EECE5698
Networked XR Systems

Some slides credits: Anthony Rowe, Intro to XR course, CMU

Last few Classes

- Intro to XR
- Internals
- Hardware aspects
- Software tools
- Networked systems

Lecture Outline for Today

- Discuss Homework1
- Sensors
- Sensing Algorithms

Sensors and Sensing Algorithms

- Popular Sensors
  - Color camera
  - Depth camera
  - Microphone
  - Inertial
  - Gyro
  - RF
- E.g., Functionality
  - Positioning
  - Tracking
  - 3D Scene Reconstruction

Positioning and Tracking

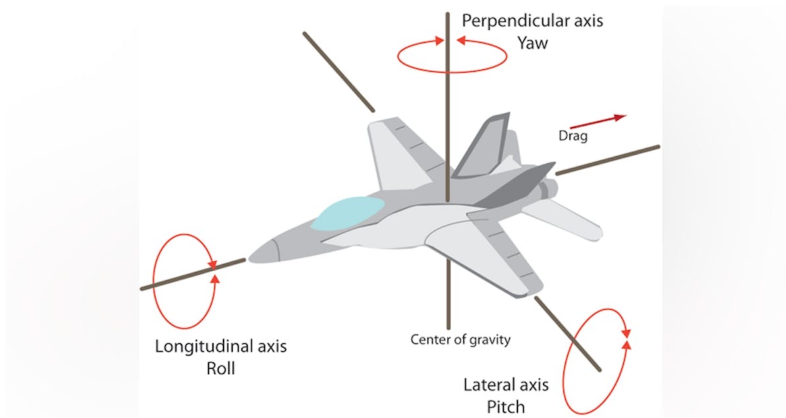
- What to position and track?
  - Users
    - Hands
    - Face
    - Eyes
    - Head
    - Body
    - Activity
    - Physiological signals
  - Environment
    - Objects

Positioning and Tracking

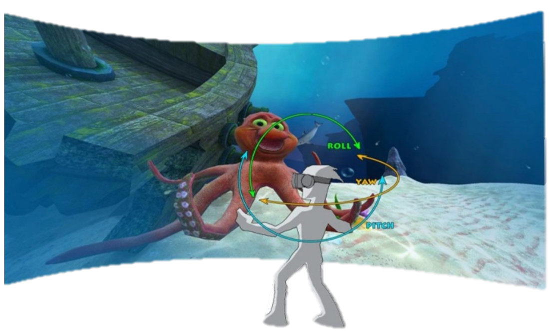
- Why do we need it?
  - For view port control
  - Place virtual content
  - Interact with virtual content
  - Occlusion
  - Adaptive rendering
  - Persistent anchors
  - And more…

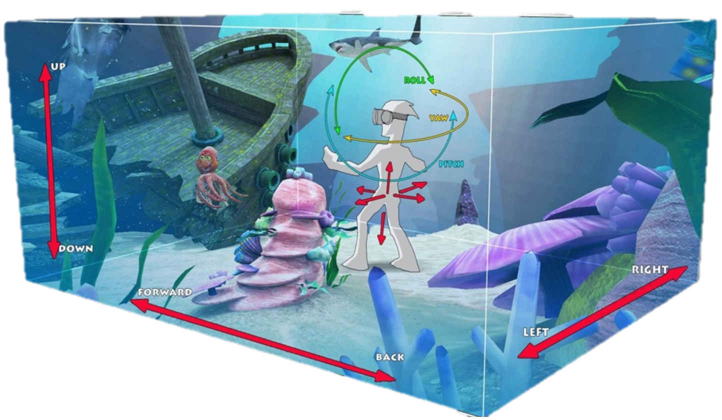
Positioning and Tracking

- You need to know where you are in the world
  - GPS?
  - Visual
  - Inertial
  - Lidar
  - RF



- 3-DoF
- 6-DoF

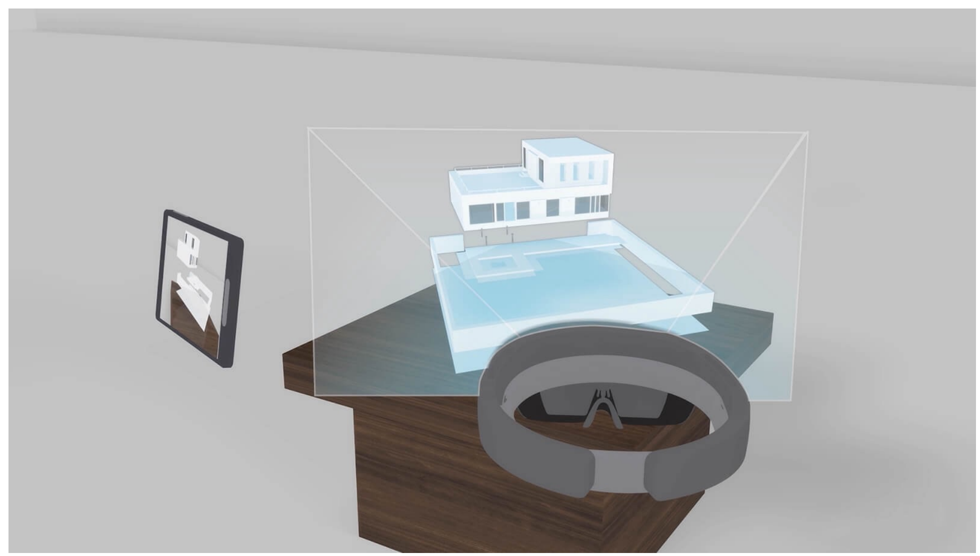




X, Y, Z & Yaw, Pitch, Roll

Positioning and Tracking

- Anchors
  - Anchors ensure that objects appear to stay at the same position and orientation in space, helping you maintain the illusion of virtual objects placed in the real world.



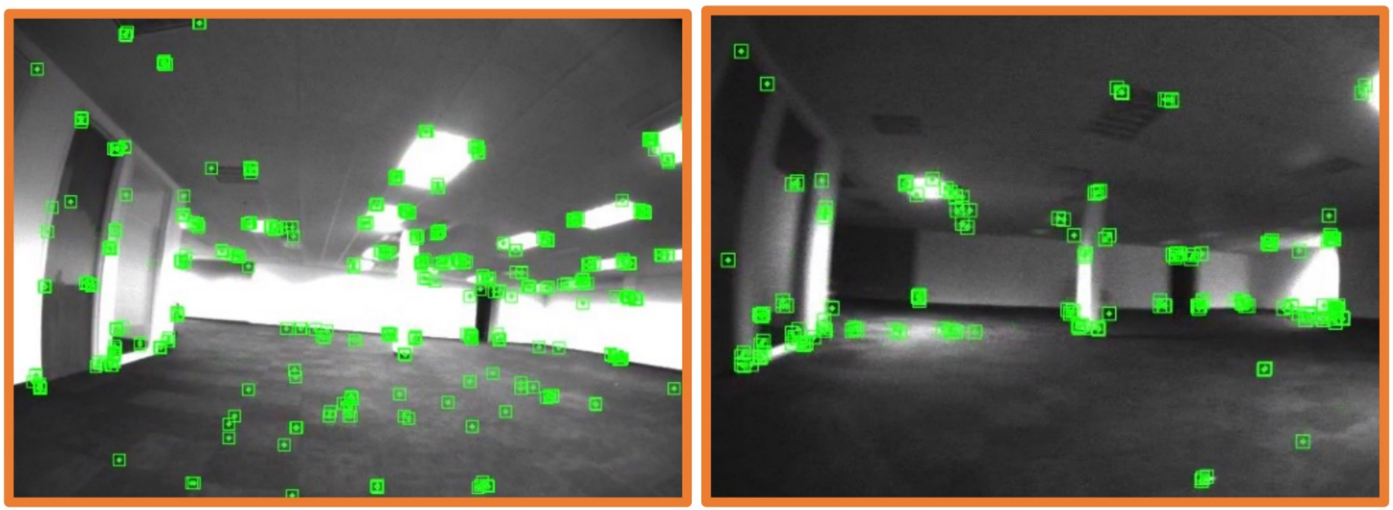
  - Plane
  - Wall
  - Floor
  - Face…
  - Anything that you can identify well

Positioning and Tracking

- Typical metrics of importance
  - Accuracy
  - Latency
  - Tracking drift
  - Tracking jitter
  - Update rate
  - Reliability

Visual Tracking Algorithm

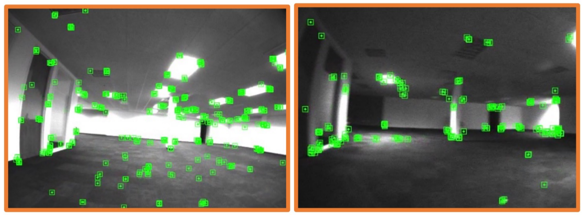
- Step1: Capture images
  - Mono or Stereo or multiple cameras
- Step2: Feature Extraction
  - Features are detected in the first frame, and then matched in the second frame.



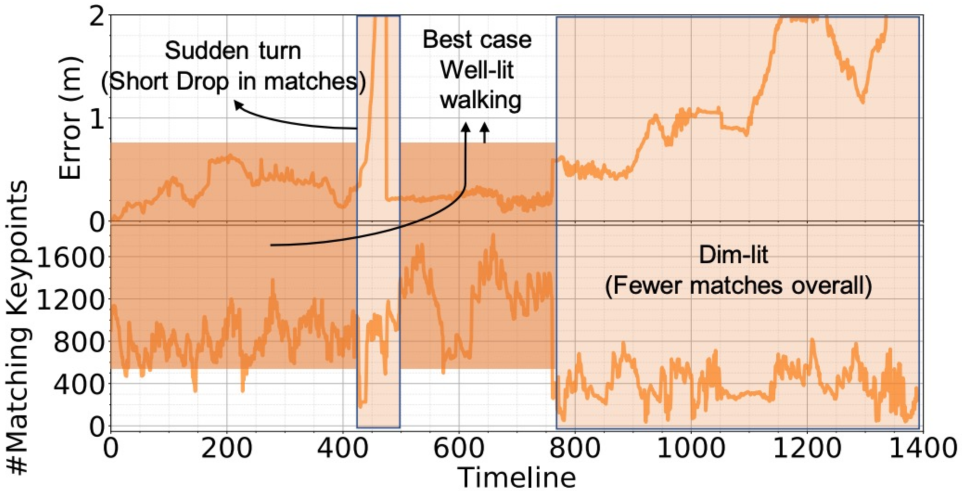
(a) Well-lit (568 matches) (b) Dim-lit (252 matches)

Visual Tracking Algorithm

- Step1: Capture images
  - Mono or Stereo or multiple cameras
- Step2: Feature Extraction
  - Features are detected in the first frame, and then matched in the second frame



(a) Well-lit (568 matches) (b) Dim-lit (252 matches)



(c) Matches vs. Error

Visual Tracking Algorithm

- Step3: Optical flow estimation

Get rid of outliers

Visual Tracking Algorithm

- Step4: Estimate camera motion from optical flow

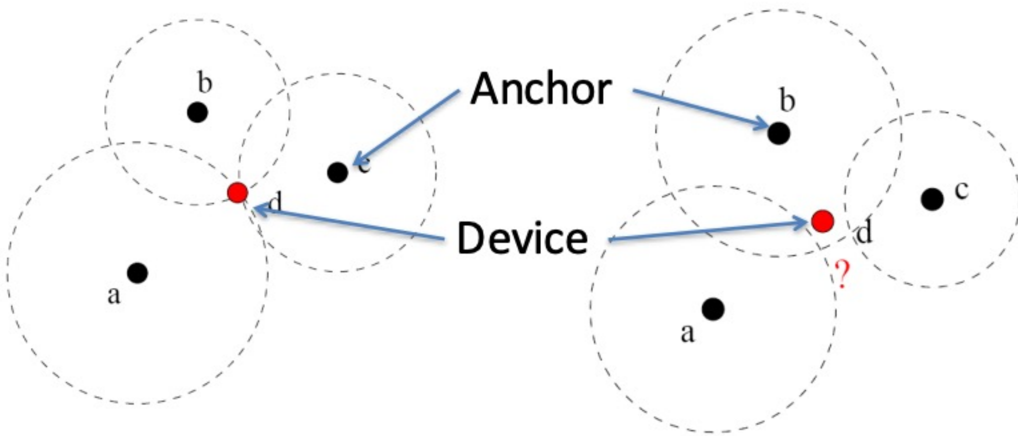
The optical flow field illustrates how features diverge from a single point, the *focus of expansion*. The focus of expansion can be detected from the optical flow field, indicating the direction of the motion of the camera, and thus providing an estimate of the camera motion.

Visual Tracking Algorithm

- Limitations:
    - Heavily depends on the environment
        - Lighting conditions
        - Geometry of the objects in the environment
        - Uniform surfaces or color
        - Moving objects
        - Fails when too close to objects; camera view occluded

RF-based Tracking

- Range based tracking
  - Convert received signal strength (RSS) or signal timing to a distance estimate with respect to anchor nodes with known locations.
  - Problem: distance estimates may be erroneous, and the circles may not intersect at a single point.



RF-based Tracking

How to estimate location when the circles do not intersect?

Idea: localize at a point that presents the minimum error to the circles by some reasonable error measure.

k anchors at positions (x_i, y_i)

Assume node to be localized has actual location at (x_0, y_0)

Distance estimate between node 0 and anchor *i* is r_i

Error:

$$f_i = r_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}$$

RF-based Tracking

Linearization and Min Mean Square Estimate

- Ideally, we would like the error to be 0

$$f_i = r_i - \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} = 0$$

- Re-arrange:

$$(x_0^{\ 2} + y_0^{\ 2}) + x_0(-2x_i) + y_0(-2y_i) - r_i^{\ 2} = -x_i^2 - y_i^2$$

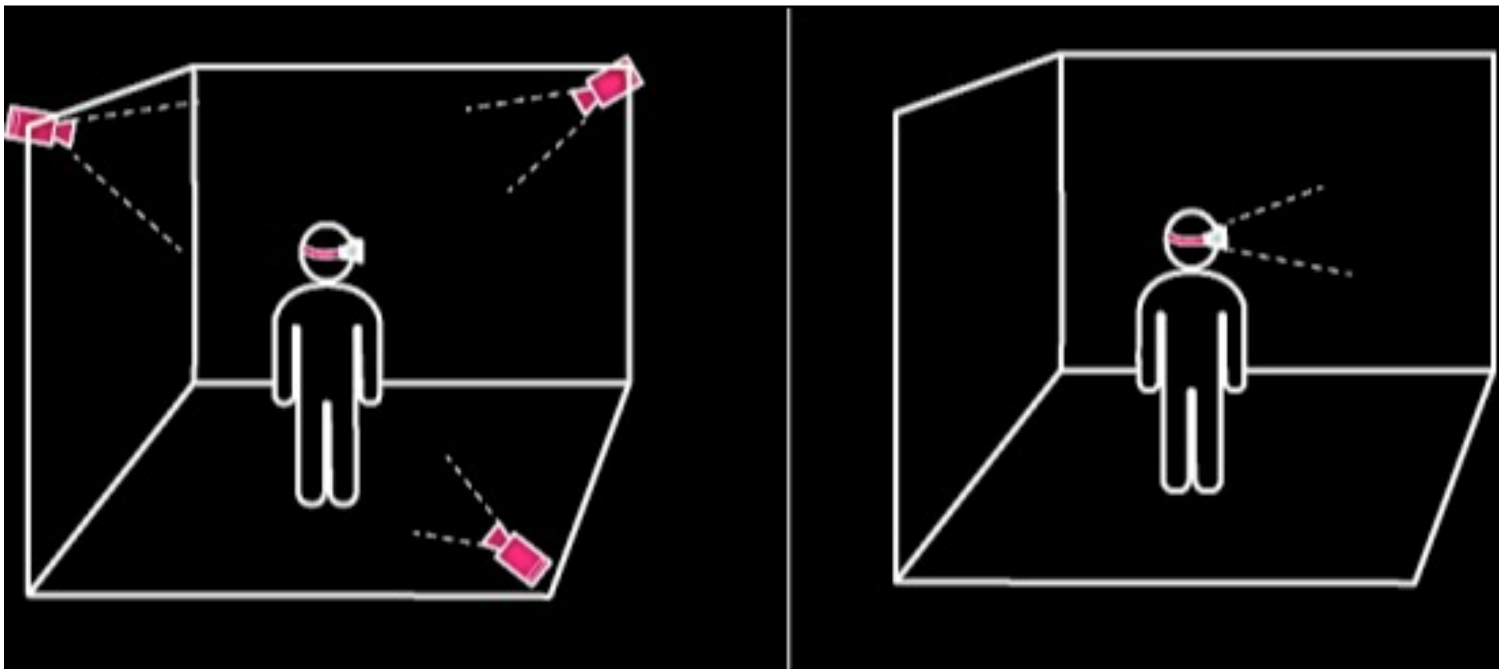
- Subtract the last equation from the previous ones to get rid of quadratic terms.

$$2x_0(x_k - x_i) + 2y_0(y_k - y_i) = r_i^2 - r_k^{\ 2} - x_i^2 - y_i^2 + x_k^2 + y_k^2$$

- Note that this is linear.

https://en.wikipedia.org/wiki/Ordinary_least_squares

Outside in and Inside out Tracking



Outside in Inside out

Inertial sensing

- Accelerometer & Gyroscope
  - Measuring linear acceleration (accelerometer) and / or angular orientation rates (gyroscope)
  - No transmitter, cheap, small, high frequency, wireless

https://youtu.be/-0hSQFbt67U?t=24

Inertial sensing

1. Acceleration Measurement

The IMU's accelerometer measures linear acceleration in three axes (a_x, a_y, a_z) relative to its local frame of reference. This is the starting point for position tracking.

Equation for acceleration:

$$\mathbf{a}(t) = \begin{pmatrix} a_x(t) \\ a_y(t) \\ a_z(t) \end{pmatrix}$$

However, the accelerometer measures the sum of the actual acceleration and gravitational acceleration ($\mathbf{g}$). So to get the actual acceleration, gravity needs to be removed using orientation data from the IMU’s gyroscope.

Inertial sensing

2. Removing Gravity

The acceleration measured by the IMU includes both the device's acceleration and the gravitational force. The orientation of the IMU, determined by the gyroscope and potentially a magnetometer, is used to rotate the measured acceleration to the global reference frame and subtract gravity.

You can rotate the accelerometer data using the orientation (quaternion or rotation matrix) to align it with the global frame and then subtract gravity:

$$\mathbf{a}_{global} = \mathbf{R} \cdot \mathbf{a}(t) - \mathbf{g}$$

Where:

- $\mathbf{R}$ is the rotation matrix obtained from the gyroscope data.
- $\mathbf{g}$ is the gravity vector, typically $(0, 0, 9.81)$ m/s² in the global reference frame.

Inertial sensing

3. Velocity Estimation

Once you have the correct acceleration in the global frame, you can integrate this acceleration to estimate velocity.

Equation for velocity:

$$\mathbf{v}(t) = \mathbf{v}(t_0) + \int_{t_0}^{t} \mathbf{a}_{global}(\tau)\, d\tau$$

Where:

- $\mathbf{v}(t)$ is the velocity at time t,
- $\mathbf{v}(t_0)$ is the initial velocity (which is often assumed to be zero),
- $\mathbf{a}_{global}(\tau)$ is the acceleration in the global frame over time τ.

Inertial sensing

4. Position Estimation

Finally, the velocity is integrated to get the position over time:

Equation for position:

$$\mathbf{p}(t) = \mathbf{p}(t_0) + \int_{t_0}^{t} \mathbf{v}(\tau)\, d\tau$$

Where:

- $\mathbf{p}(t)$ is the position at time t,
- $\mathbf{p}(t_0)$ is the initial position (which may be assumed to be known),
- $\mathbf{v}(\tau)$ is the velocity over time τ.

IMU Position Tracking Example Problem

An AR/VR headset is equipped with an IMU that provides the following data:

- The headset starts at rest at position $(0, 0, 0)$ and time $t = 0$.
- At $t = 1$ second, the accelerometer readings are $a_x = 2\,\mathrm{m/s}^2$, $a_y = 0\,\mathrm{m/s}^2$, and $a_z = 0\,\mathrm{m/s}^2$.
- The gyroscope indicates that the headset is not rotating (so no need to remove gravity in this case).
- Assume that gravity does not affect the motion since the headset is moving horizontally.

Question:

1. What is the headset's velocity after 1 second?
2. What is the headset's position after 1 second?
3. What happens if the acceleration remains constant for the next 2 seconds (total time = 3 seconds)? Calculate the position at $t = 3$ seconds.

IMU Position Tracking Example Problem

Given:

- Initial velocity $\mathbf{v}(0) = (0, 0, 0)\,\mathrm{m/s}$
- Acceleration $\mathbf{a} = (2, 0, 0)\,\mathrm{m/s^2}$

Velocity is calculated by integrating the acceleration over time:

$$\mathbf{v}(t) = \mathbf{v}(0) + \int_0^1 \mathbf{a}(t)\,dt$$

Since the acceleration is constant, the velocity after 1 second is:

$$\mathbf{v}(1) = \mathbf{v}(0) + \mathbf{a} \cdot t = (0, 0, 0) + (2, 0, 0) \cdot 1 = (2, 0, 0)\,\mathrm{m/s}$$

Answer:

The velocity at $t = 1$ second is $(2, 0, 0)\,\mathrm{m/s}$.

IMU Position Tracking Example Problem

Now, use the velocity to calculate the position. The position is calculated by integrating the velocity over time:

$$\mathbf{p}(t) = \mathbf{p}(0) + \int_0^1 \mathbf{v}(t)\, dt$$

Since the velocity is increasing linearly from 0 to 2 m/s, the average velocity over the first second is:

$$\mathbf{v}_{\text{avg}} = \frac{\mathbf{v}(0) + \mathbf{v}(1)}{2} = \frac{(0,0,0) + (2,0,0)}{2} = (1,0,0)\,\mathrm{m/s}$$

Now, calculate the position:

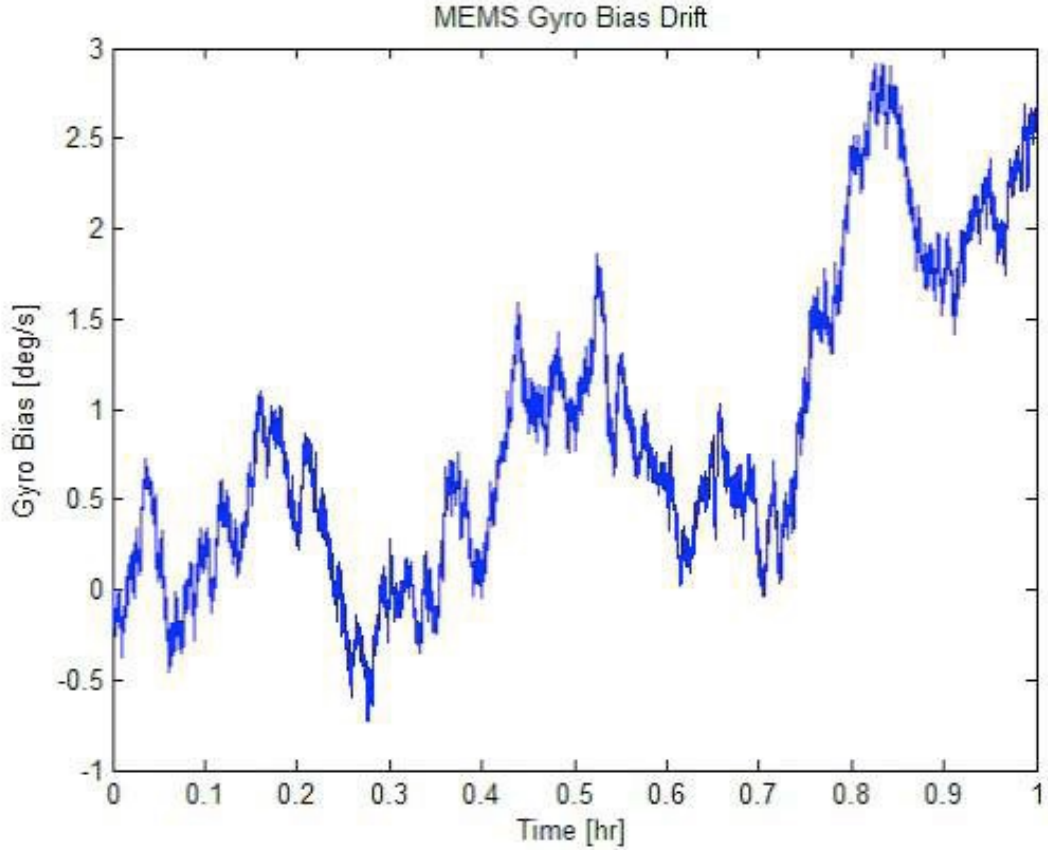
$$\mathbf{p}(1) = \mathbf{p}(0) + \mathbf{v}_{\text{avg}} \cdot t = (0,0,0) + (1,0,0) \cdot 1 = (1,0,0)\,\mathrm{m}$$

Answer:

The position at $t = 1$ second is $(1, 0, 0)\,\mathrm{m}$.

Inertial sensing

- Drift

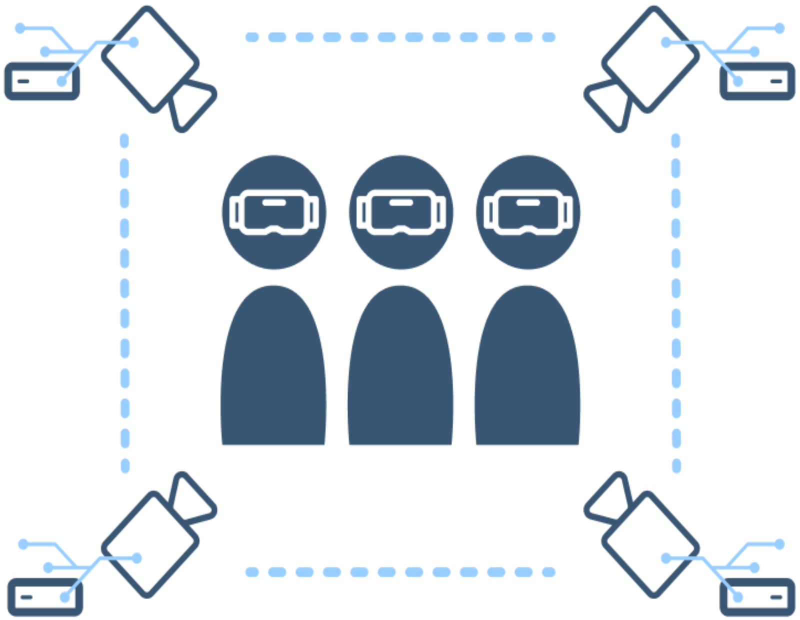


3D Reconstruction Algorithm

- Camera Calibration
- Depth Sensing
- Surface Extraction
- Texture Generation

3D Reconstruction

- Camera Calibration
  - Multiple cameras
  - Distortion
  - Intrinsic and extrinsic parameters are different for different cameras



3D Reconstruction Algorithm

- Camera Calibration

- **Input**: set of pictures
- **Output**: camera position, orientation, intrinsic parameters (focal length, optical center)

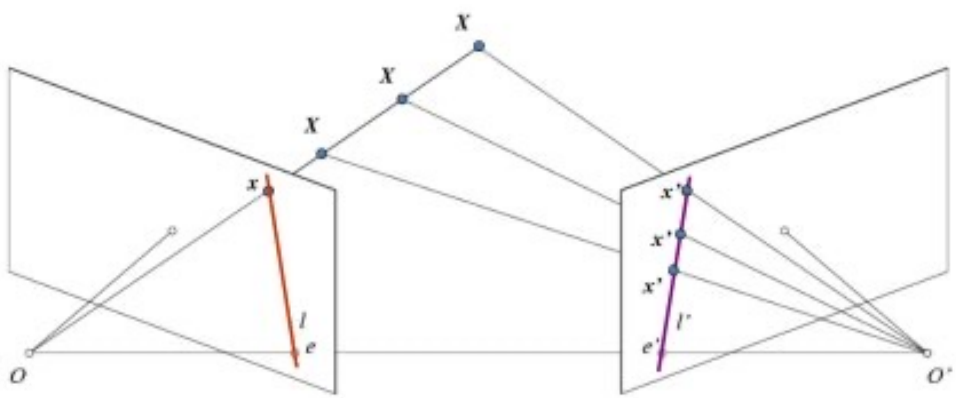
3D Reconstruction Algorithm

- Depth Sensing
  - Input: set of calibrated images
  - Output: distance to object for each pixel in the image

- Popular methods
  - Stereo triangulation
  - Time of flight
  - Structured light projection

3D Reconstruction Algorithm

- Depth Sensing

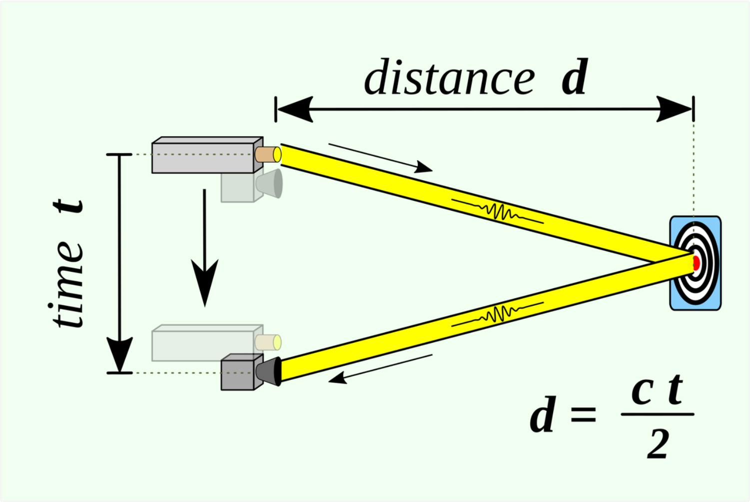


Stereo Triangulation

Zed Camera

3D Reconstruction Algorithm

- Depth Sensing

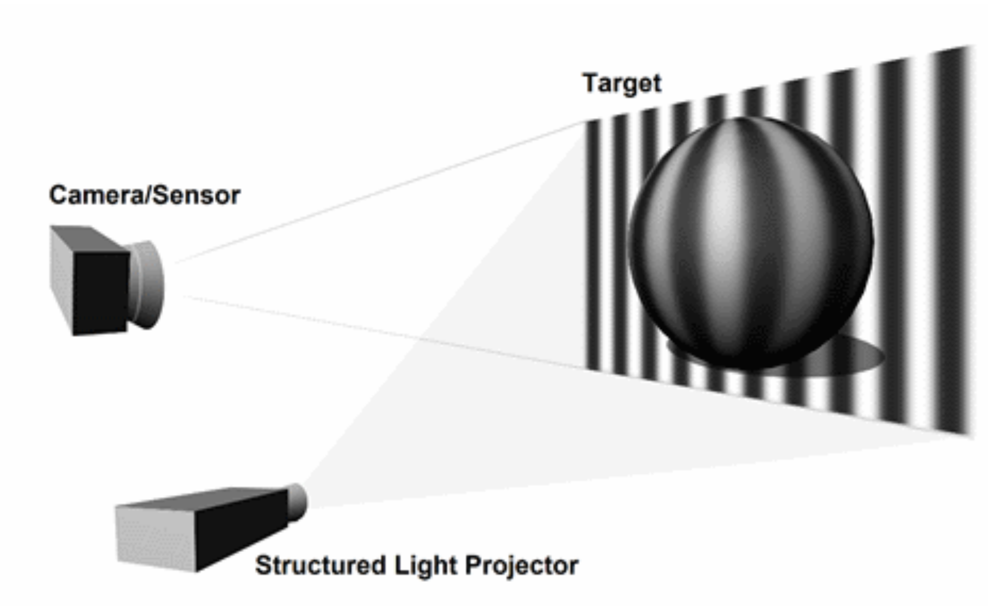


Time of flight

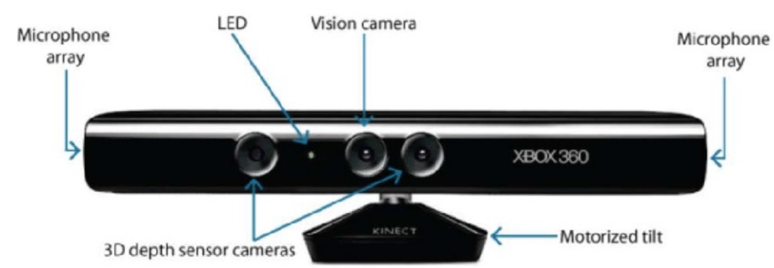
Helios

3D Reconstruction Algorithm

- Depth Sensing



Structured light projection



Azure Kinectv1

3D Reconstruction Algorithm

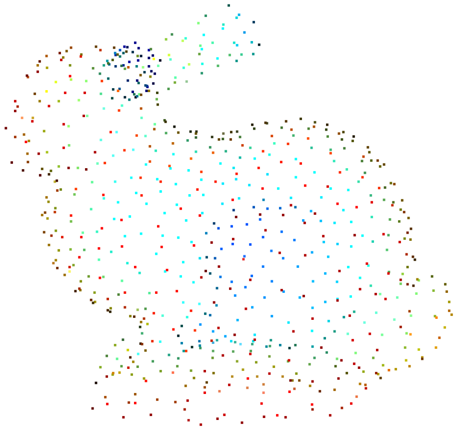
- Depth Sensing

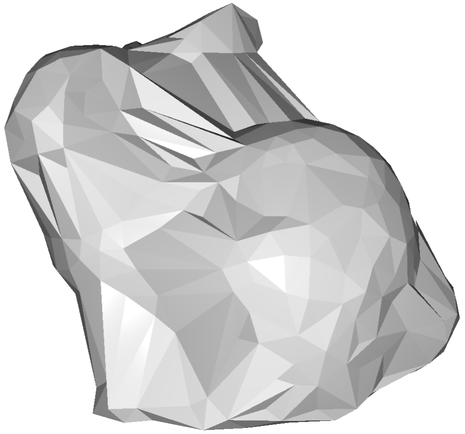
	Stereo vision	Structured light	Laser triangulation	Time of Flight
Distance & range	Medium to far (depending on the distance of the 2 cameras) & limited 2m to 5m	Short to medium & scalable cm to 2m	Short & Limited cms	Far & scalable 30-50cm to 20-50m
Resolution	Medium	Medium	Varies	High
Depth accuracy	Medium	Medium to very high in short range	Very high	Medium
Software complexity	High	Medium	High	Low
Real-time capability	Low	Low	Low	High
Low light behaviour	Weak	Good	Good	Good
Outdoor light	Good	Weak	Weak	Weak to good
Compactness	Medium	Medium	Medium	Very compact
Material costs	Low	High	High	Medium
Total operating cost (including calibration efforts)	High	Medium to high	High	Medium

https://www.azom.com/article.aspx?ArticleID=16003

3D Reconstruction Algorithm

- Surface Extraction from Depth
    - Input: set of calibrated images & depth maps
    - Output: mesh of object





3D Reconstruction

- Texture Generation
  - Input: set of calibrated images and mesh of object
  - Output: atlas and texture

Summary of the Lecture

- Discuss Homework1
- XR Internals
- Sensors
- Sensing Algorithms

Next Up: XR Data Structures