Announcements

- **MP4**, Nov 18 @11am

**Final project upcoming deadlines:**

- Nov 30, short 2-3min video of your project
- Dec 12, 7:00-10:00pm, Final project presentation

**Second Midterm:** Dec 7, 7:00-8:30pm

**Grades are out for:**

- MP2
- MP3
Tracking Systems in VR

Tracking: estimation of motions in physical world that bring changes to the VR world

What do we want to track? Think about rigid bodies:

1. Head wearing HMD (orientation + position)
2. Eyes
3. Palms of hands
4. Fingers
5. Entire body
6. Movable objects - controller, coffee cup, desk
7. Other people in the space

Tracking is crucial to minimizing latency.
Tracking Systems in VR

What do we want to track: Rigid bodies

For each rigid body, estimate

Rotation: \(3 \times 3\) rotation matrix, \(R_t\), or \(Q_t = (a, b, c, d)\)

Position: \((x_t, y_t, z_t)\)

OR

Homogeneous transformation matrix:

\[
H_t = \begin{bmatrix}
R_t & x_t \\
y_t & y_t & z_t \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Task: Estimate \(H_t\) or \(Q_t/R_t\) over time
Axis-angle: \(\text{ora}_i: (v_i, \Delta \Theta_i)\)  
3-axis gyroscope measures:  
\[\vec{\omega}_i = (\omega_x, \omega_y, \omega_z)\]

For every \(\Delta t\) suppose \(\vec{\omega}_i\) is constant, then rotation over \(\Delta t\) is by

angle: \(\Delta \Theta_i = \|\vec{\omega}_i\| \cdot \Delta t\)

axis: \(\vec{v}_i = \frac{\vec{\omega}_i}{\|\vec{\omega}_i\|}\)

\(\vec{\omega}_i\) is NEVER accurate!  
\(\vec{\omega}_i \rightarrow \hat{\vec{\omega}}_i\)

How large is \(\Delta Q_i = \)
Tracking Systems in VR: Estimating 3D Orientation

Integrate sensor readings to estimate orientation:

**Recall 2D:** \( \hat{\Theta}_k = \theta_0 + \sum_{i=1}^{k-1} \Delta \hat{\Theta}_i = \theta_0 + \Delta \hat{\Theta}_1 + \Delta \hat{\Theta}_2 + \ldots + \Delta \hat{\Theta}_{k-2} + \Delta \hat{\Theta}_{k-1} \)

**3D:** \( \hat{Q}_k = Q_0 \circ \Delta \hat{Q}_1 \circ \Delta \hat{Q}_2 \circ \ldots \circ \Delta \hat{Q}_{k-2} \circ \Delta \hat{Q}_{k-1} \)

Recursively/Incrementally:

**2D:** \( \hat{\Theta}_{\text{current}} = \hat{\Theta}_{\text{prev}} + \Delta \hat{\Theta}_{k-1} \)

**3D:** \( \hat{Q}_{\text{current}} = \hat{Q}_{\text{prev}} \circ \Delta \hat{Q}_{k-1} \)

Problem: **Drift** (or dead reckoning)
Estimating 3D Orientation: Drift Correction

Expression for drift error: divergence of the current estimate from the true value

2D: \( d_k = \theta_k - \hat{\theta}_k \)

3D: \( d_k = \uparrow \text{true} \uparrow \text{estimate} \)

This error needs to be minimized to minimize latency!

An unusual quantity to have for an error!
Correcting for drift errors challenge:

1) Use another sensor(s) to provide global reference

2) Gradually apply corrections
   - Fast enough to **eliminate** error
   - Slow enough to **avoid** VR sickness
Estimating 3D Orientation: Drift Correction

Separate the rotational drift error into two components:

1) **Tilt error**: pitch + roll
   
   To correct: need a gravity or "up" sensor

2) **Yaw error**
   
   To correct: need a "compass"

- Complementary Filters on SO(3), Mahoney, 2008

- Head Tracking for the Oculus Rift, ICRA 2014
  S. LaValle, A. Yershova, M. Katsev, and M. Antonov
Use Accelerometer to Correct "Tilt Error"

What is \( \vec{a} \) in world frame? \( \vec{a} = \)

Assume \( Q \), \( \vec{a} \) accurate. What is \( \vec{a} \) in terms of \( Q \) and \( \vec{u} \)?

What if \( \vec{a} \) is not aligned with \( y \)-axis?
Use Accelerometer to Correct "Tilt Error"

Apply $\hat{\mathbf{Q}}$ to accelerometer reading, $\hat{\mathbf{a}}$

Obtain: $\hat{\mathbf{a}} = (\hat{a}_x, \hat{a}_y, \hat{a}_z)$

Estimated world frame

Tilt axis in XZ plane:

$\vec{V}_{\text{tilt}} = \ldots$

Now rotate by $\phi$ about $\vec{V}_{\text{tilt}}$ to fix $\hat{\mathbf{Q}}$

$\hat{\mathbf{Q}}_{\text{corrected}} = \ldots$
Use Accelerometer to Correct "Tilt Error"

Complementary filter:

In each $dt$,
- rotate by $\lambda \xi$
- about $\mathbf{v}_{\text{tilt}}$ to fix $\hat{Q}$

Gain coefficient $\lambda > 0$, $\lambda \ll 0$ (ex. $\lambda = 0.0001$)

$Q_{\text{corrected}} = \text{Quat}(\mathbf{v}_{\text{tilt}}, \lambda \xi) \cdot \hat{Q}$

$\lambda$ needs to be large enough to correct tilt error
but small enough to be comfortable and without jitter.
What Does Accelerometer Measure?

-or in vacuum or on the Moon

Rift is free falling: on Earth

\[ \vec{a} = (0, 0, 0) \]

Raw reading

Rift lying on a side: on a table

\[ (0, g, 0) \]

\[ (-g, 0, 0) \]

\[ (-g \sin \theta, 0, 0) \]

\[ (-g \sin \theta, 0, 0) \]

\[ (g \cos \theta, 0, 0) \]

\[ (9.81, 0, 0) \]
Use Accelerometer to Correct "Tilt Error"

Problem:
Accelerometer measures vector sum of gravity and linear acceleration of sensor.

Solution:
Use heuristic to detect when "not moving" and apply correction only then.

Example: $||\mathbf{a}|| \approx 9.81 \Rightarrow \frac{||\mathbf{w}||}{||\mathbf{a}||} \Rightarrow \text{over some } \Delta t$
Use Magnetometer to Correct Yaw Error

Similar to tilt correction:

- Calculate reference error
- Gradually apply using complementary filter

Problems:

- Vector sum of Earth magnetic field + building magnetic field + sensor board m.f. difficult
- Calibration is difficult
- Field might vary over time & space

accuracy ≈ 5 degrees
Estimating Position and Orientation

Problem setup:
- Allow and track parallax motions
- IMU (gyro + accelerometer + magnetometer) not enough
  - Drift too fast from double integration
  - No way to detect drift errors
- Need sub-millimeter accuracy, stable estimates

Solutions:
1. Generate non-constant magnetic or EM field
   - RazerHydra, STEM Sixense
   - UWBradio
2. Visibility or line of sight methods
Position Tracking: Visibility Methods

Camera arrangements:

On headset:
- inside-out

In world:
- outside-in
Position Tracking: Visibility Methods

Pinhole camera:

perspective projection

model

image plane

Features in an image:
Position Tracking: Visibility Methods

FEATURES:

1) **Natural**
   - Hard computer vision
   - Extract and maintain from natural scenes
   - Remove moving objects
   - Reliability low

2) **Artificial**
   - Trivial computer vision (blob detection)
   - QR tags, retro reflective markers, LEDs, laser projections
   - Can stay in IR spectrum (invisible to humans)
Position Tracking: Blob Detection

PnP Problem:
Determine rigid body transformation from identified, observed features on a rigid body.

P1P Problem:

DOF analysis:
- Start with 6 DOFs (rigid body)
- Each feature subtracts 2 DOFs

4 DOFs left: 3D
Position Tracking: Blob Detection

P2P Problem:

Determine position and orientation of triangle from features in image.

DOFs left:
Position Tracking: Blob Detection

P3P Problem:

Determine position and orientation of triangle from features in image.

DOFs left:

Solution: A system of polynomial equations leads to 8 solutions (but only 4 in front of the camera). The beginnings of computational real algebraic geometry.
Position Tracking: Incremental Blob Detection

Incremental PnP Problem:

Determine position and orientation of triangle from features in image, given current estimate.