Quaternions & IMU Sensor Fusion with Complementary Filtering

Gordon Wetzstein

Stanford University

EE 267 Virtual Reality

Lecture 10

stanford.edu/class/ee267/

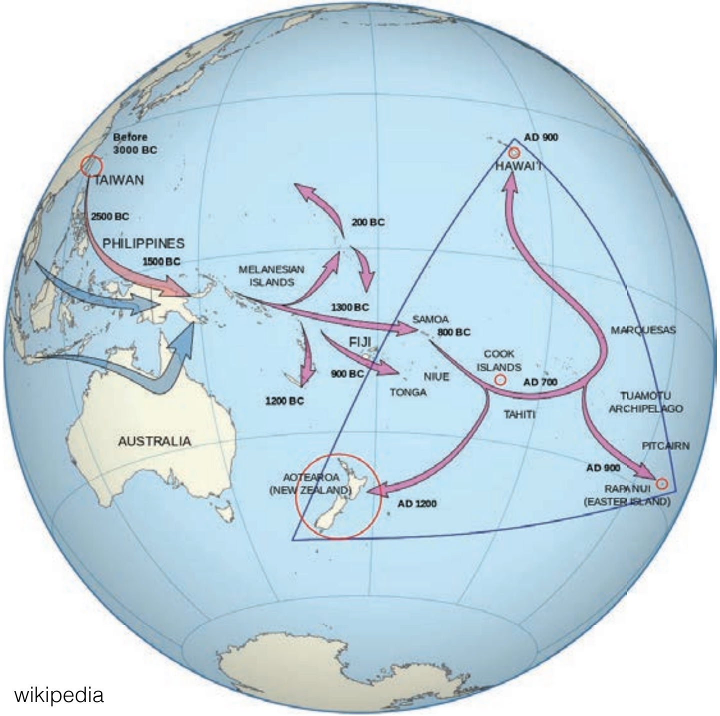
April 27, 2016

Updates

- project proposals due: May 6, 2016 (next Friday)
    - 1 page
    - at least 3 scientific references (no websites)
    - brief intro & motivation
    - milestones and timeline
    - clearly list expected outcomes of project

- project proposal pitch: May 9 (the Mon after)
    - 1 slide per team presented in 1 minute!

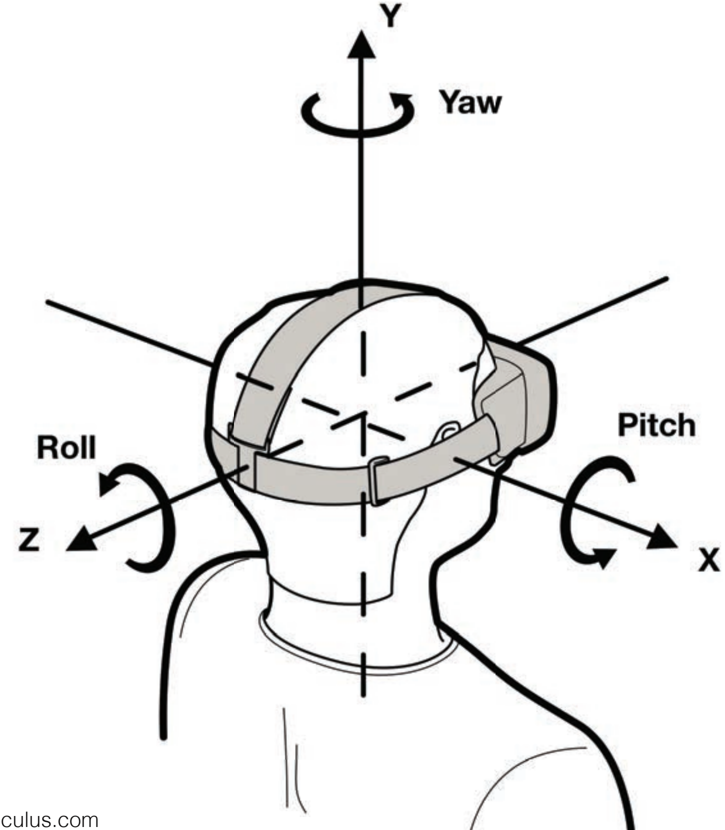
Polynesian Migration



wikipedia

Lecture Overview

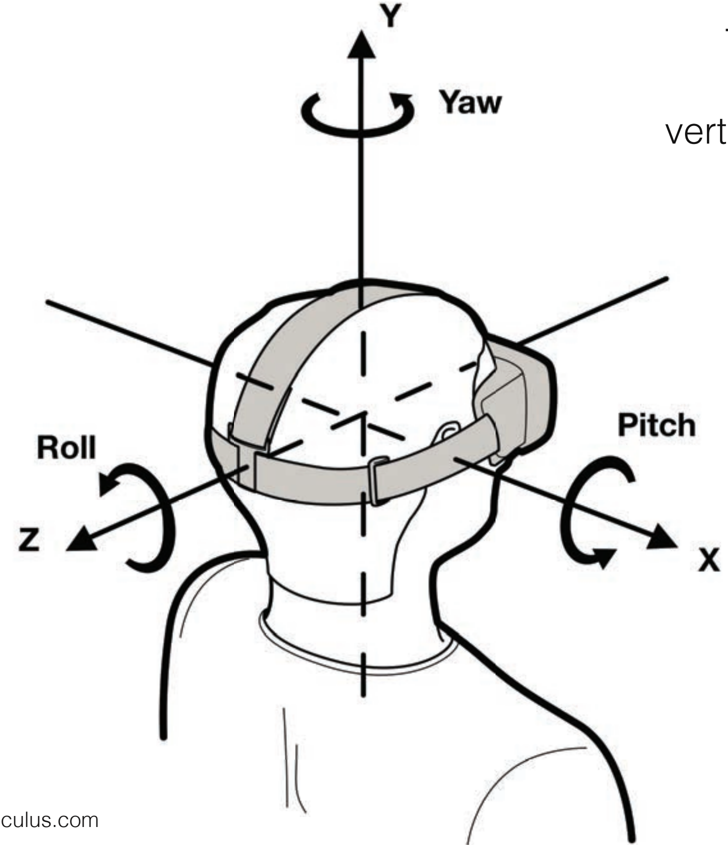
- short review of coordinate sytems
- gyro-based head orientation tracking
- tilt correction with complementary filtering
- rotations: Euler angles, axis & angle, gimbal lock
- rotations with quaternions
- 9 DOF IMU sensor fusion with complementary filtering



oculus.com

- primary goal: track head orientation

- inertial sensors required to determine pitch, yaw, and roll



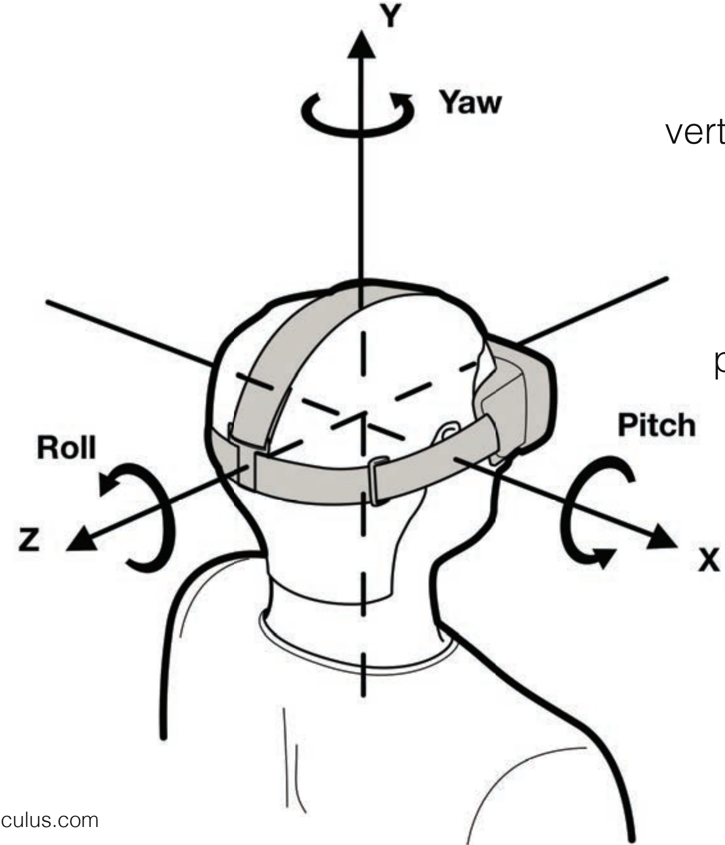
oculus.com

from lecture 2:

vertex in clip space

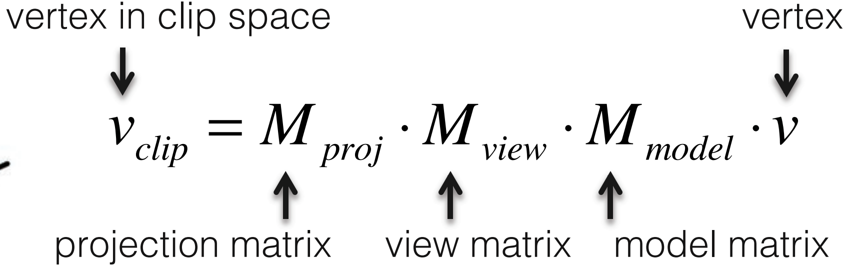
vertex

$$v_{clip} = M_{proj} \cdot M_{view} \cdot M_{model} \cdot v$$



oculus.com

from lecture 2:



vertex in clip space

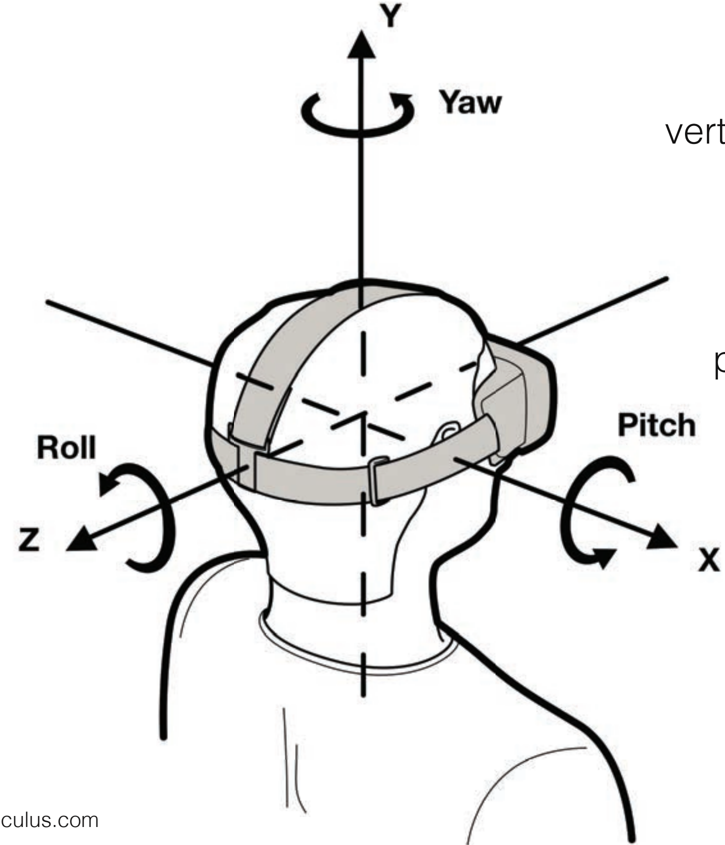
vertex

$$v_{clip} = M_{proj} \cdot M_{view} \cdot M_{model} \cdot v$$

projection matrix

view matrix

model matrix



oculus.com

from lecture 2:

vertex in clip space

vertex

$$v_{clip} = M_{proj} \cdot M_{view} \cdot M_{model} \cdot v$$

projection matrix

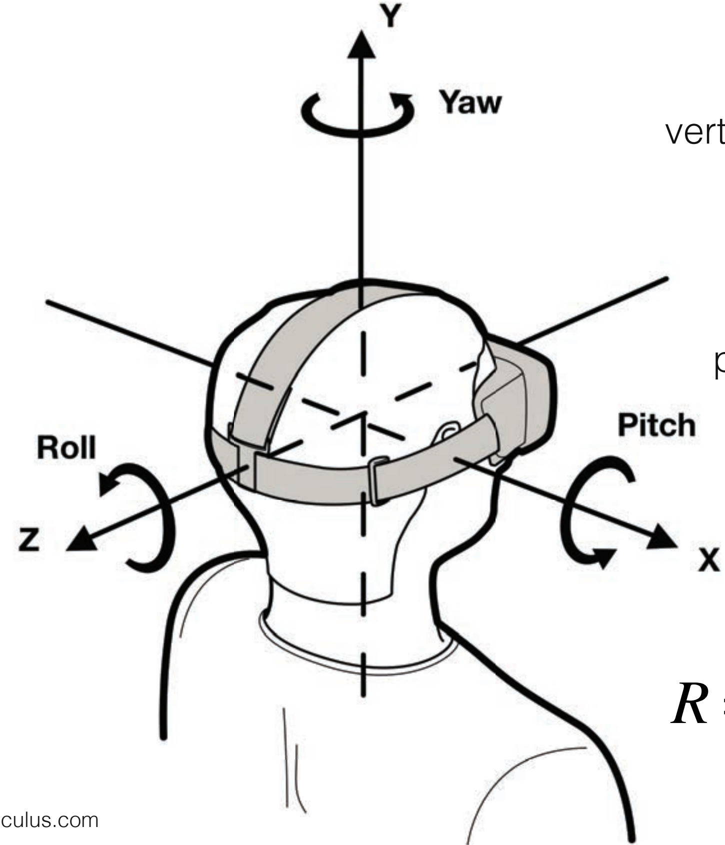
view matrix

model matrix

rotation

translation

$$M_{view} = R \cdot T(-eye)$$



oculus.com

from lecture 2:

vertex in clip space ↓ vertex ↓

$$v_{clip} = M_{proj} \cdot M_{view} \cdot M_{model} \cdot v$$

↑ projection matrix ↑ view matrix ↑ model matrix

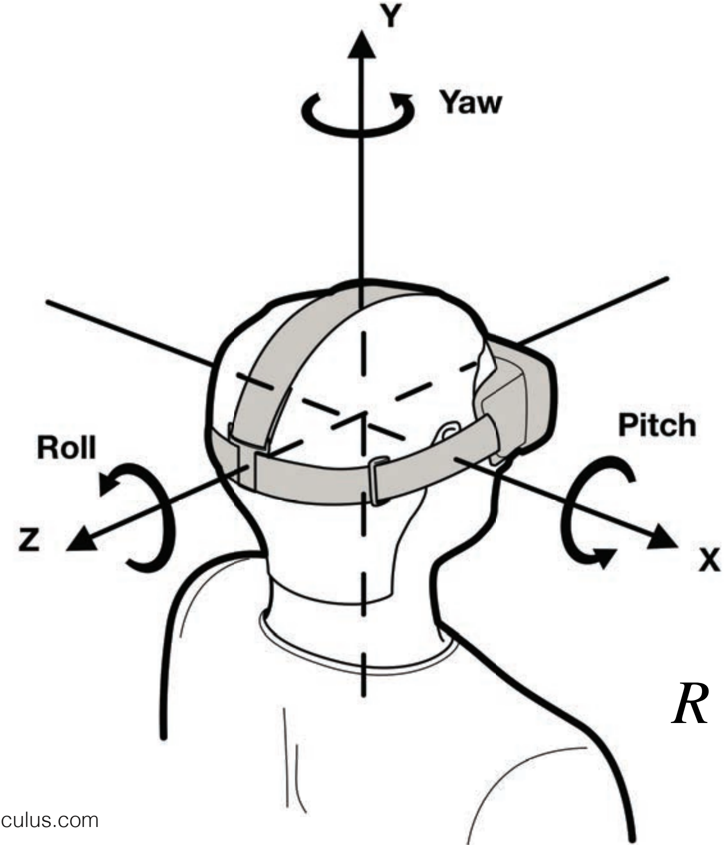
rotation ↓ translation ↓

$$M_{view} = R \cdot T\left(-eye\right)$$

↑

$$R = R_z\left(-\theta_z\right) \cdot R_x\left(-\theta_x\right) \cdot R_y\left(-\theta_y\right)$$

roll pitch yaw



oculus.com

2 important
coordinate systems:

<u>body/sensor frame</u> <u>inertial frame</u>

↓ ↓

$$M_{view} = R \cdot T\left(-eye\right)$$

↑

$$R = R_z\left(-\theta_z\right) \cdot R_x\left(-\theta_x\right) \cdot R_y\left(-\theta_y\right)$$

roll pitch yaw

Remember: Dead Reckoning with Gyro

- integrate angular velocity from gyro measurements:

$$\theta(t+\Delta t) \approx \theta(t) + \dot{\theta}(t)\Delta t + O(\Delta t^2)$$

- for each axis separately:

$$\theta_x = \theta_x + \omega_x \Delta t$$
$$\theta_y = \theta_y + \omega_y \Delta t$$
$$\theta_z = \theta_z + \omega_z \Delta t$$

Remember: Dead Reckoning with Gyro

- big challenge with gyro: drift over time
- why drift? mostly integration error
- possible solution (?): try using accelerometer data instead of gyro

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

normalize gravity vector in
inertial coordinates

$$\frac{1}{||\tilde{a}||}\tilde{a} = R\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = R_z\left(-\theta_z\right)\cdot R_x\left(-\theta_x\right)\cdot R_y\left(-\theta_y\right)\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

normalize gravity vector rotated into
sensor coordinates

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

$$\frac{1}{||\tilde{a}||}\tilde{a} = R\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = R_z\left(-\theta_z\right)\cdot R_x\left(-\theta_x\right)\cdot R_y\left(-\theta_y\right)\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} \cos\left(-\theta_z\right) & -\sin\left(-\theta_z\right) & 0 \\ \sin\left(-\theta_z\right) & \cos\left(-\theta_z\right) & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\left(-\theta_x\right) & -\sin\left(-\theta_x\right) \\ 0 & \sin\left(-\theta_x\right) & \cos\left(-\theta_x\right) \end{pmatrix}\begin{pmatrix} \cos\left(-\theta_y\right) & 0 & \sin\left(-\theta_y\right) \\ 0 & 1 & 0 \\ -\sin\left(-\theta_y\right) & 0 & \cos\left(-\theta_y\right) \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

$$\hat{a} = \frac{1}{||\tilde{a}||}\tilde{a} = \begin{pmatrix} -\sin(-\theta_z)\cos(-\theta_x) \\ \cos(-\theta_z)\cos(-\theta_x) \\ \sin(-\theta_x) \end{pmatrix}$$

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

roll

$$\hat{a} = \frac{1}{\|\tilde{a}\|}\tilde{a} = \begin{pmatrix} -\sin(-\theta_z)\cos(-\theta_x) \\ \cos(-\theta_z)\cos(-\theta_x) \\ \sin(-\theta_x) \end{pmatrix} \Rightarrow \frac{\hat{a}_x}{\hat{a}_y} = \frac{-\sin(-\theta_z)}{\cos(-\theta_z)} = -\tan(-\theta_z)$$

$$\boxed{\theta_z = -\operatorname{atan2}(-\hat{a}_x, \hat{a}_y) \text{ in rad} \in [-\pi, \pi]}$$

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

pitch

$$\hat{a} = \frac{1}{\|\tilde{a}\|}\tilde{a} = \begin{pmatrix} -\sin(-\theta_z)\cos(-\theta_x) \\ \cos(-\theta_z)\cos(-\theta_x) \\ \sin(-\theta_x) \end{pmatrix}$$

⇨

$$\frac{\hat{a}_z}{\sqrt{\hat{a}_x^2 + \hat{a}_y^2}} = \frac{\sin(-\theta_x)}{\sqrt{\cos^2(-\theta_x)\underbrace{\left(\sin^2(-\theta_z) + \cos^2(-\theta_z)\right)}_{=1}}}$$

$$= \frac{\sin(-\theta_x)}{\cos(-\theta_x)} = \tan(-\theta_x)$$

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

pitch

$$\hat{a} = \frac{1}{\|\tilde{a}\|}\tilde{a} = \begin{pmatrix} -\sin(-\theta_z)\cos(-\theta_x) \\ \cos(-\theta_z)\cos(-\theta_x) \\ \sin(-\theta_x) \end{pmatrix} \Rightarrow \frac{\hat{a}_z}{\sqrt{\hat{a}_x^2 + \hat{a}_y^2}} = \frac{\sin(-\theta_x)}{\sqrt{\cos^2(-\theta_x)\underbrace{\left(\sin^2(-\theta_z) + \cos^2(-\theta_z)\right)}_{=1}}}$$

$$\theta_x = -\operatorname{atan2}\left(\hat{a}_z, \sqrt{\hat{a}_x^2 + \hat{a}_y^2}\right) \text{ in rad } \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$$

Pitch and Roll from Accelerometer

- use only accelerometer data – can estimate pitch & roll, not yaw
- assume no external forces (only gravity) – acc is pointing UP!

pitch

$$\frac{\hat{a}_z}{\sqrt{\hat{a}_x^2+\hat{a}_y^2}} = \frac{\sin(-\theta_x)}{\sqrt{\cos^2(-\theta_x)\underbrace{\left(\sin^2(-\theta_z)+\cos^2(-\theta_z)\right)}_{=1}}}$$

$$\boxed{\theta_x = -\operatorname{atan2}\left(\hat{a}_z, \operatorname{sign}\left(\hat{a}_y\right)\cdot\sqrt{\hat{a}_x^2+\hat{a}_y^2}\right) \text{ in rad} \quad \in[-\pi,\pi]}$$

Tilt Correction

- idea: stabilize gyro drift using noisy accelerometer = 6 DOF sensor fusion
- remember:
    - gyro is reliable in the short run, but drifts over time, mainly due to accumulation of integration error
    - accelerometer is noisy and affected by motion, thus reliable in the long run but not over short periods of time
    - with accelerometer, can only measure tilt not heading

Tilt Correction with Complementary Filtering

- idea for sensor fusion: low-pass filter accelerometer and high-pass filter gyro

<u>linear interpolation between gyro & accelerometer!</u>

pitch: $$\theta_x = \alpha\left(\theta_x + \omega_x \Delta t\right) + (1-\alpha)\,\mathrm{rad2deg}\left(-\mathrm{atan2}\left(\hat{a}_z, \mathrm{sign}\left(\hat{a}_y\right)\cdot\sqrt{\hat{a}_x^2 + \hat{a}_y^2}\right)\right)$$

yaw: $$\theta_y = \theta_y + \omega_y \Delta t$$

roll: $$\theta_z = \alpha\left(\theta_z + \omega_z \Delta t\right) + (1-\alpha)\,\mathrm{rad2deg}\left(-\mathrm{atan2}\left(-\hat{a}_x, \hat{a}_y\right)\right)$$

$$0 \le \alpha \le 1$$

Tilt Correction with Complementary Filtering

- low-pass filter accelerometer and high-pass filter gyro
- very popular among hobbyists – super easy to implement!
- filter used in Oculus DK 1, see LaValle et al. ICRA 2014

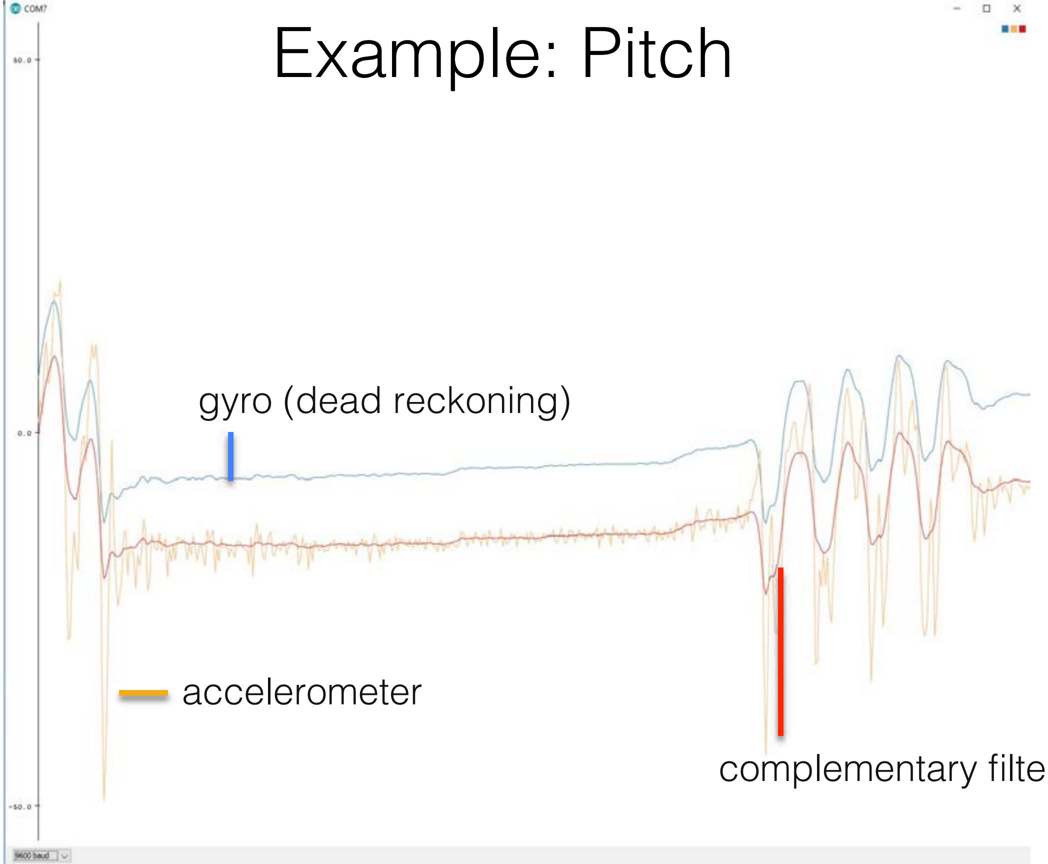
pitch: $$\theta_x = \alpha\left(\theta_x + \omega_x \Delta t\right) + (1-\alpha)\,\mathrm{rad2deg}\left(-\mathrm{atan2}\left(\hat{a}_z, \mathrm{sign}\left(\hat{a}_y\right) \cdot \sqrt{\hat{a}_x^2 + \hat{a}_y^2}\right)\right)$$

yaw: $$\theta_y = \theta_y + \omega_y \Delta t$$

roll: $$\theta_z = \alpha\left(\theta_z + \omega_z \Delta t\right) + (1-\alpha)\,\mathrm{rad2deg}\left(-\mathrm{atan2}\left(-\hat{a}_x, \hat{a}_y\right)\right)$$

$$0 \le \alpha \le 1$$

Example: Pitch



gyro (dead reckoning)

accelerometer

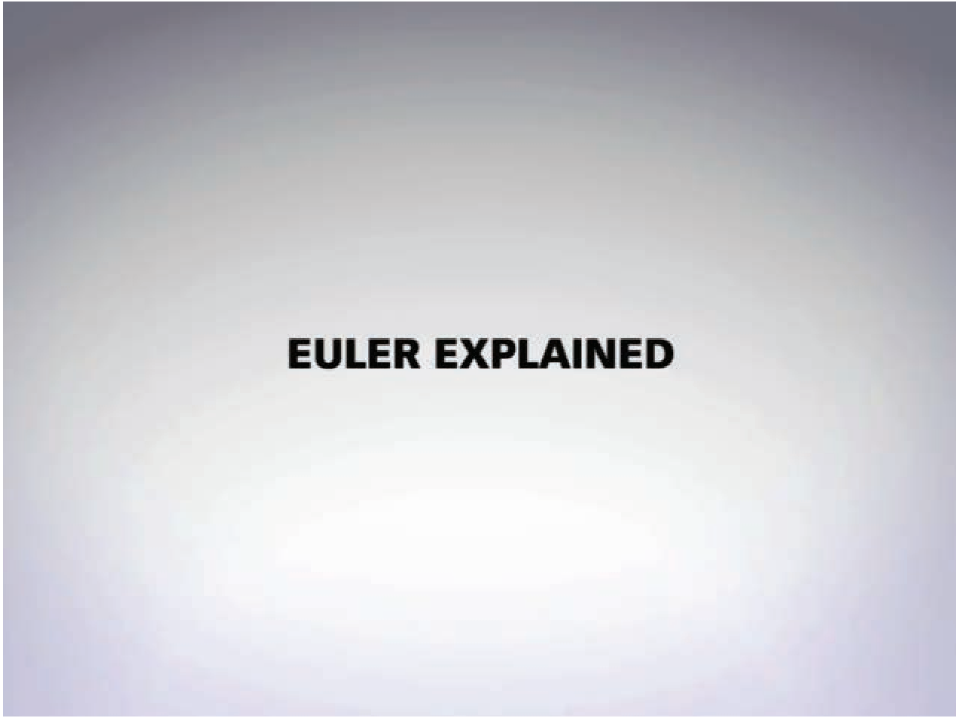
complementary filter

Euler Angles and Gimbal Lock

- so far we have represented head rotations with 3 angles

- problematic when interpolating between keyframes (in computer animation) or integration → singularities

Gimbal Lock

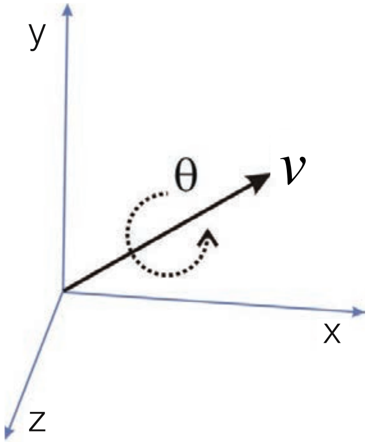


The Guerrilla CG Project, The Euler (gimbal lock) Explained – see youtube.com

Rotations with Axis and Angle Representation

- solution to gimbal lock: use a different representation for head rotation

- axis and angle:



Quaternions!

Quaternions

- what's a quaternion?
- extension of complex numbers – 3 imaginary numbers

$$q = q_0 + q_1 i + q_2 j + q_3 k$$

$$i^2 = j^2 = k^2 = ijk = -1$$

$$i \neq j \neq k$$

Quaternions

- what's a quaternion?
- singularity-free description of rotation

$$q(\theta, v) = \left(\cos\frac{\theta}{2}, v_x \sin\frac{\theta}{2}, v_y \sin\frac{\theta}{2}, v_z \sin\frac{\theta}{2} \right)$$

Quaternions

- 4 values for 3 degrees of freedom (DOF)?
- all quaternions representing a rotation are normalized! → 3DOF
- may have to normalize in practice due to roundoff

$$q(\theta, v) = \left(\cos\frac{\theta}{2}, v_x \sin\frac{\theta}{2}, v_y \sin\frac{\theta}{2}, v_z \sin\frac{\theta}{2} \right)$$

$$||q|| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$$

Quaternion from Axis & Angle

angle in radians

unit-length axis

$$q\left(\overset{\downarrow}{\theta},\overset{\downarrow}{v}\right)=\left(\underbrace{\cos\frac{\theta}{2}}_{\text{scalar = angle}},\underbrace{v_x\sin\frac{\theta}{2},v_y\sin\frac{\theta}{2},v_z\sin\frac{\theta}{2}}_{\text{vector = axis}}\right)$$

Quaternion to Axis & Angle

- convert quaternion to rotation, then use *glm::rotate(*$\boldsymbol{\theta}$*, v_x, v_y, v_z)*

↑

in degrees

$$\theta = 2 \cdot \operatorname{acos}\left(q_0\right)$$

← in radians

$$s = \sqrt{1 - q_0^2}$$

$$v_x = \frac{q_1}{s}, \quad v_y = \frac{q_2}{s}, \quad v_z = \frac{q_3}{s}$$

Quaternion Algebra

- unit quaternion: $\|q\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_2^2} = 1$

- vector quaternion (represents 3D point or vector), need not be normalized:

$$q_v = \left(0, v_x, v_y, v_z\right)$$

- addition: $q + p = \left(q_0 + p_0\right) + \left(q_1 + p_1\right) i + \left(q_2 + p_2\right) j + \left(q_3 + p_3\right) k$

Quaternion Algebra

- conjugate: $q^* = q_0 - q_1 i - q_2 j - q_3 k$
- inverse: $q^{-1} = \frac{q^*}{\|q\|^2}$

Quaternion Algebra

- multiplication: $qp = \left(q_0 + q_1 i + q_2 j + q_3 k\right)\left(p_0 + p_1 i + p_2 j + p_3 k\right)$

$$
\begin{aligned}
= \quad & \left(q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3\right) + \\
& \left(q_0 p_1 + q_1 p_0 + q_2 p_3 - q_3 p_2\right) i \\
& \left(q_0 p_2 - q_1 p_3 + q_2 p_0 + q_3 p_1\right) j \\
& \left(q_0 p_3 + q_1 p_2 - q_2 p_1 + q_3 p_0\right) k
\end{aligned}
$$

Quaternion Algebra

- all quaternions representing rotations are unit quaternions (as opposed to vector quaternions)

- rotation of vector quaternion q_p by q : $q'_p = q q_p q^{-1}$
- inverse rotation: $q'_p = q^{-1} q_p q$

- successive rotations: $q_2 q_1$ rotate first by q_1 and then by q_2

Quaternion-based
Orientation Tracking

Quaternion-based Orientation Tracking

- orientation representation: $q^{(t)}$ is rotation from body/sensor frame to inertial frame at time *t*

- start with initial quaternion: $q^{(0)} = (1,0,0,0)$

- integration – how does the gyro data fit in here?

- 6 DOF sensor fusion gyro + accelerometer (with quaternions)

Quaternion-based Orientation Tracking

- how to get rotation quaternion from gyro measurements?

measurements: $\widetilde{\omega} = \left(\widetilde{\omega}_x, \widetilde{\omega}_y, \widetilde{\omega}_z\right)$ magnitude: $l = \left\|\widetilde{\omega}\right\|$

<u>need this in radians / second!</u>

$$q_\Delta = q\left(\Delta t l, \frac{1}{l}\tilde{\omega}\right) = \left(\underline{\cos\left(\frac{\Delta t l}{2}\right)}, \frac{\tilde{\omega}_x}{l}\sin\left(\frac{\Delta t l}{2}\right), \frac{\tilde{\omega}_y}{l}\sin\left(\frac{\Delta t l}{2}\right), \frac{\tilde{\omega}_z}{l}\sin\left(\frac{\Delta t l}{2}\right)\right)$$

angle = rate of rotation (radians/sec)

Quaternion-based Orientation Tracking

- make sure not to divide by 0 if l=0!

measurements: $\widetilde{\omega} = \left(\widetilde{\omega}_x, \widetilde{\omega}_y, \widetilde{\omega}_z\right)$ magnitude: $l = \left\|\widetilde{\omega}\right\|$

need this in radians / second!

$$q_\Delta = q\left(\Delta t l, \frac{1}{l}\tilde{\omega}\right) = \left(\underline{\cos\left(\frac{\Delta t l}{2}\right)}, \frac{\tilde{\omega}_x}{l}\sin\left(\frac{\Delta t l}{2}\right), \frac{\tilde{\omega}_y}{l}\sin\left(\frac{\Delta t l}{2}\right), \frac{\tilde{\omega}_z}{l}\sin\left(\frac{\Delta t l}{2}\right)\right)$$

angle = rate of rotation (radians/sec)

Quaternion-based Orientation Tracking

- how to integrate rotation quaternion with gyro measurements?

$$q^{(t+\Delta t)} = q^{(t)} q_{\Delta}$$

↑

$$q_{\Delta} = q\left(\Delta t l, \frac{1}{l}\tilde{\omega}\right) = \left(\underline{\cos\left(\frac{\Delta t l}{2}\right),} \frac{\tilde{\omega}_x}{l}\sin\left(\frac{\Delta t l}{2}\right), \frac{\tilde{\omega}_y}{l}\sin\left(\frac{\Delta t l}{2}\right), \frac{\tilde{\omega}_z}{l}\sin\left(\frac{\Delta t l}{2}\right)\right)$$

angle = rate of rotation (radians/sec)

Quaternion-based Orientation Tracking

- how to integrate rotation quaternion with gyro measurements?

$$\boxed{q^{(t+\Delta t)} = q^{(t)} q_{\Delta}}$$

forward (process) model – simple!

- assumption: angular velocity is constant from *t* to *t+1*

Tilt Correction with Quaternions

- assume accelerometer measures gravity vector in body (sensor) coordinates

- transform measurements into inertial coordinates $q_a^{(t)}$

$$q_a^{(\text{inertial})} = q^{(t)} q_a^{(\text{body})} q^{(t)^{-1}}, \quad q_a = \left(0, \tilde{a}_x, \tilde{a}_y, \tilde{a}_z\right)$$

- $q_a^{(\text{body})}, q_a^{(\text{inertial})}$ are vector quaternions

Tilt Correction with Quaternions

- assume accelerometer measures only gravity vector in body (sensor) coordinates
- <u>without gyro drift</u>: in inertial coordinates, accelerometer vector should point UP (0,1,0) with 9.81 m/s^2
- <u>with gyro drift</u>: doesn't point up

- 6DOF sensor fusion to stabilize drift w.r.t. tilt (pitch and roll)

Tilt Correction with Quaternions

1. compute accelerometer in inertial coords $q_a^{(\text{inertial})} = q^{(t)} q_a^{(\text{body})} {q^{(t)}}^{-1}$
2. get normalized vector part of vector quaternion $q_a^{(\text{inertial})}$

$$v = \left(\frac{q_{a_1}^{(\text{inertial})}}{\left\| q_a^{(\text{inertial})} \right\|}, \frac{q_{a_2}^{(\text{inertial})}}{\left\| q_a^{(\text{inertial})} \right\|}, \frac{q_{a_3}^{(\text{inertial})}}{\left\| q_a^{(\text{inertial})} \right\|} \right)$$

3. compute angle between v and UP (0,1,0) with dot product

$$v \bullet (0,1,0) = \|v\| \|(0,1,0)\| \cos\phi \longrightarrow \phi = \operatorname{acos}(v_y) = \operatorname{acos}\left(\frac{q_{a_2}^{(\text{inertial})}}{\left\| q_a^{(\text{inertial})} \right\|} \right)$$

Tilt Correction with Quaternions

4. compute normalized rotation axis between v and UP with cross product

$$v_{tilt} = v \times (0,1,0) = \left(-\frac{v_z}{\sqrt{v_x^2 + v_z^2}}, 0, \frac{v_x}{\sqrt{v_x^2 + v_z^2}} \right)$$

5. tilt correction quaternion is $q_{tilt}(\phi, v_{tilt})$

i.e. $$(0,0,1,0) = q_{tilt}(\phi, v_{tilt}) q^{(t)} q_a^{(\text{body})} q^{(t)^{-1}} q_{tilt}^{-1}(\phi, v_{tilt})$$

Complementary Filter

- same idea as for Euler angles: use gyro measurements and correct tilt

$$q_{tilt\_corrected}^{(t+\Delta t)} = q\left(\left(1-\alpha\right)\phi, v_{tilt}\right) q_{gyro}^{(t+\Delta t)} \quad 0 \leq \alpha \leq 1$$

where $q_{gyro}^{(t+\Delta t)} = q_{tilt\_corrected}^{(t)} q_{\Delta}$ is the updated rotation quaternion from only the gyro measurements (see slide 42)

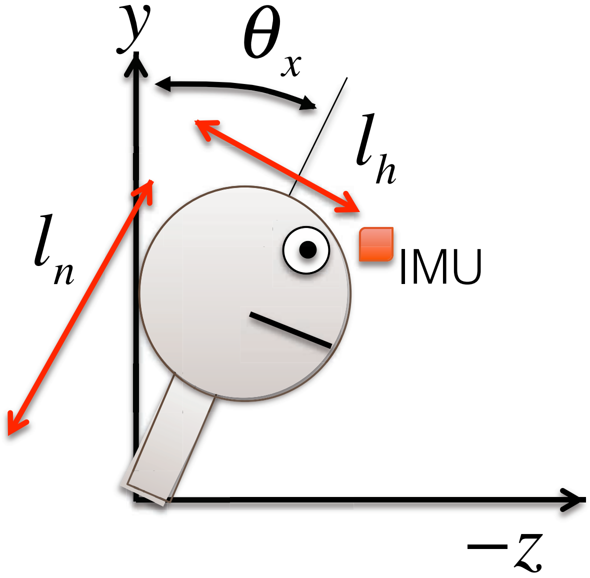
Complementary Filter

- same idea as for Euler angles: use gyro measurements and correct tilt a bit into tilt of accelerometer

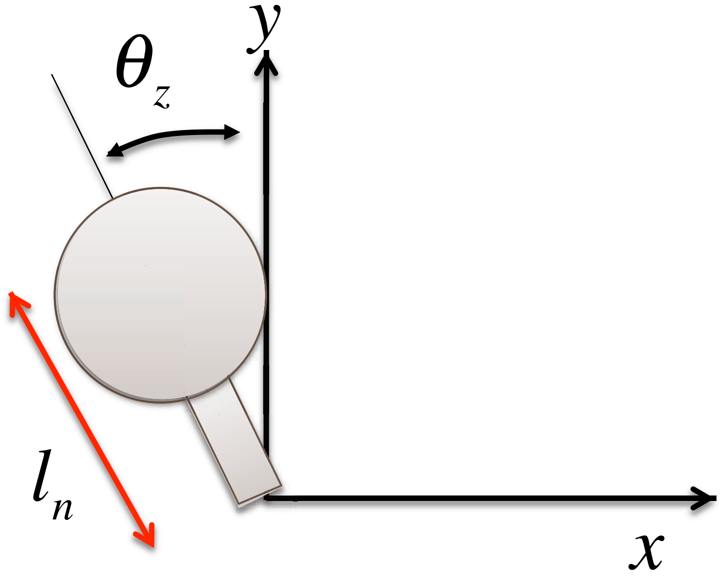
$$q_{tilt\_corrected}^{(t+\Delta t)} = q\left((1-\alpha)\phi, v_{tilt}\right) q_{gyro}^{(t+\Delta t)} \qquad 0 \leq \alpha \leq 1$$

- just remember: all rotation quaternions need to be normalized, vector quaternions (i.e. q_0=0) need not be

Head and Neck Model



pitch around base of neck!



roll around base of neck!

Head and Neck Model

- why? there is not always positional tracking! this gives some motion parallax
- can extend to torso, and using other kinematic constraints
- integrate into pipeline as

$$M_{view} = T\left(0,-l_n,-l_h\right)\cdot R\cdot T\left(0,l_n,l_h\right)\cdot T\left(-eye\right)$$

Additional Information

- S. LaValle, A. Yershova, M. Katsev, M. Antonov “Head Tracking for the Oculus Rift”, Proc. ICRA 2014

- E. Kraft “A Quaternion-based Unscented Kalman Filter for Orientation Tracking”, IEEE Proc. Information Fusion, 2003
- N. Trawny, I. Roumeliotis “Indirect Kalman Filter for 3D Attitude Estimation: A Tutorial for Quaternion Algebra”, Technical Report 2005

Note: I found that none of these papers alone are free of errors or follow the coordinate system we use, so it’s best to use them as general guidelines but follow our notation in class!