Virtual Reality

3D Space & Tracking

Ján Cíger

Reviatech SAS

jan.ciger@gmail.com

http://janoc.rd-h.com/

16.10.2014

Outline

Who am I?

PhD from VRlab, EPFL, Switzerland

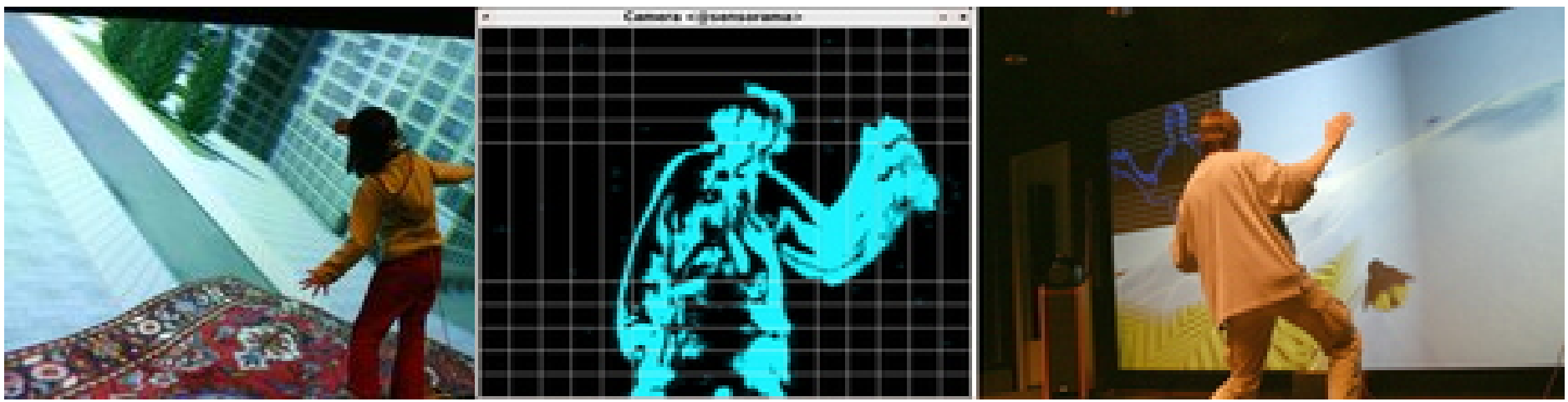
- AI for virtual humans, how to make autonomous characters ("NPCs") smarter
- Lab focused on human character animation, motion capture, interaction in VR
- $\approx$ 20 PhD students, several post-docs, many master students...

VRlab: `http://vrlab.epfl.ch` IIG: `http://iig.epfl.ch`

Who am I?

SensoramaLab, Aalborg University, Denmark

- Building a new lab focusing on new media, interaction, use in rehabilitation
- Large projection system, tracking, many interactive demos
- Teaching bachelor & master students, project supervision.



Who am I?

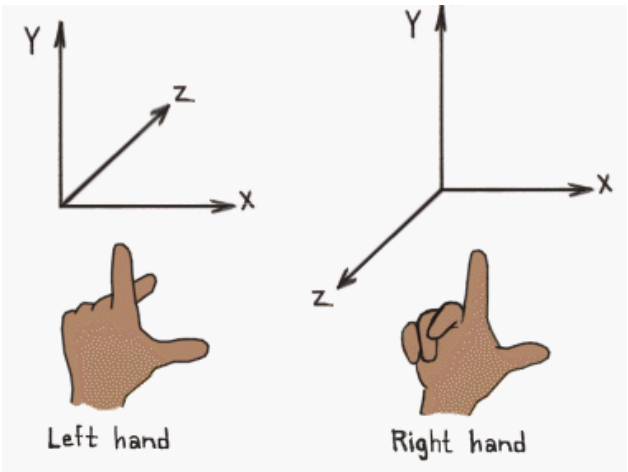
Reviatech SAS

- Applied research, looking for new technologies useful for our products/customers
- Tracking, augmented reality, AI, building software & hardware prototypes. . .

http://www.reviatech.com/

Outline

3D Position

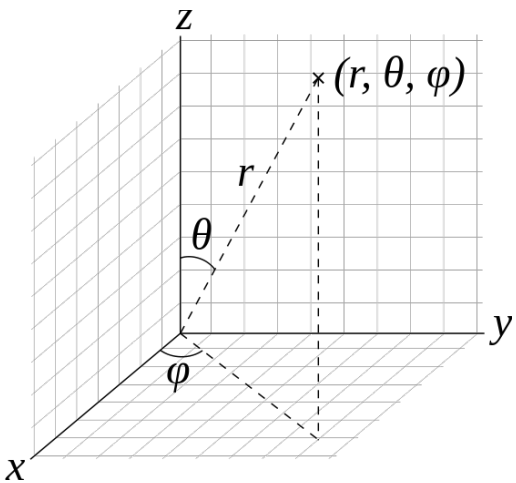


http://viz.aset.psu.edu/gho/sem_notes/3d_fundamentals/html/3d_coordinates.html

Carthesian

- 3 perpendicular axes
- Left or right handed!
- Z-up vs Y-up!
- Usually same scale on all axes
- Common units – meters, millimetres, inches, feet...

3D Position



http://en.wikipedia.org/wiki/Spherical_coordinate_system

Spherical

- Angles θ, ϕ, radius r
- Conventions! (θ, ϕ could be swapped)
- GameTrak uses it (see later!)

Outline

Euler theorem

Definition

Any two orthonormal three-dimensional coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis.

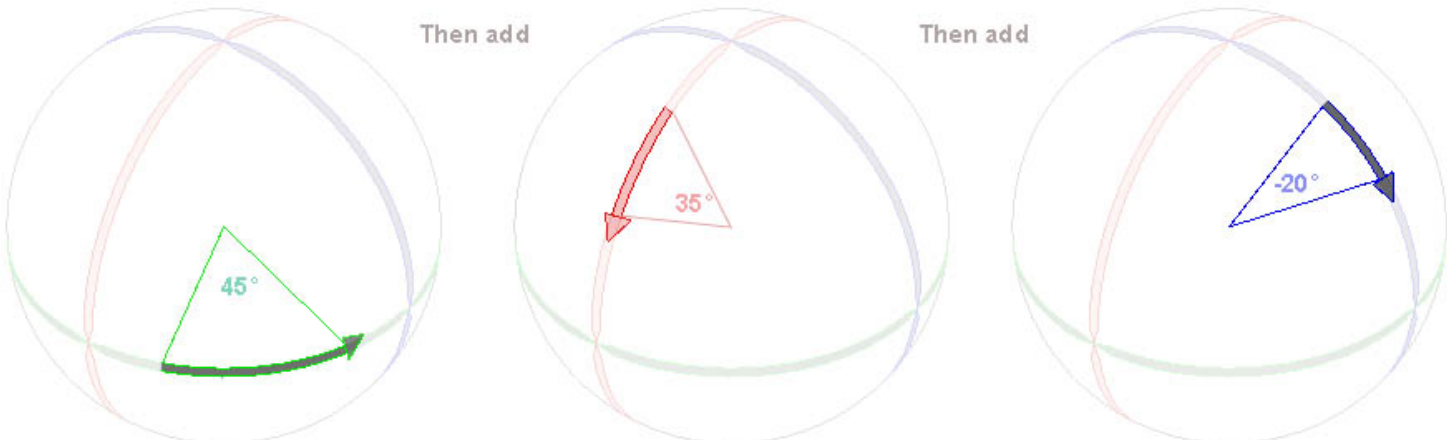
- Leonard Euler (1707-1783)

Euler angles

Euler angles

- Euler theorem means that we can describe any rotation as a composition of three basic rotations around the principal axes.

Example



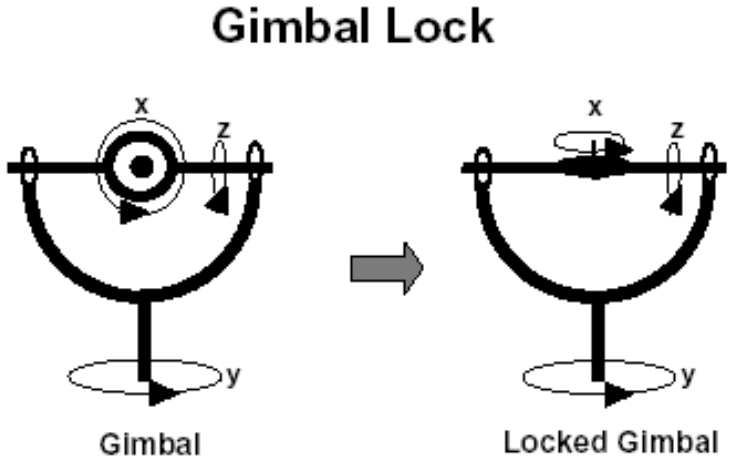
http://www.isner.com/tutorials/quatSpells/quaternion_spells_14.htm

Euler angles

Why Euler angles are bad

- Non-unique way of describing orientation. Rotating in XYZ, ZYX, XZY, . . . produces different trajectory for the object. Good for a robot or space ship but not for a human body.
- Gimbal lock – loss of degree of freedom
- Conversion to/from a matrix is complex (trigonometry)

Example



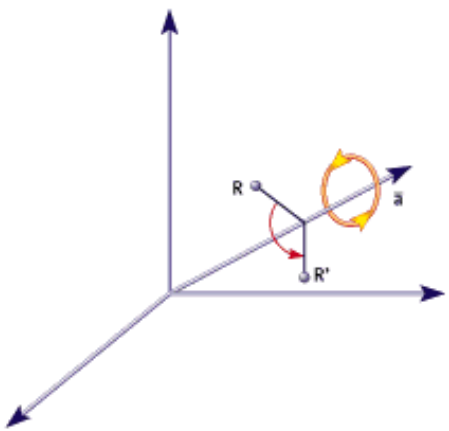
http://www.cs.duke.edu/courses/fall02/cps124/notes/07_rotations/

Axis/angle representation

Axis/angle representation

- Axis of rotation (vector) + angle
- Easy to imagine, no gimbal lock.
- Very hard to interpolate two rotations :(

Example



http://www.cs.duke.edu/courses/fall02/cps124/notes/07_rotations/

Rotation matrices

3×3 matrices

- Non-ambiguous, no gimbal lock, easy to concatenate rotations
- Very verbose – 9 numbers, compare with 3 for Euler angles or 4 for axis/angle.
- Non-intuitive
- Very tricky to interpolate (e.g. turn head from left to right)

Example

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}$$

Rotation around x-axis

$$\begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}$$

Rotation around y-axis

$$\begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation around z-axis

Quaternions

Description

- Invented by Hamilton in 1843
- Deep foundation in algebra
- Each quaternion has four components:

$$q = [q_0, q_1, q_2, q_3]$$

Quaternions

How does it work?

- Extension to complex numbers
- 4 components, one is real, 3 are imaginary forming a vector in imaginary *ijk* space

Example

$$q = q_0 + iq_1 + jq_2 + kq_3$$

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

Quaternions

Notation

- Sometimes written as: $q = \langle s, v \rangle$, where:

$$s = q_0$$

$$v = [q_1, q_2, q_3]$$

- Also written as: $q = [w, q_0, q_1, q_2]$ or $q = [q_0, q_1, q_2, w]$, where q_i are imaginary components and w is real.
- Be very careful which convention is in use if you use some 3rd party library $\rightsquigarrow$ hard to find bugs.

Quaternions

Unit quaternions

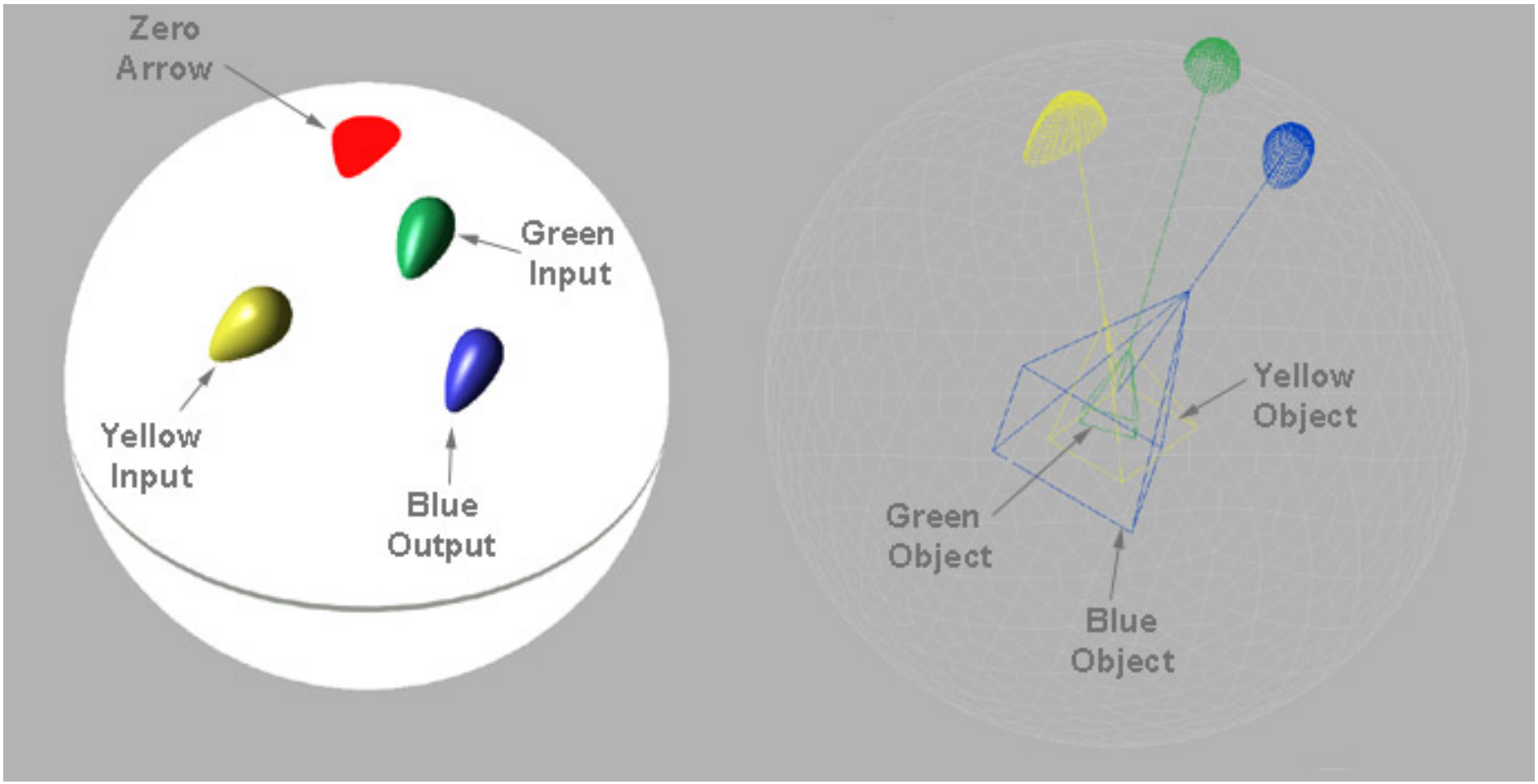
- Rotations are expressed only using unit quaternions:

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

- Correspond to a surface of 4D hypersphere of radius 1
- We have 4D space, but only 3 independent coordinates $\rightsquigarrow$ representation of a 3D object using 4D coordinates (similar to spherical coordinates – two angles, radius 1 and the 4th coordinate is a "twist" around this axis).
- How to visualize that?

Quaternions

How quaternions represent orientation/rotations



http://www.isner.com/tutorials/quatSpells/quaternion_spells_14.htm

Basic properties of quaternions

Quaternions as rotations

- Quaternion q can represent rotation of angle θ around an unit axis a:

$$q = [cos\frac{\theta}{2}, a_x sin\frac{\theta}{2}, a_y sin\frac{\theta}{2}, a_z sin\frac{\theta}{2}]$$

- Or:

$$q = \langle cos\frac{\theta}{2}, a\ sin\frac{\theta}{2} \rangle$$

Basic properties of quaternions

Axis/angle to quaternion

Axis: $\vec{v}$

Angle: α

$$q = [w, q_0, q_1, q_2]$$

$$\vec{u} = \frac{\vec{v}}{|\vec{v}|}$$

$$q_i = u_i sin\frac{\alpha}{2}, i \in [0, 1, 2]$$

$$w = cos\frac{\alpha}{2}$$

Basic properties of quaternions

Quaternion to axis/angle

Quaternion $q = [w, q_0, q_1, q_2]$

$$\alpha = 2acos\ w$$

$$v_i = \frac{q_i}{\sqrt{1 - w^2}}, i \in [0, 1, 2]$$

If $\sqrt{1 - w^2} = 0$ then:

$$v_i = q_i, i \in [0, 1, 2]$$

Basic properties of quaternions

Other conversions

- Left as an exercise for the reader...
  See: “Matrix & quaternion FAQ”:

  `http://skal.planet-d.net/demo/matrixfaq.htm`

Basic properties of quaternions

Angle between two quaternions

- Dot product of two quaternions:

$$p \cdot q = p_0 q_0 + p_1 q_1 + p_2 q_2 + p_3 q_3 = |p||q| cos\ \alpha$$

- Angle between two quaternions in 4D space is one half of the angle needed to rotate the object from one orientation to the other.

Basic properties of quaternions

Conjugate and inverse of a quaternion

- Conjugate:

$$q = [q_0, q_1, q_2, q_3]$$

$$q^* = [q_0, -q_1, -q_2, -q_3]$$

- Inverse:

$$q^{-1} = \frac{q^*}{|q|^2}$$

Basic properties of quaternions

Quaternion multiplication

- We can multiply quaternions if we expand them into their complex number form:

$$p = p_0 + ip_1 + jp_2 + kp_3$$

$$q = q_0 + iq_1 + jq_2 + kq_3$$

$$pq = (p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3) + i(p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2)$$

$$+j(p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1) + k(p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0)$$

- If $\vec{u} = [p_1, p_2, p_3]$ and $\vec{v} = [q_1, q_2, q_3]$:

$$pq = \langle p_0q_0 - \vec{u} \cdot \vec{v}, p_0\vec{v} + q_0\vec{u} + \vec{u} \times \vec{v} \rangle$$

Basic properties of quaternions

Quaternion multiplication

- If quaternions p, q represent rotations, then pq represents p rotated by q
- Quaternion multiplication is non-commutative – i.e. order matters.

Basic properties of quaternions

Quaternion multiplication

- Multiplication of a vector by a quaternion – rotation of a vector:

$$v' = q \cdot v \cdot q^{-1}$$

- Simpler and faster to calculate than matrix multiplication!

Summary

Quaternions

Pros:

- Compact – only 4 numbers
- Easy to concatenate, interpolate.
- Robust – no gimbal lock

Cons:

- Non-intuitive :(

Outline

Why tracking?

Why do we need to track?

- Interaction – hand, finger
- Stereoscopy – head tracking
- Locomotion
- User activity observation
- ...

Terminology

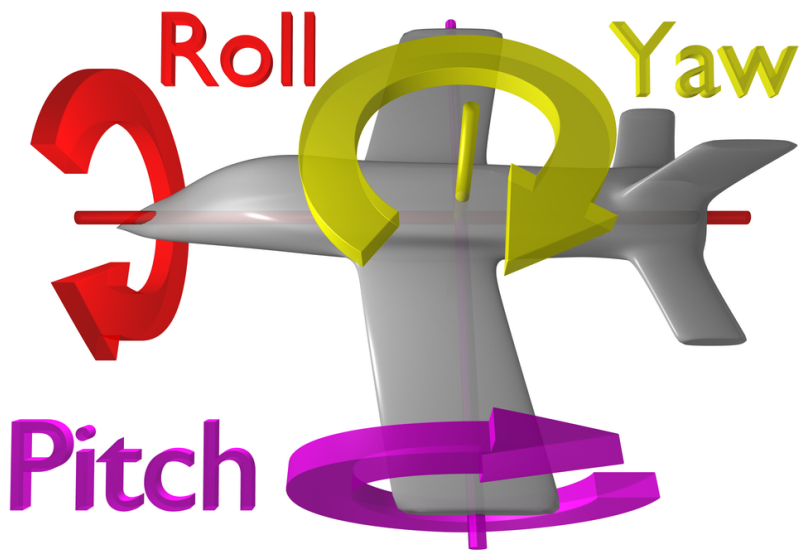
Degrees of Freedom (DoF)

Degree of freedom – one independent parameter describing the movement of an object

6 DoF in 3D space

- Translation X, Y, Z
- Rotation X, Y, Z

A bit of aside – Euler angles are often called yaw, pitch, roll. Which is which axis depends – check documentation!



http://en.wikipedia.org/wiki/Degrees_of_freedom_

Terminology

9 DoF? 9-axis? 10 DoF??

HOME / PRODUCT CATEGORIES / IMU / SEN-11486

9 Degrees of Freedom Breakout - MPU-9150

SEN-11486 ROHS

Description: The 9DOF MPU-9150 is the w
MotionTracking MEMS device designed for t
low cost, and high performance requirement
electronics equipment including smartphone

`https://www.sparkfun.com/products/11486`

Terminology

9 DoF? 9-axis? 10 DoF??

HOME / PRODUCT CATEGORIES / IMU / SEN-11486

9 Degrees of Freedom Breakout - MPU-9150

SEN-11486 ROHS

Description: The 9DOF MPU-9150 is the w
MotionTracking MEMS device designed for t
low cost, and high performance requirement
electronics equipment including smartphone

`https://www.sparkfun.com/products/11486`

Terminology abuse!

- 3 DoF inertial motion unit (IMU) – orientation in space.
- "9 DoF/9-axis" = 3 sensors (accelerometer, gyro, magnetometer), 3 axes each $\rightsquigarrow$ "9 DoF"
- "6 DoF IMU" – still 3 DoF, lack magnetometer.
- "10 DoF IMU" – has extra pressure or temp sensor...

Outline

Tracking technologies

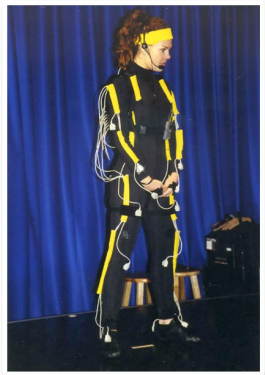
Mechanical

- Linkages + encoders (optical, potentiometres, capacitive. . .)
- **Pros:** Robust, fast, cheap (usually), excellent accuracy
- **Cons:** Range
- **Examples:** Gametrak, FakeSpace BOOM, Phantom, MetaMotion Gypsy

Tracking technologies

Magnetic

- 3 magnetic coils as emitters
- 3 magnetic coils as receivers (sensors)
- **Pros:** Robust, fast
- **Cons:** Complicated math, sensitivity to metal, range
- **Examples:** Razer Hydra, Flock of Birds, MotionStar, Polhemus...



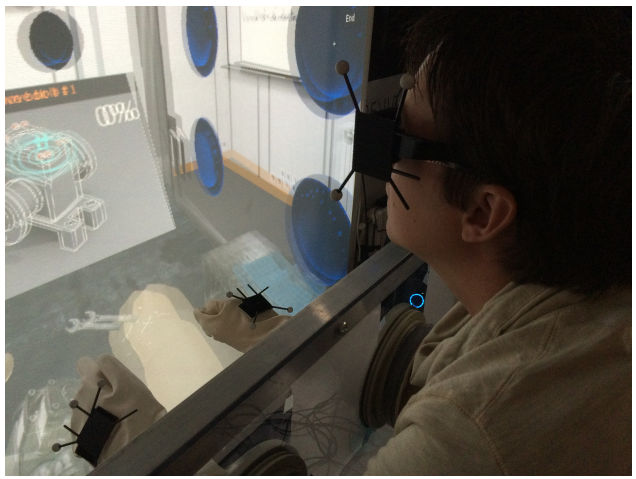
http://www.sfu.ca/~yma15/iat445/research.html

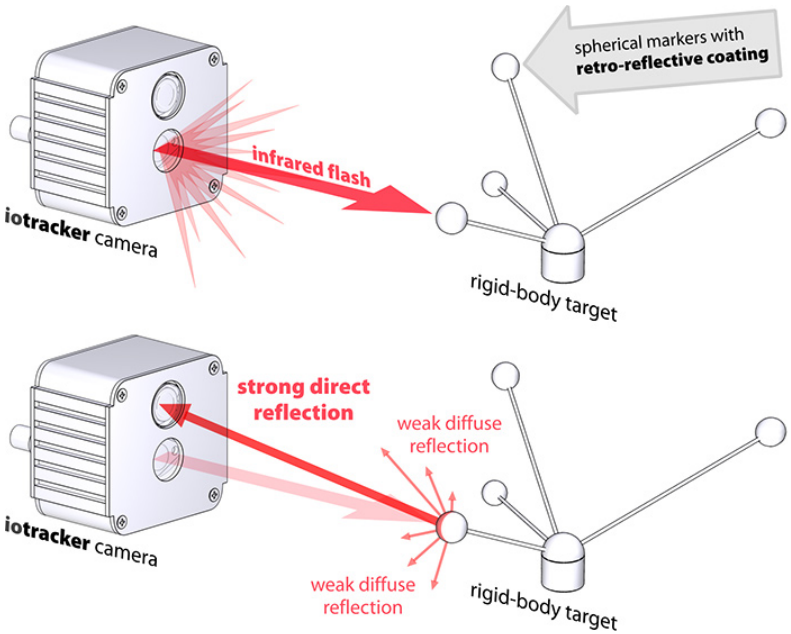
http://souvrsophie.blog.com/page/7/

Tracking technologies

Optical

- 1+ cameras
- May or may not use markers
- **Pros:** Accurate
- **Cons:** Expensive, difficult for occluded space
- **Examples:** Vicon, OptiTrack, ARTrack, iotracker, Kinect...





http://iotracker.com/indexdaed.html?q=optical_tracking

Tracking technologies

Inertial + hybrid

- Usually only 3 DoF (orientation)
- Hybrid systems permit 6 DoF
- **Pros:** Cheap, self-contained
- **Cons:** Limited DoF, mocap
- **Examples:** Wiimote, PS Move, XSens...

http://www.xsens.com/products/xsens-mvn/

Low cost tracking

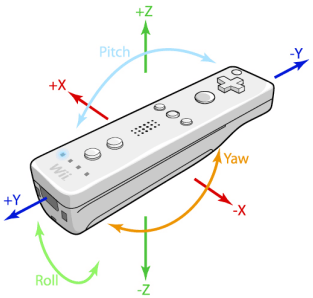
Low cost is relative

- In "pro" field anything $\leq$10000€ is considered low cost. . .
- "Enthusiast" low cost $\leq$500€

Devices (not exhaustive list)

- Wiimote
- PS Move
- Kinect (Asus Xtion, etc)
- GameTrak
- Leap Motion
- Razer Hydra (STEM)
- Novint Falcon
- Freetrack
- TrackIR
- Webcam (ARToolkit, . . .)

Low cost tracking

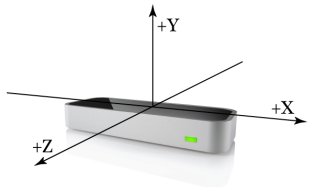


Wiimote

PS Move

Kinect

GameTrak



Leap Motion

Razer Hydra

Novint Falcon

Freetrack

Track IR

Webcam

Low cost tracking

IMUs

- Useful for head tracking (3DoF)
- Combine with Kinect or GameTrak for full 6DoF!
- Usually requires an Arduino or something similar (electronics)
- Ready-made ones are expensive (e.g. Inertia Cube $\approx$ 600€!)

Where to get one

- Just use Wiimote (simplest, ugly)
- US: SparkFun (expensive, but support)
- China: DealXtreme, AliExpress (cheap, easy to get ripped off!)
- Build one! (Can you solder 3x3mm lead-less 24pin chip?)

Outline

What is VRPN?

Virtual Reality Peripheral Network

- `http://www.cs.unc.edu/Research/vrpn/`
- Originally research project, University of Nothern Carolina Chapel Hill, maintainer Russell M. Taylor II
- Industry standard, almost every tracker, motion capture system and VR device supports it
- Open source, portable, free to use

What is VRPN?

Virtual Reality Peripheral Network

- Low latency/overhead
- Simple to use
- Network transparent
- Device abstraction & factoring
- Device sharing between multiple applications
- Many devices supported natively

Device abstraction

Types of devices

Analog	float value $\in < -1, 1 >$, e.g. joystick axis
Button	Abstract button on/off
Dial	Incremental rotation, e.g. Griffin Powermate jog wheel
ForceDevice	Force feedback devices, e.g. Novint Falcon
Sound	Sound generators
Text	Allows passing text messages to/from the device
Tracker	Abstract 6DoF tracker class

Device abstraction

Device factorization

- Every device can be decomposed into abstract devices
- Applications only need to support the abstract device types, never specific hardware!

Example

Razer Hydra	1 6DoF tracker (2 sensors), 14 buttons, 2 analogs (triggers)
Wiimote	11 buttons, 3 analogs (6 if Motion+ installed)
Kinect	6DoF tracker with 15 sensors, however, some return only 3DoF data (position)

VRPN

Server

- Native, see vrpn.cfg – all supported devices are documented there
- 3rd party – e.g. OptiTrack or other vendors provide VRPN compatible servers for their hw.
- One server can handle multiple devices!
- Device ID: `device@host:port`, e.g. `Hydra0@192.168.1.1:5000`

Client(s)

- Receive data from the server over TCP/IP
- Typically some sort of 3D/VR application
- Make sure firewall isn't blocking it (both TCP and UDP needed!)

VRPN – Coordinate systems

Device to Sensor M_{d2s}

Fixed offset between the "active" part of the tracked object & the marker (we measure it).

Sensor to Tracker M_{s2t}

Position/orientation of the sensor relative to the tracker origin/base (tracker reports this).

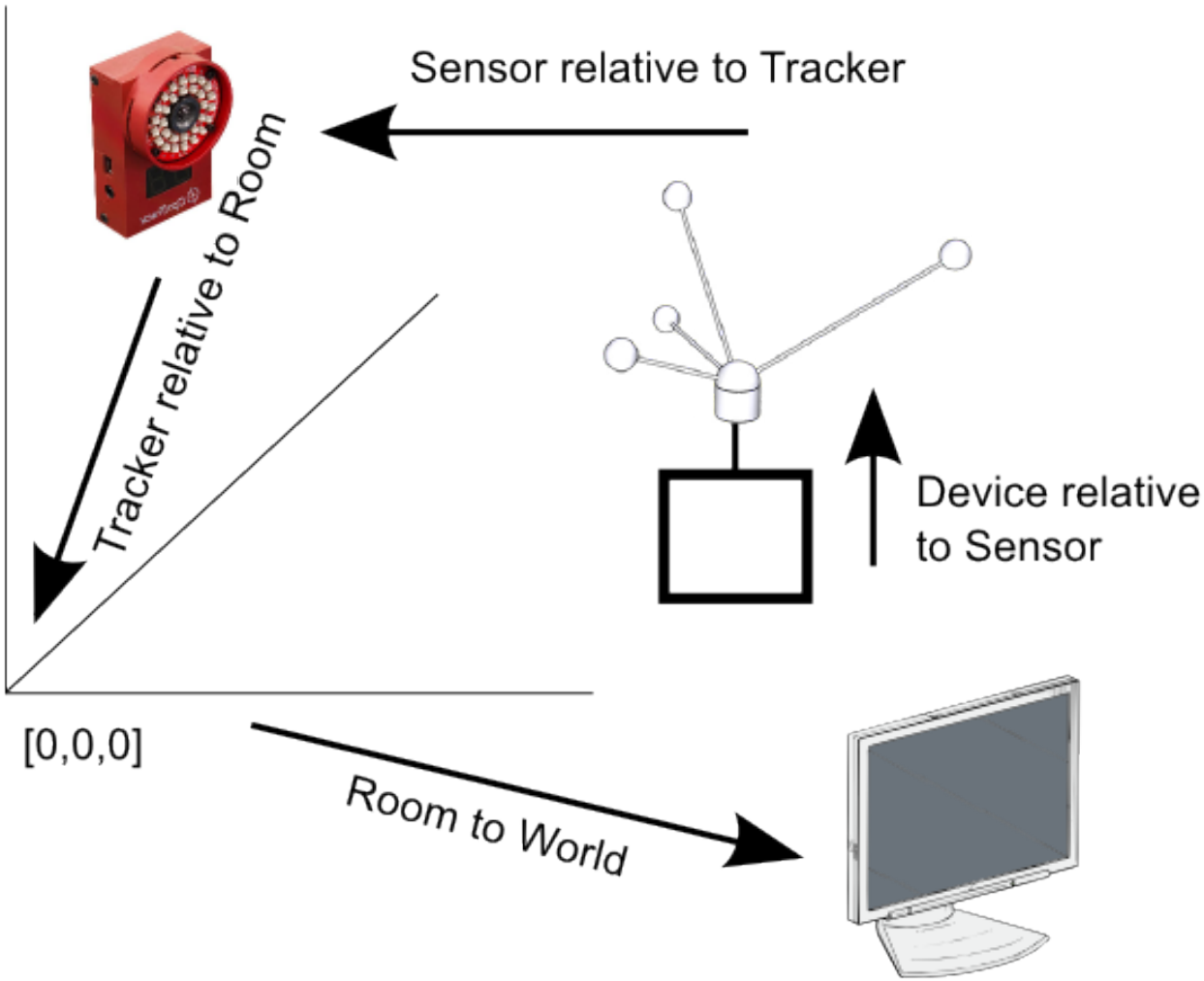
Tracker to room M_{t2r}

Positions the tracker in the physical space – the origin of the tracker doesn't always correspond to the origin of the room!

Room to world $M_{room2world}$

Adapts the 3D rendering coordinates to match the physical space (not always required).

VRPN – Coordinate systems



VRPN – Coordinate systems

Moving an object in the 3D world

Object transformation matrix:

$$M_{sensor} = (M_{s2t} \times M_{d2s})$$

$$M_{world} = M_{room2world} \times M_{t2r} \times M_{sensor} \times M_{t2r}^{-1} \times M_{room2world}^{-1}$$

$$p'_{world} = M_{world} \times p_{world}$$

Why handling position & orientation separately?

- If you don't it would translate in a rotated coordinate space or rotate around a translated origin.
- Likely not what you want!

VRPN – Coordinate systems

Determining the matrices

- $M_{room2world}$, M_{t2r}, M_{d2s} are constant!
- M_{d2s} can be measured (calipers, ruler, etc.)
- $M_{room2world}$ is arbitrary, depends on the app (can be identity!)
- M_{t2r} can be determined by calibration, typically orientation + scale of the tracker relative to the room.

VRPN – Coordinate systems

Simple calibration procedure

1. Move sensor one meter to the front, then to the right from the same spot (desired room origin).
2. Record the 3 points p_1, p_2, p_3 reported by the tracker.
3. Calculate vectors: $\vec{v}_x = p_2 - p_1$, $\vec{v}_y = p_3 - p_1$, $\vec{v}_z = \vec{v}_x \times \vec{v}_y$
4. Normalize: $\vec{n}_x = \frac{\vec{v}_x}{|\vec{v}_x|}$, $\vec{n}_y = \frac{\vec{v}_y}{|\vec{v}_y|}$, $\vec{n}_z = \frac{\vec{v}_z}{|\vec{v}_z|}$
5. Assemble the matrix: $M_{t2r}^{-1} = \begin{pmatrix} n_x[0] & n_y[0] & n_z[0] & 0 \\ n_x[1] & n_y[1] & n_z[1] & 0 \\ n_x[2] & n_y[2] & n_z[2] & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
6. The vectors are orthogonal, so we can transpose instead of inverting to obtain M_{t2r}: $M_{t2r} = (M_{t2r}^{-1})^T$

Outline

Simple example

VRPN tutorial

`http://www.vrgeeks.org/vrpn/tutorial---use-vrpn`

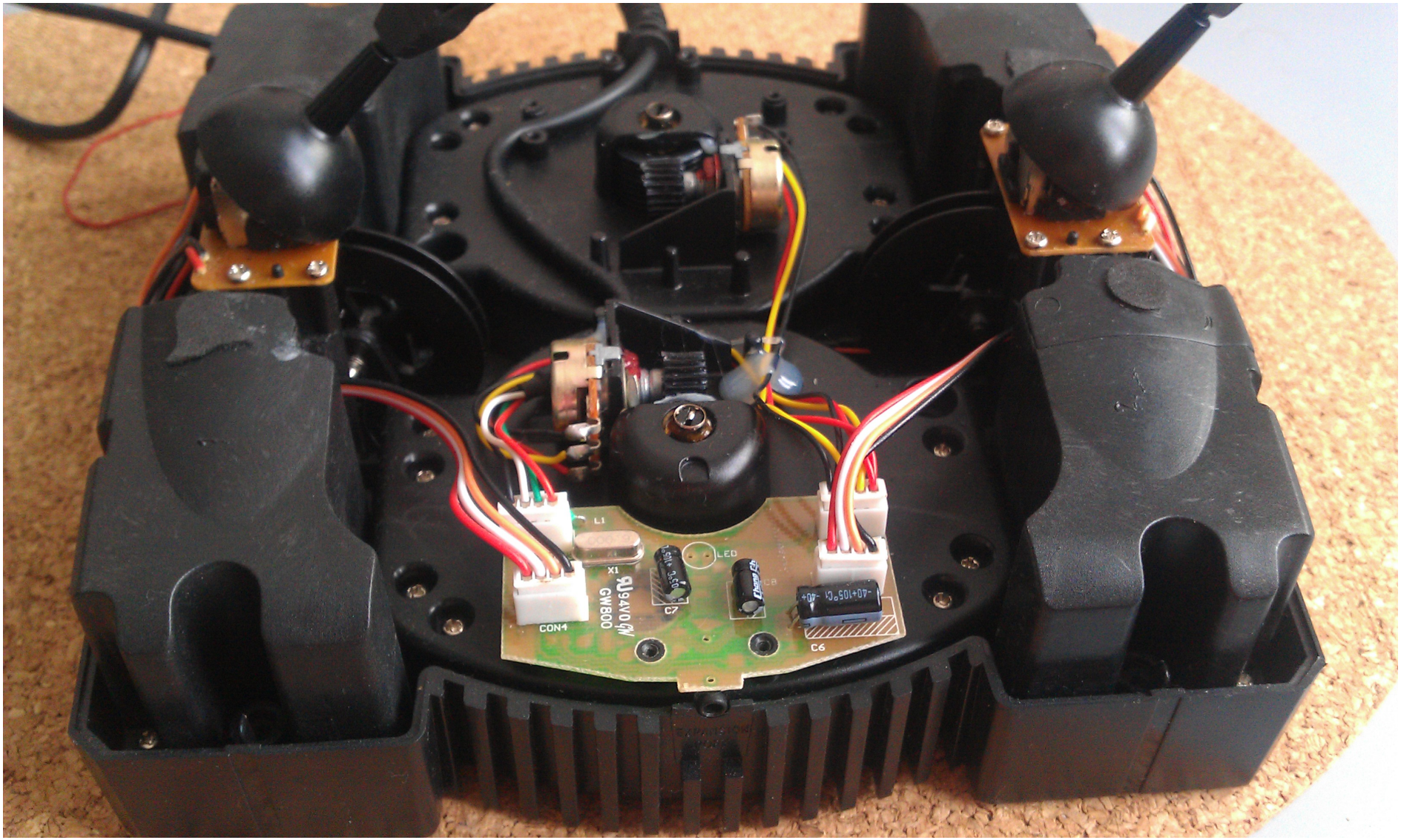
Resources

Few useful links

- Quaternion FAQ `http://www.j3d.org/matrix_faq/matrfaq_latest.html`
- VR Geeks `http://www.vrgeeks.org/`
- VRPN `http://www.cs.unc.edu/Research/vrpn/`
- GlovePie `http://glovepie.org/glovepie.php`
- FreeTrack `http://www.free-track.net/english/`
- Wiiuse `http://public.vrac.iastate.edu/~rpavlik/doxygen/wiiuse-fork/`
- GameTrak hacking `http://janoc.rd-h.com/archives/129`
- FAAST `http://projects.ict.usc.edu/mxr/faast/`
- Free version of Unity 3D + VRPN requires hacks

  `https://github.com/TermWay/UniVRPNity`
- MiddleVR `http://www.imin-vr.com/middlevr-for-unity/`

GameTrak



GameTrak