Pose Tracking II

CS 6334 Virtual Reality

Professor Yapeng Tian
The University of Texas at Dallas

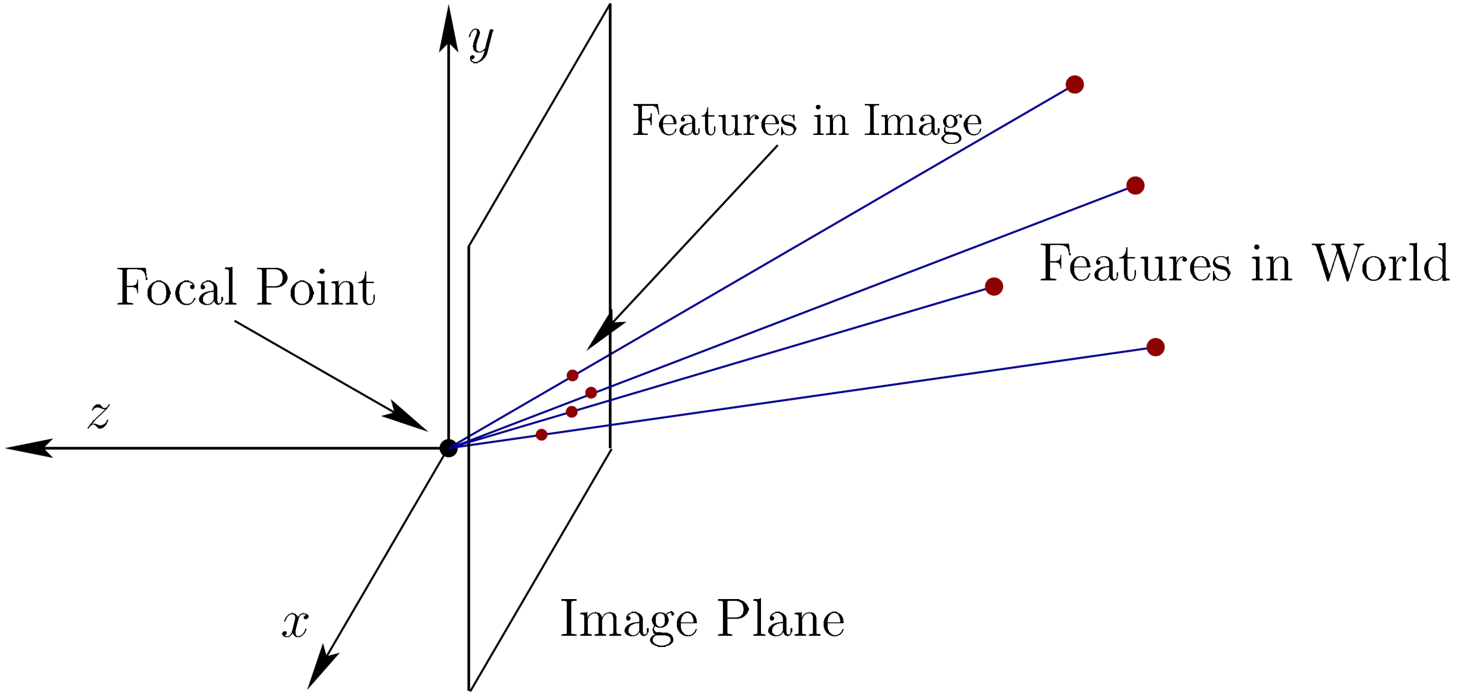
Tracking in VR

- Tracking the user's sense organs
  - E.g., Head and eye
  - Render stimulus accordingly
- Tracking user's other body parts
  - E.g., human body and hands
  - Locomotion and manipulation
- Tracking the rest of the environment
  - Augmented reality
  - Obstacle avoidance in the real world

Oculus Pose Tracking

https://youtu.be/nrj3JE-NHMw

Feature-based Tracking



The PnP problem

- Known: 3D locations, 2D locations, camera intrinsics
- Unknown:

  6D pose of the camera

Passive features

- Image features
- Markers

Active features

- LEDs

The PnP Problem

- Many different algorithms to solve the PnP problem

- General idea
  - Retrieve the coordinates of the 3D points in the camera coordinate system $\mathbf{p}_i^c$
  - Compute rotation and translation that align the world coordinates and the camera coordinates

$$\mathbf{p}_i^w \xrightarrow{R, T} \mathbf{p}_i^c$$

EPnP

- EPnP: uses 4 control points $\mathbf{c}_j, \quad j = 1, \ldots, 4$

3D coordinates in the world frame $\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w$ Known

Weights $\sum_{j=1}^{4} \alpha_{ij} = 1$ Known

3D coordinates in the camera frame $\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$ Unknown

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.

EPnP

- Projection of the points in the camera frame

$$\forall i\ ,\ w_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = K\mathbf{p}_i^c = K\sum_{j=1}^{4} \alpha_{ij}\mathbf{c}_j^c$$

$$\forall i\ ,\ w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}$$

Unknown $\{(x_j^c, y_j^c, z_j^c)\}_{j=1,\dots,4}$ $\{w_i\}_{i=1,\dots,n}$ $\quad w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV’09.

EPnP

$$\forall i\ ,\ w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix}$$

$$w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$$

$$\sum_{j=1}^{4} \alpha_{ij} f_u x_j^c + \alpha_{ij}(u_c - u_i) z_j^c = 0$$

$$\sum_{j=1}^{4} \alpha_{ij} f_v y_j^c + \alpha_{ij}(v_c - v_i) z_j^c = 0$$

Unknown $\left\{(x_j^c, y_j^c, z_j^c)\right\}_{j=1,\dots,4}$

$$\mathbf{M}\mathbf{x} = \mathbf{0}$$

$$\mathbf{x} = \left[\mathbf{c}_1^{c\top}, \mathbf{c}_2^{c\top}, \mathbf{c}_3^{c\top}, \mathbf{c}_4^{c\top}\right]^\top \ 12 \times 1$$

$\mathbf{M}$ is a $2n \times 12$ matrix

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV’09.

EPnP

- Solve $\mathbf{M}\mathbf{x} = \mathbf{0}$ to obtain $\mathbf{x} = [\mathbf{c}_1^{c\top}, \mathbf{c}_2^{c\top}, \mathbf{c}_3^{c\top}, \mathbf{c}_4^{c\top}]^\top$
- Compute 3D coordinates in camera frame $\mathbf{p}_i^c = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^c$
- We know the 3D coordinates in world frame $\mathbf{p}_i^w = \sum_{j=1}^{4} \alpha_{ij} \mathbf{c}_j^w$
- Compute R and T using the two sets of 3D coordinates

$$\mathbf{p}_i^w \xrightarrow{R,T} \mathbf{p}_i^c$$

EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.

Rotation and Translation from Two Point Sets

$$\mathbf{p}_i^w \xrightarrow{R,T} \mathbf{p}_i^c$$

Closed-form solution

K.S. Arun, T.S. Huang, and S.D. Blostein. Least-Squares Fitting of Two 3-D Points Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

$$\Sigma^2 = \sum_{i=1}^{N} \left\| p_i' - (Rp_i + T) \right\|^2.$$

Or https://cs.gmu.edu/~kosecka/cs685/cs685-icp.pdf

PnP in practice

- SolvePnPMethod in OpenCV

◆ SolvePnPMethod

enum **cv::SolvePnPMethod**

```
#include <opencv2/calib3d.hpp>
```

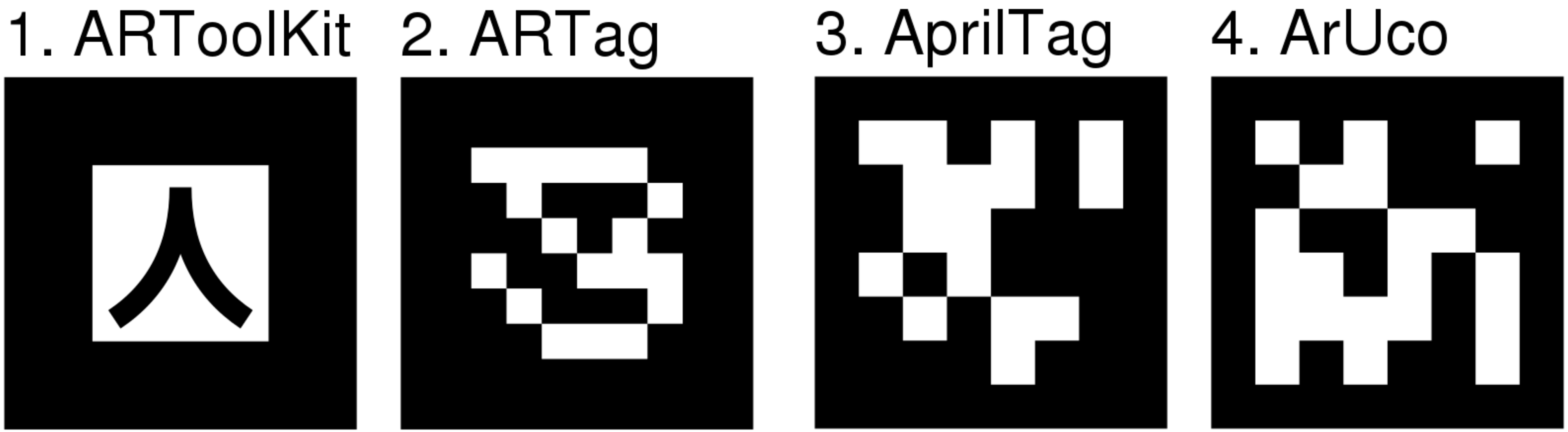
Enumerator	
SOLVEPNP_ITERATIVE Python: cv.SOLVEPNP_ITERATIVE	
SOLVEPNP_EPNP Python: cv.SOLVEPNP_EPNP	EPnP: Efficient Perspective-n-Point Camera Pose Estimation [125].
SOLVEPNP_P3P Python: cv.SOLVEPNP_P3P	Complete Solution Classification for the Perspective-Three-Point Problem [80].
SOLVEPNP_DLS Python: cv.SOLVEPNP_DLS	**Broken implementation. Using this flag will fallback to EPnP.** A Direct Least-Squares (DLS) Method for PnP [101]
SOLVEPNP_UPNP Python: cv.SOLVEPNP_UPNP	**Broken implementation. Using this flag will fallback to EPnP.** Exhaustive Linearization for Robust Camera Pose and Focal Length Estimation [169]
SOLVEPNP_AP3P Python: cv.SOLVEPNP_AP3P	An Efficient Algebraic Solution to the Perspective-Three-Point Problem [114].
SOLVEPNP_IPPE Python: cv.SOLVEPNP_IPPE	Infinitesimal Plane-Based Pose Estimation [46] Object points must be coplanar.
SOLVEPNP_IPPE_SQUARE Python: cv.SOLVEPNP_IPPE_SQUARE	Infinitesimal Plane-Based Pose Estimation [46] This is a special case suitable for marker pose estimation. 4 coplanar object points must be defined in the following order: • point 0: [-squareLength / 2, squareLength / 2, 0] • point 1: [squareLength / 2, squareLength / 2, 0] • point 2: [squareLength / 2, -squareLength / 2, 0] • point 3: [-squareLength / 2, -squareLength / 2, 0]
SOLVEPNP_SQPNP Python: cv.SOLVEPNP_SQPNP	SQPnP: A Consistently Fast and Globally OptimalSolution to the Perspective-n-Point Problem [208].

QR Code Pose Tracking Example

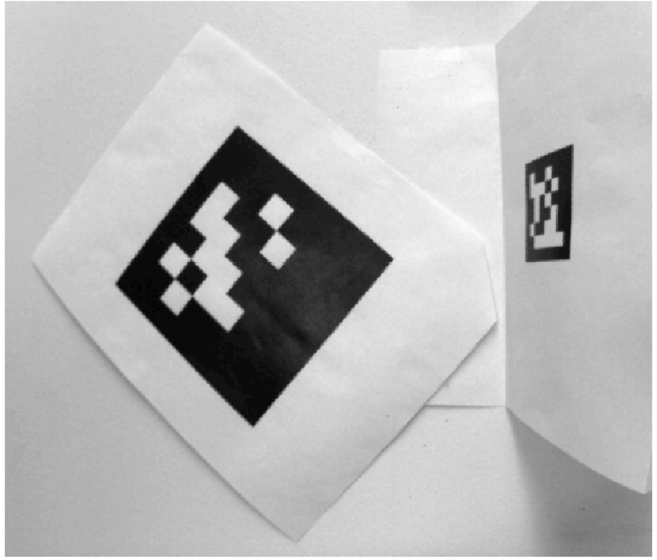
https://levelup.gitconnected.com/qr-code-scanner-in-kotlin-e15dd9bfbb1f

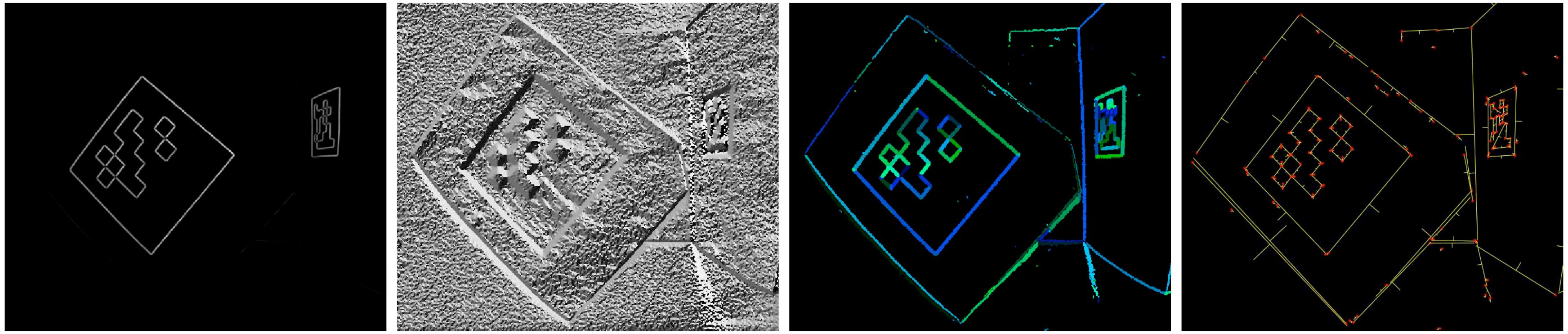
Visual Fiducials

- QR code is good at storing information
- AR tags store less information, but can be quickly and reliably detected



AprilTag



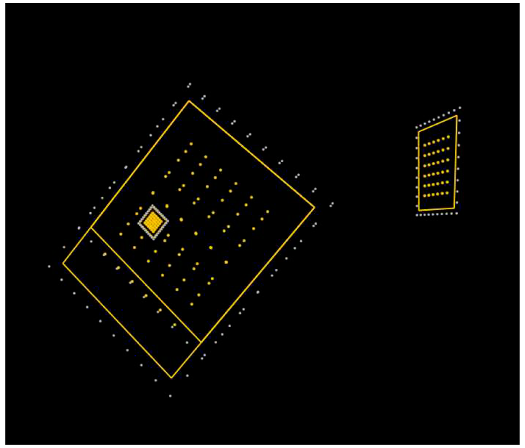


Gradient magnitudes

Gradient directions

Clustering

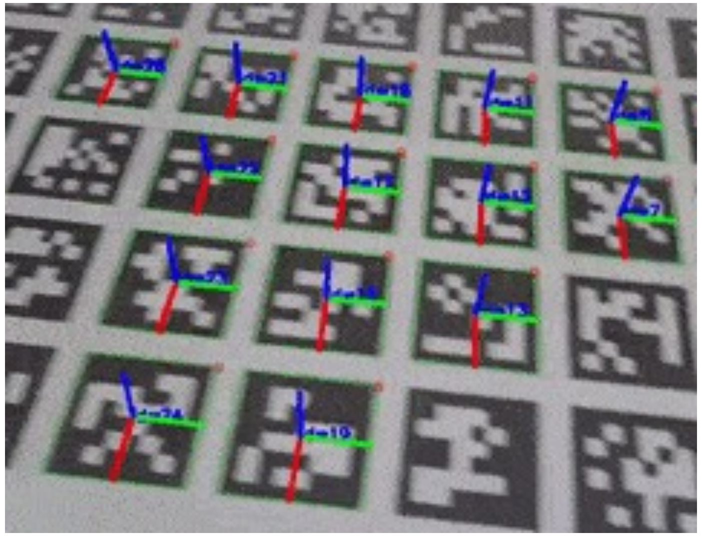
Line segments for cluster components

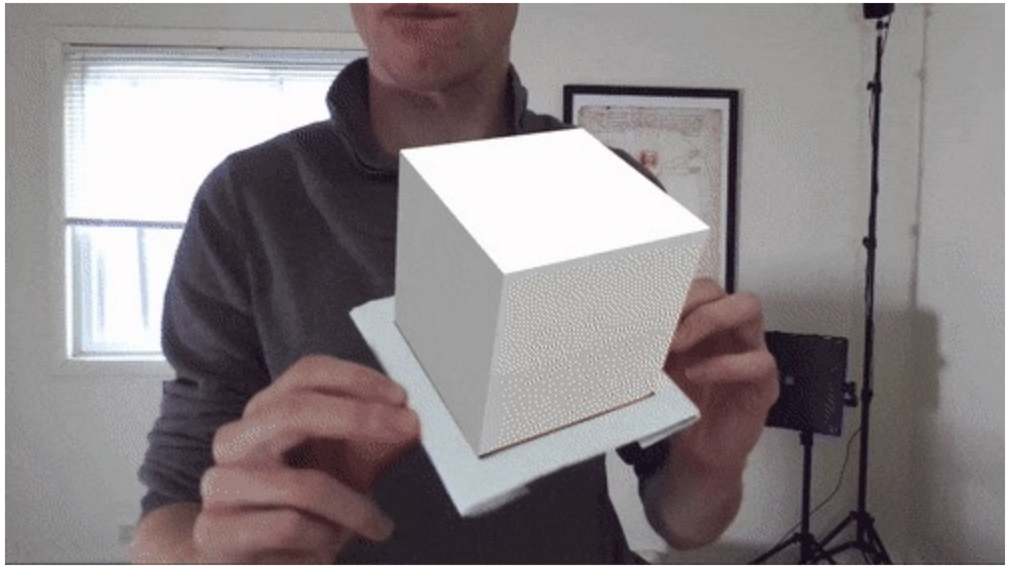


Quad detection

AprilTag: A robust and flexible visual fiducial system. Edwin Olson. ICRA, 2011

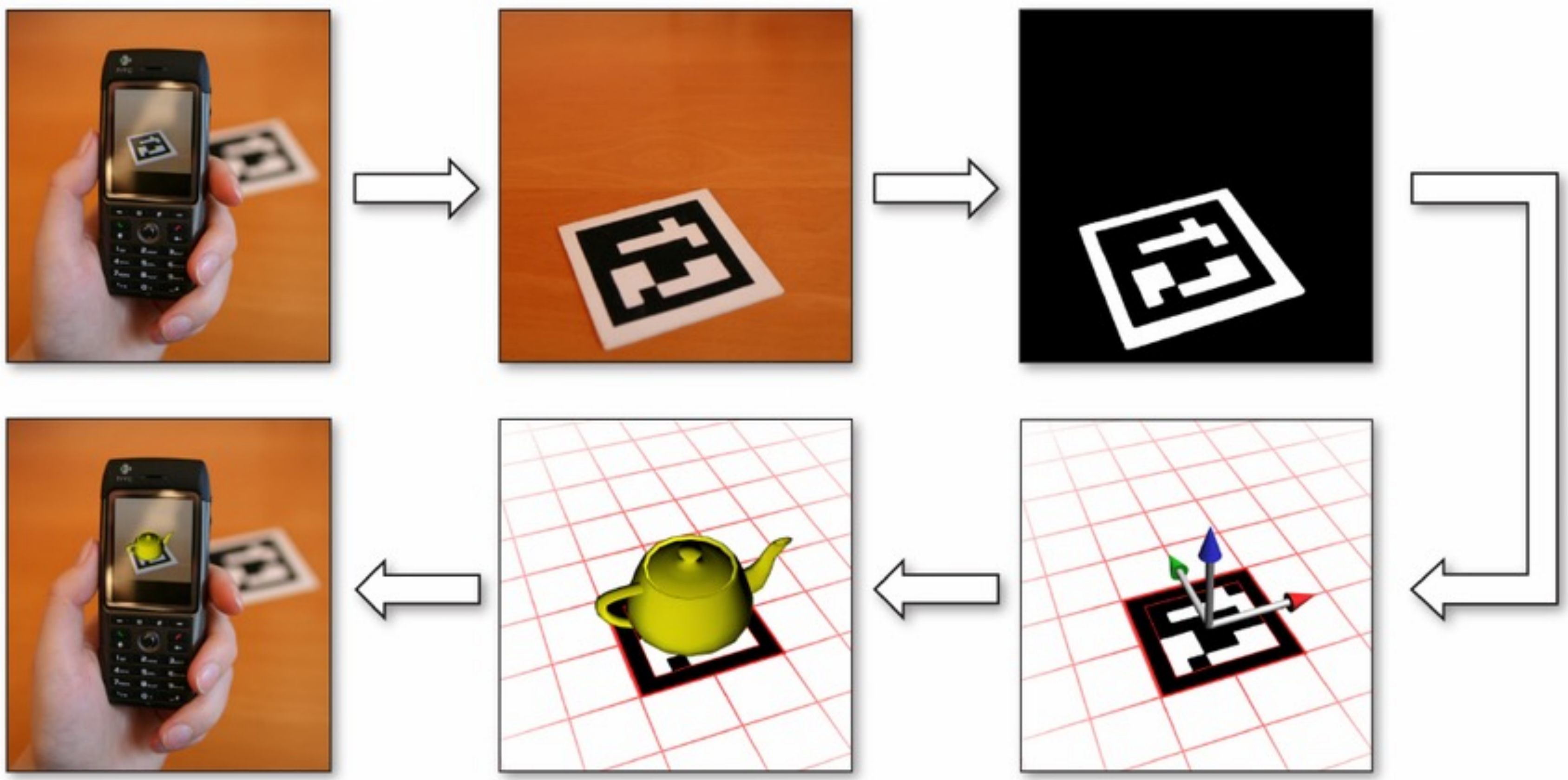
ArUco Examples





https://docs.opencv.org/4.5.2/d5/dae/tutorial_aruco_detection.html

Augmented Reality with AR Markers



https://www.researchgate.net/figure/Basic-workflow-of-an-AR-application-using-fiducial-marker-tracking_fig1_216813818

Cloaking with Infrared (IR)

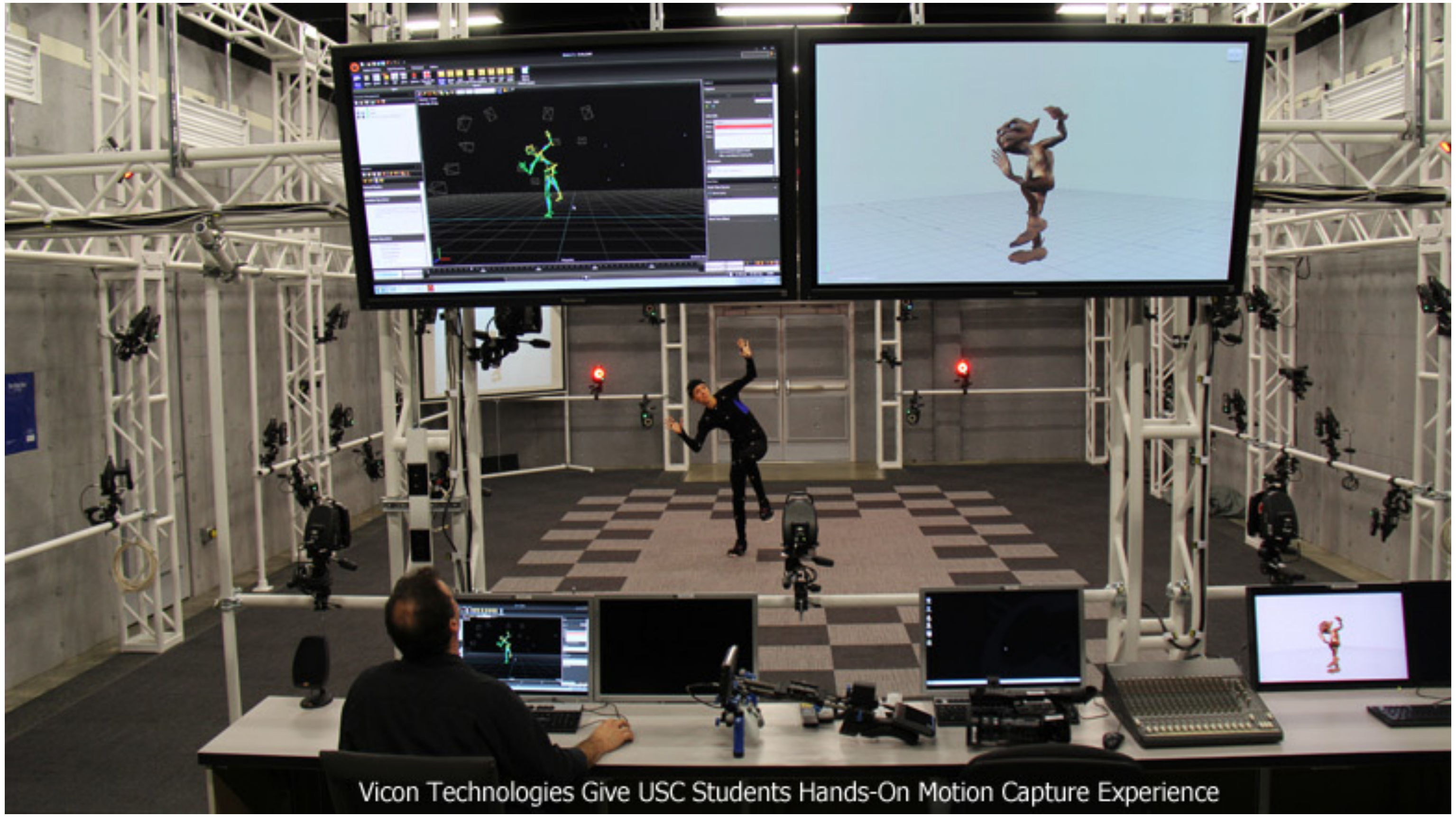
- AR markers introduce artificial features in the scene
- IR features are visible to cameras, but not to humans

Active features

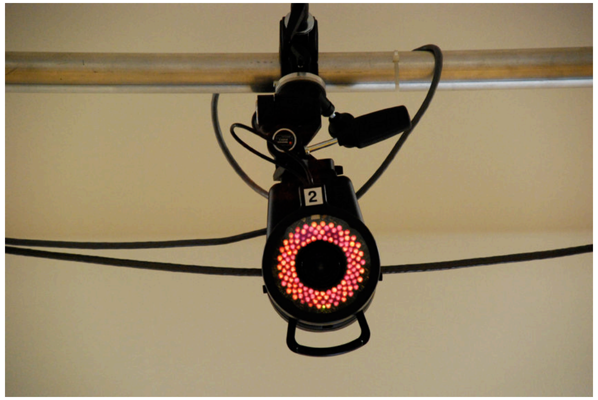
Need to have external cameras (old version)

The Oculus Rift headset contains IR LEDs hidden behind IR transparent plastic

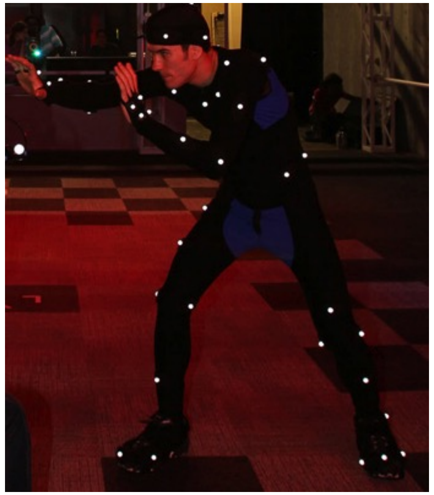
Motion Capture (MOCAP)



http://www.cgarena.com/newsworld/vicon-usc-motion-capture.php

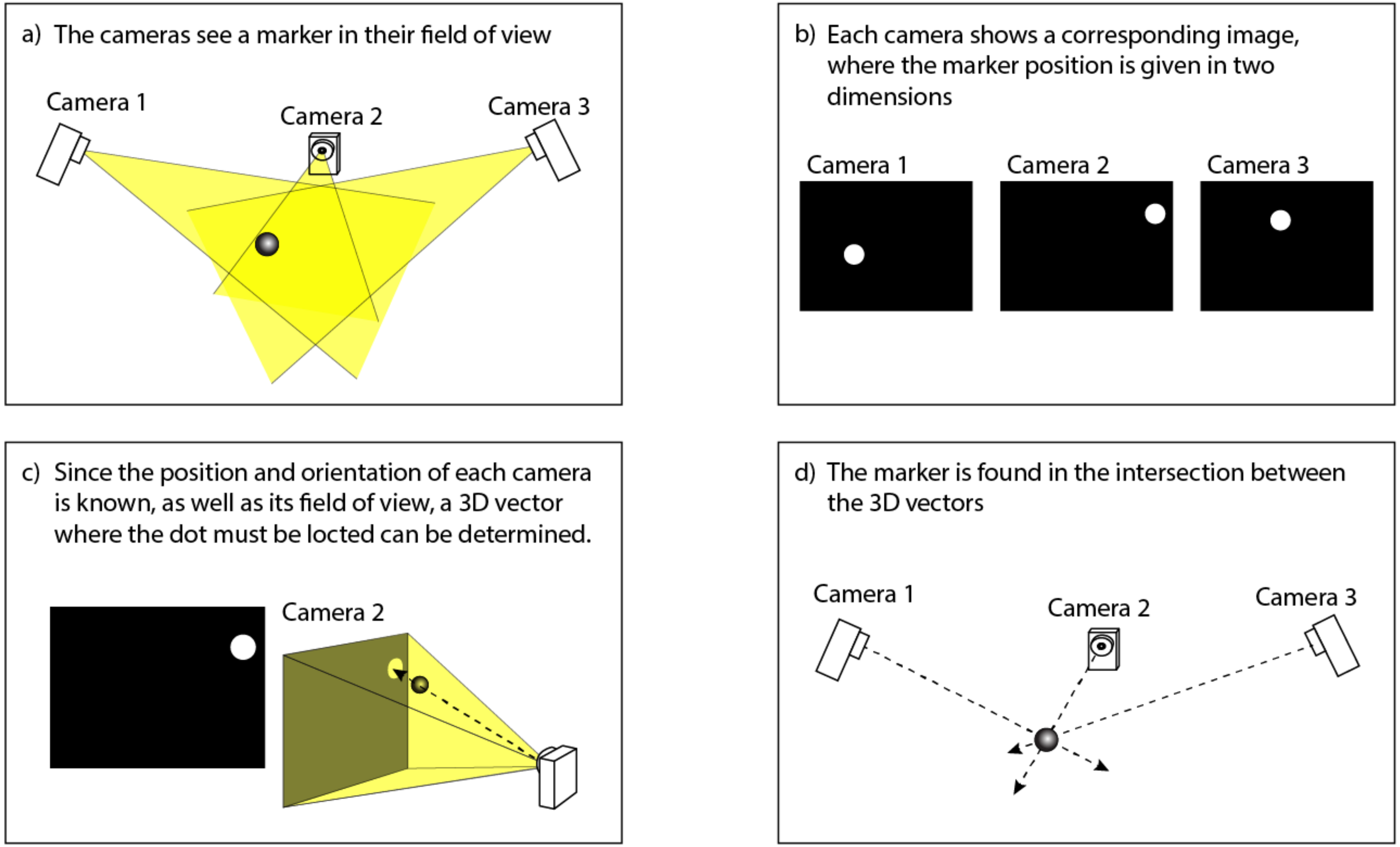


Infrared camera



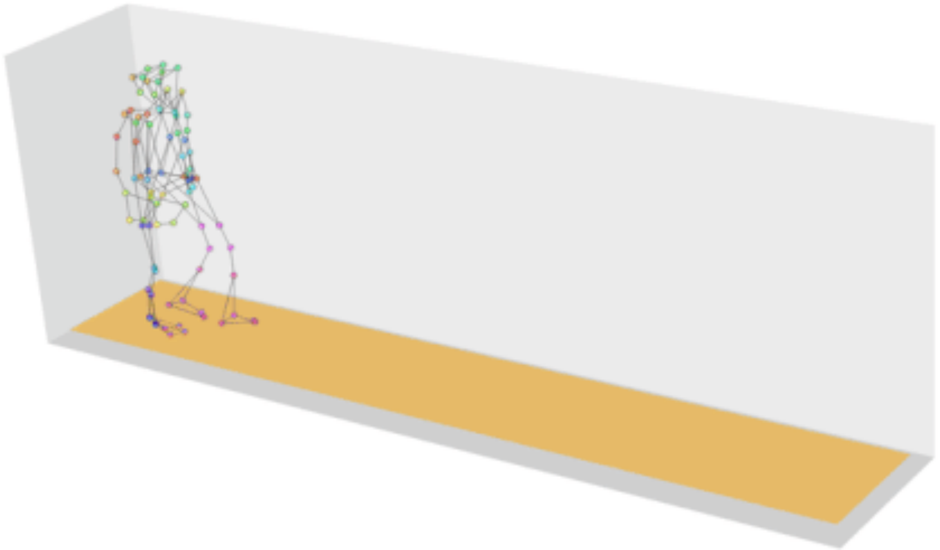
Retroreflective markers

Motion Capture (MOCAP)



https://www.futurelearn.com/info/courses/music-moves/0/steps/12692

MOCAP Examples



Wikipedia

Further Reading

- Sections 9.3, Virtual Reality, Steven LaValle

- EP*n*P: An Accurate O(n) Solution to the P*n*P Problem. Lepetit et al., IJCV'09.