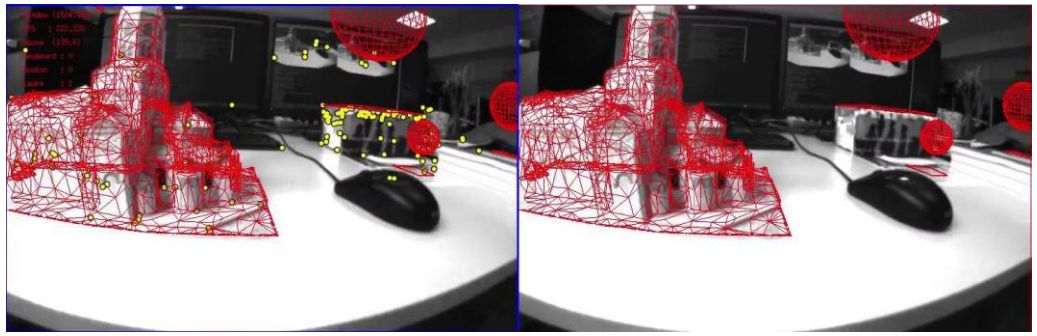
Visual SLAM

- Early SLAM systems (1986 -)
  - Computer visions and sensors (e.g. IMU, laser, etc.)
  - One of the most important algorithms in Robotics
- Visual SLAM
  - Using cameras only, such as stereo view
  - MonoSLAM (single camera) developed in 2007 (Davidson)





How SLAM Works



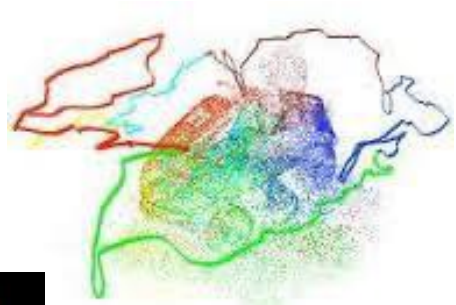
- Three main steps
  1. Tracking a set of points through successive camera frames
  2. Using these tracks to triangulate their 3D position
  3. Simultaneously use the estimated point locations to calculate the camera pose which could have observed them

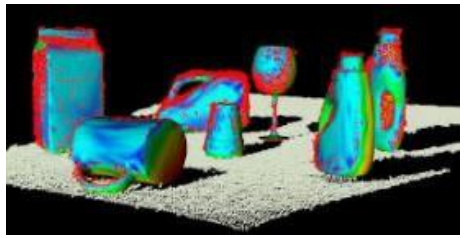
  By observing enough points can solve for both structure and motion (camera path and scene structure).

Applications of SLAM Systems

Many possible applications

- Augmented Reality camera tracking
- Mobile robot localization
- Real world navigation aid
- 3D scene reconstruction
- 3D Object reconstruction





Assumptions

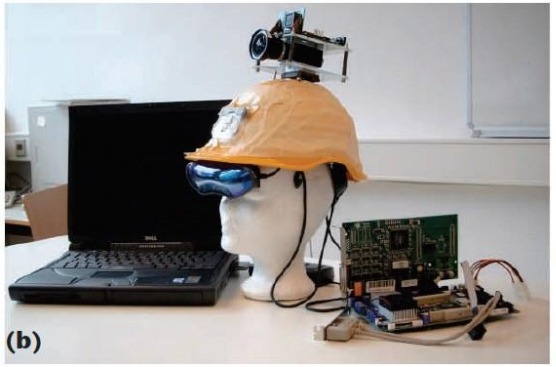
- Camera moves through an unchanging scene
- Not suitable for person tracking or gesture recognition

AR Tracking - Hybrid

Hybrid Tracking Interfaces

Combine multiple tracking technologies together

- Active-Passive: magnetic, vision
- Active-Inertial: vison, inertial
- Passive-Inertial: compass, inertial



Combining Sensors and Vision

Sensors

- Produces noisy output (= jittering augmentations)
- Are not sufficiently accurate (= wrongly placed augmentations)
- Gives us first information on where we are in the world, and what we are looking at

Vision

- Is more accurate (= stable and correct augmentations)
- Requires choosing the correct keypoint database to track from
- Requires registering our local coordinate frame (online- generated model) to the global one (world)

Types of Sensor Fusion

Complementary

- Combining sensors with different degrees of freedom
- Sensors must be synchronized (or requires inter-/extrapolation)
- E.g., combine position-only and orientation-only sensor
- E.g., orthogonal 1D sensors in gyro or magnetometer are complementary

Competitive

- Different sensor types measure the same degree of freedom
- Redundant sensor fusion
- Use worse sensor only if better sensor is unavailable
- E.g., GPS + pedometer
- Statistical sensor fusion

Example: Outdoor Hybrid Tracking

Combines

- computer vision
- inertial gyroscope sensors

Both correct for each other

- Inertial gyro
  - provides frame to frame prediction of camera orientation, fast sensing
  - drifts over time
- Computer vision
  - Natural feature tracking, corrects for gyro drift
  - Slower, less accurate

AR Input

Input Methods

- Handheld Controller
  - Multiple buttons, trackpad input
  - 6 DOF magnetic tracking
- Eye gaze
  - Integrated eye tracking
- Hand tracking
  - Natural hand input

Controller Input

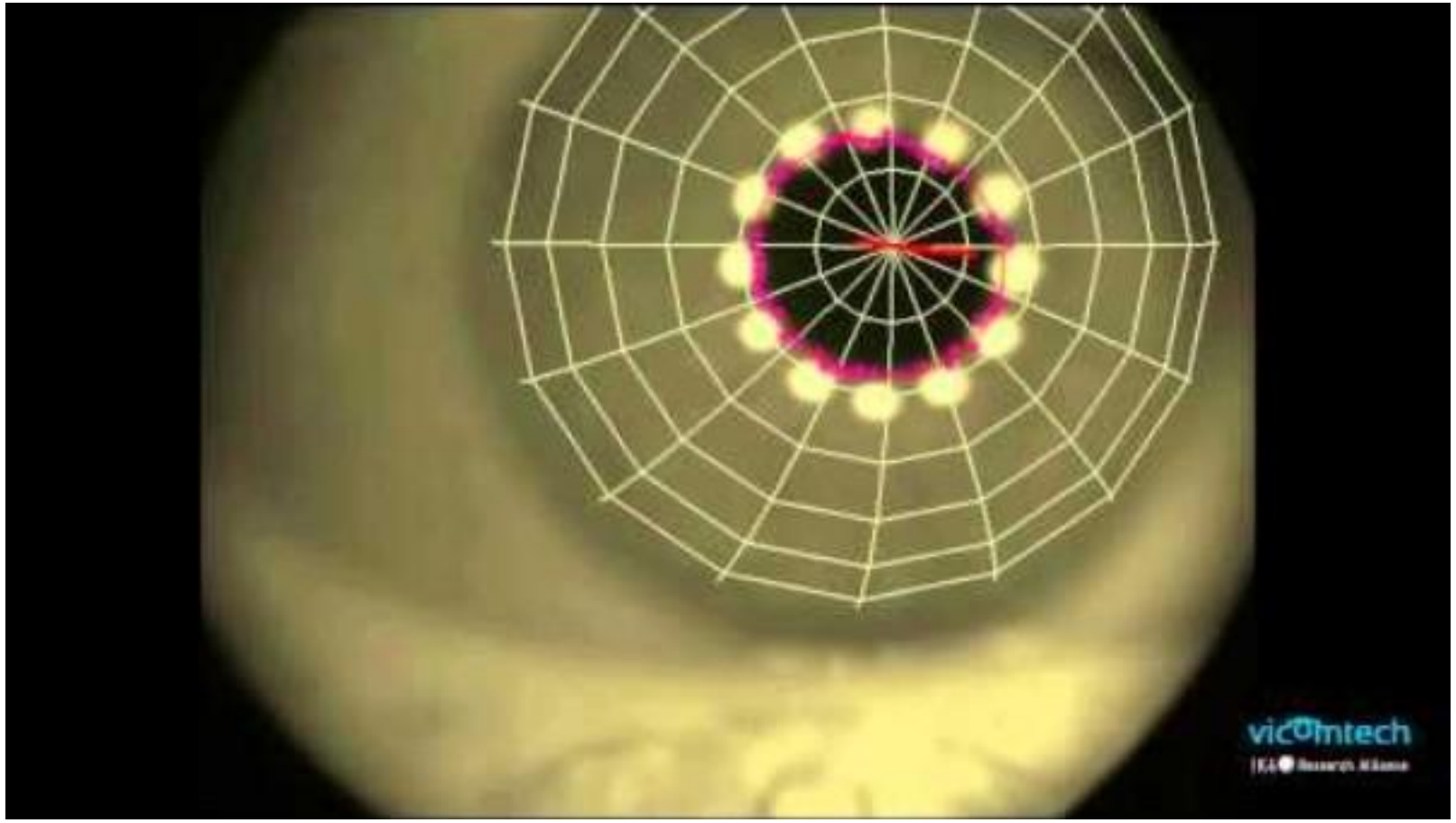
Magnetic tracking (6 DoF)

Receiver Coils

Transmitter Coils

Eye Gaze

Link to video



Hand Tracking

Link to video

Questions?

Resources

- Documentation:
  - Three.js: https://threejs.org/docs/
  - A-Frame: https://aframe.io/docs/