Virtual Humans – Winter 23/24

Lecture 2_2 – Rotations and Kinematic chains

Prof. Dr.-Ing. Gerard Pons-Moll
University of Tübingen / MPI-Informatics

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Ingredients to build a Virtual Human

Building a human model

- ➢ Kinematic parameterization
  - Rotation Matrices
  - Euler Angles
  - Quaternions
  - Twists and Exponential maps
  - Kinematic chains
- ➢ Subject shape model
  - Geometric primitives
  - Detailed Body Scans
  - Human Shape models

Fitting model to observations

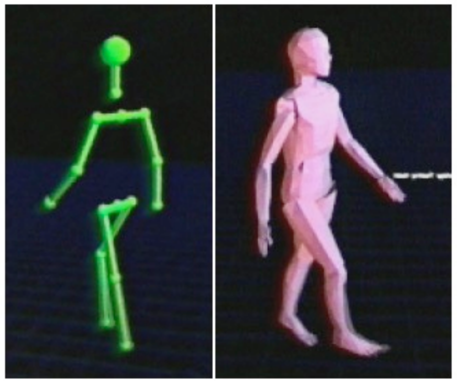
- ➢ Inference
  - Observation likelihood
  - Local optimization
  - Particle Based optimization
  - Directly regressing parameters

Kinematic Chains

Motivated from robotics:
The human motion can be expressed via a „**kinematic chain**“, a series of local rigid body motions (along the limbs).

The model parameters to optimize, correspond to rigid body motions (RBM).

How to model RBM ?



Bregler et.al. CVPR-98

Kinematic Parameterization

1) Pose configurations are represented with a **minimum** number of **parameters**

2) Singularities can be avoided during optimization

3) Easy computation of **derivatives** segment positions and orientations w.r.t parameters

4) Human **motion contrains** such as articulated motion are naturally described

5) Simple rules for **concatenating** motions

Ingredients to build a Virtual Human

Building a human model

- Kinematic parameterization
  - **Rotation Matrices**
  - Euler Angles
  - Quaternions
  - Twists and Exponential maps
  - Kinematic chains
- Subject shape model
  - Geometric primitives
  - Detailed Body Scans
  - Human Shape models

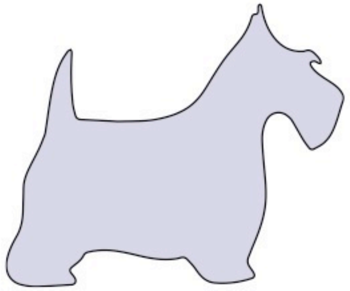
Fitting model to observations

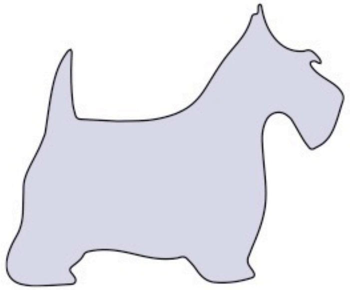
- Inference
  - Observation likelihood
  - Local optimization
  - Particle Based optimization
  - Directly regressing parameters

Informally, what is a rotation?

- It is useful to characterize a **transformation** by its **invariances**.
- A rotation is a linear transformation which preserves angles and distances, and does not mirror the object

Commutativity of Rotations – 2D



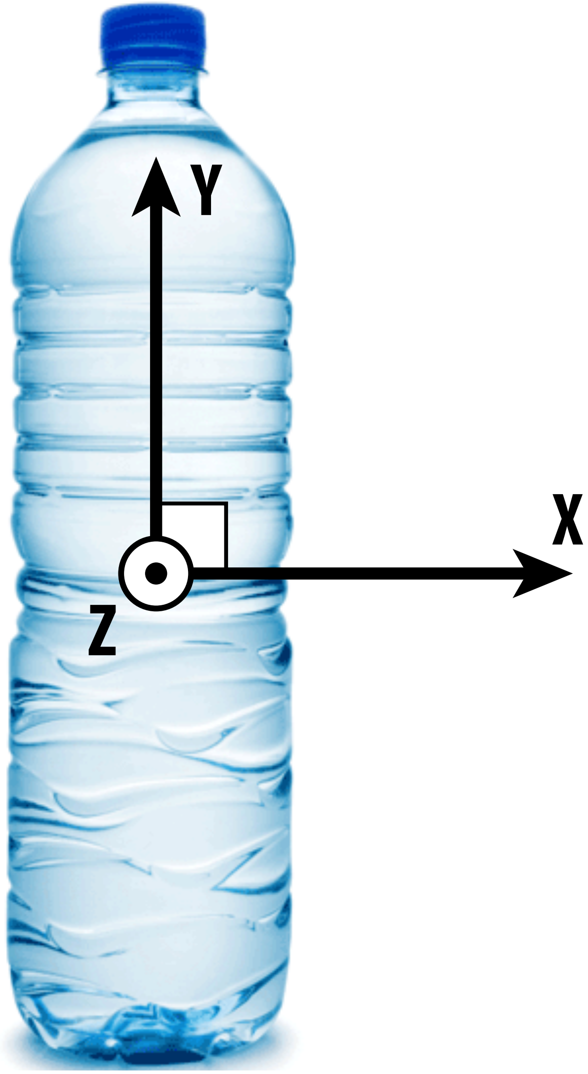


Commutativity of Rotations – 3D

Try it at home – grab a bottle!

- Rotate 90^o around Y, then Z, then X
- Rotate 90^o around Z, then Y, then X
- Was there any difference?

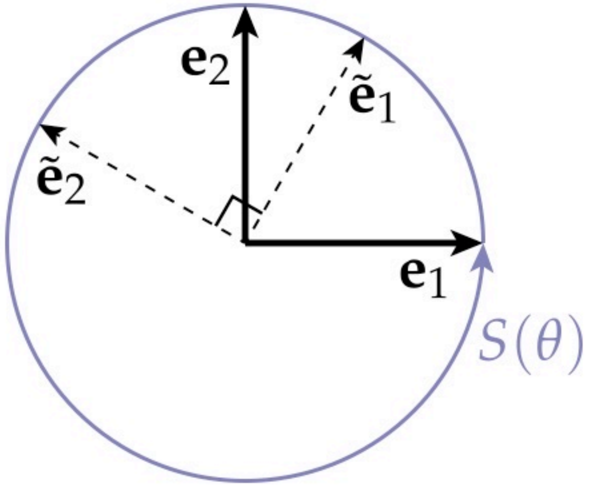
CONCLUSION: bad things can happen if we're not careful about the order in which we apply rotations!



Representing rotations – 2D

- How to get a rotation matrix in 2D?
- Suppose we have a function $S(\theta)$, that for a given θ, gives me the point (x, y) around a circle.
- What's e_1 rotated by θ? $\tilde{\mathbf{e}}_1 = S(\theta)$
- What's e_2 rotated by θ? $\tilde{\mathbf{e}}_2 = S(\theta + \pi/2)$
- How about $u := a.e_1 + b.e_2$?

$$\mathbf{u} := aS(\theta) + bS(\theta + \pi/2)$$



- What then must the matrix look like?

$$\left[\begin{array}{cc} S(\theta) & S(\theta+\pi/2) \end{array}\right] = \left[\begin{array}{cc} \cos(\theta) & \cos(\theta+\pi/2) \\ \sin(\theta) & \sin(\theta+\pi/2) \end{array}\right] = \left[\begin{array}{cc} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{array}\right]$$

Rotation Matrices

$$\mathbf{p}_b = (\lambda_x, \lambda_y, \lambda_z)^T$$

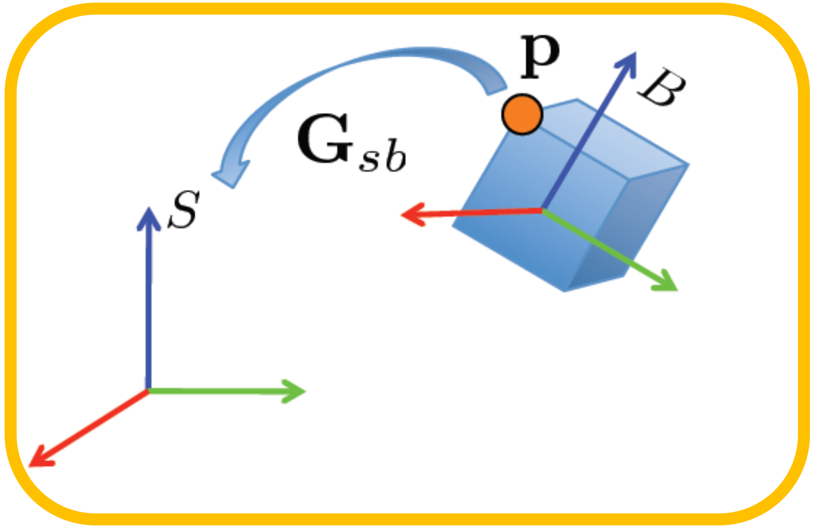
$$\mathbf{p}_s = \lambda_x \mathbf{x}_s^B + \lambda_y \mathbf{y}_s^B + \lambda_z \mathbf{z}_s^B$$

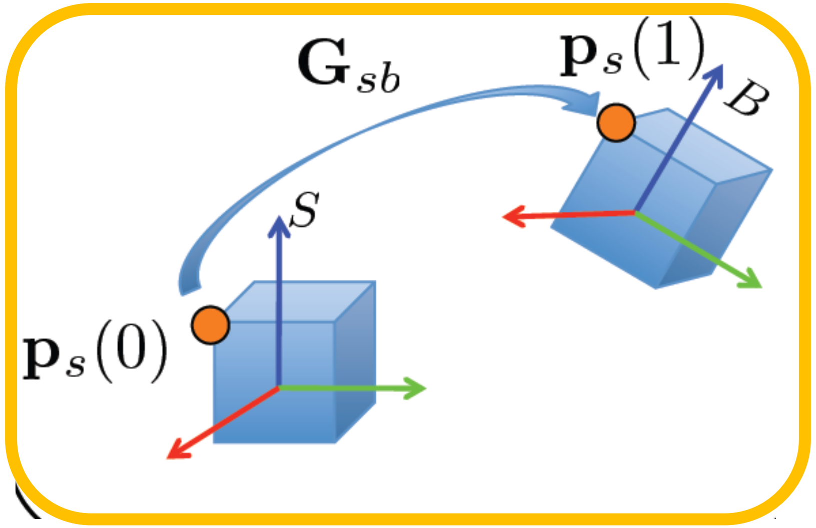
$$\mathbf{p}_s = \mathbf{R}_{sb}\,\mathbf{p}_b \Rightarrow \boxed{\mathbf{R}_{sb} = \begin{bmatrix} \mathbf{x}_s^B & \mathbf{y}_s^B & \mathbf{z}_s^B \end{bmatrix}}$$

The columns of a rotation matrix are the principal axis of one frame expressed relative to another

2 Views of Rotations

Rotations can be interpreted either as





Coordinate transformation

Relative motion in time

Rotation matrix drawbacks

- Need for **9 numbers**

- **6 additional constrains** to ensure that the matrix is orthonormal and belongs to SO(3)

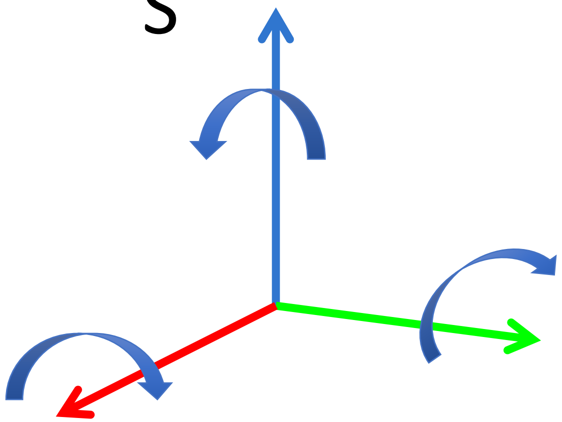
$$SO(3) := \{R \in \mathbb{R}^{3\times 3} \mid RR^T = Id,\ \det(R) = 1\}$$

- Suboptimal for numerical optimization

Euler Angles

- One of the most **popular** parameterizations
- Rotation is encoded as **the successive rotations** about three principal axis
- Only **3 parameters** to encode a rotation
- **Derivatives** easy to compute

Euler Angles



$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\beta & 0 & -sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{\mathbf{R}(\alpha,\beta,\gamma) = \mathbf{R_x}(\alpha)\,\mathbf{R_y}(\beta)\,\mathbf{R_z}(\gamma)}$$

Euler Angles: Confusion

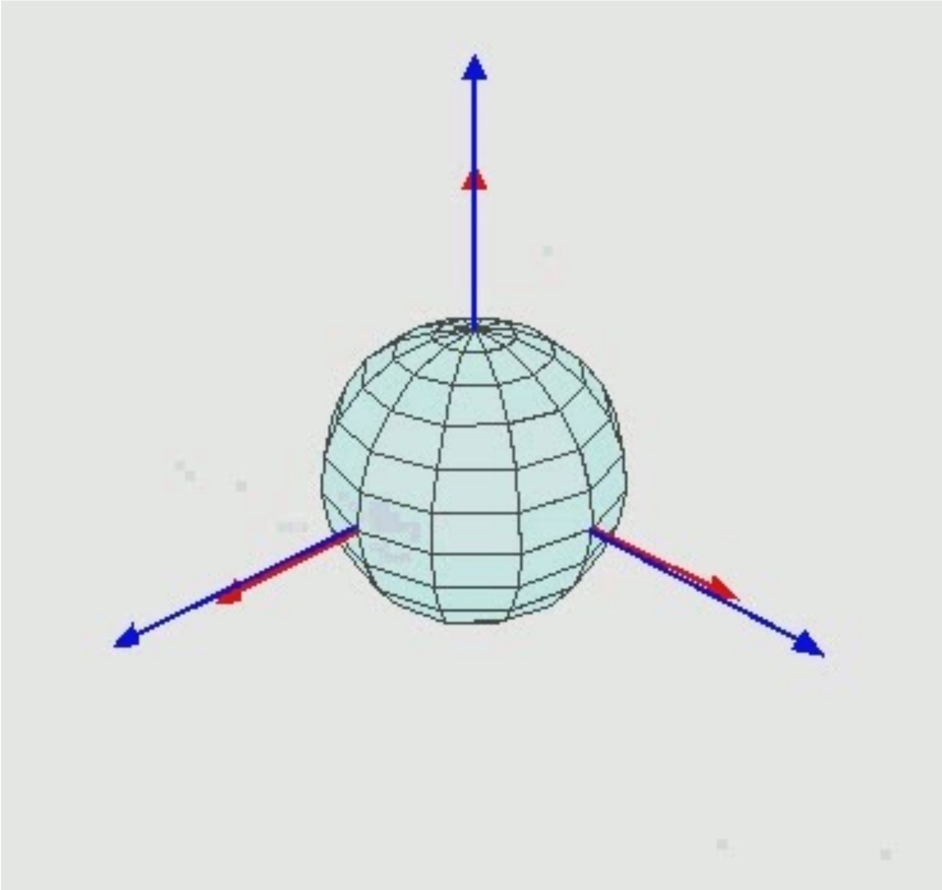
- Careful: Euler angles are a typical source of confusion!

- When using Euler angles **2 things** have to be specified:

  1. Convention: X-Y-Z, Z-Y-X, Z-Y-Z …

  2. Rotations about the static spatial frame or the moving body frame (intrinsic vs extrinsic rotation)

Example of intrinsic rotations (z,x',z'')



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

Gimbal Lock

- When using Euler angles θ_x, θ_y, θ_z, may reach a configuration where there is no way to rotate around one of the three axes!
- Recall rotation matrices around the three axes:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The product of these represents rotation by the three Euler angles.

$$R_x R_y R_z = \begin{bmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ -\cos\theta_x \cos\theta_z \sin\theta_y + \sin\theta_x \sin\theta_z & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{bmatrix}$$

Gimbal Lock

- Consider the special case where $\theta_y = \pi/2$ (so, $\cos(\theta_y)=0$, $\sin(\theta_y)=1$)

$$R_x R_y R_z = \begin{bmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ -\cos\theta_x \cos\theta_z \sin\theta_y + \sin\theta_x \sin\theta_z & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{bmatrix}$$

$$\implies \begin{bmatrix} 0 & 0 & 1 \\ \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_z & 0 \\ -\cos\theta_x \cos\theta_z + \sin\theta_x \sin\theta_z & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_z & 0 \end{bmatrix}$$

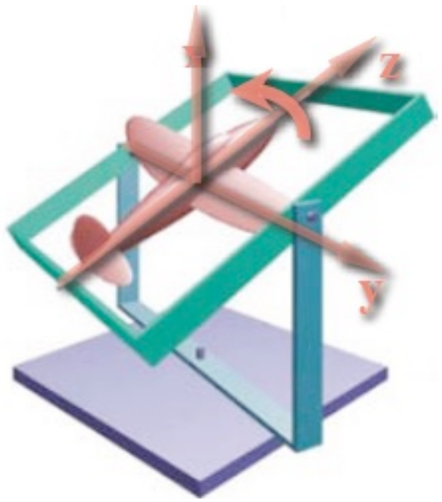
- We are left with a planar rotation. Notice it depends only of θ_x, θ_z. Not on θ_y.

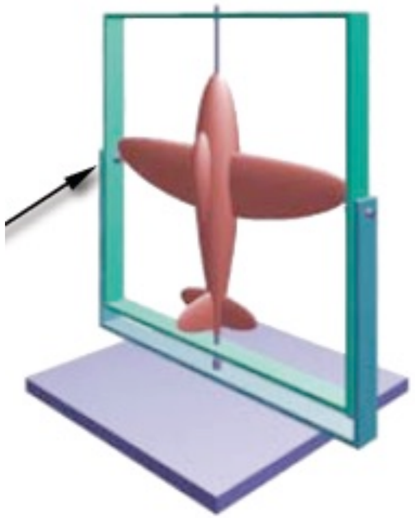
Euler Angles: Drawbacks

- Gimbal lock: When two of the axis align one degree of freedom is lost!

- Parameterization is not unique

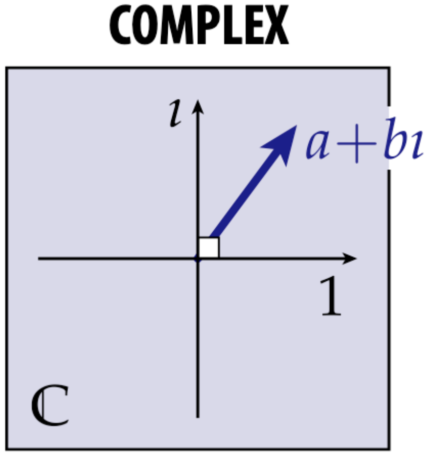
- Lots of conventions

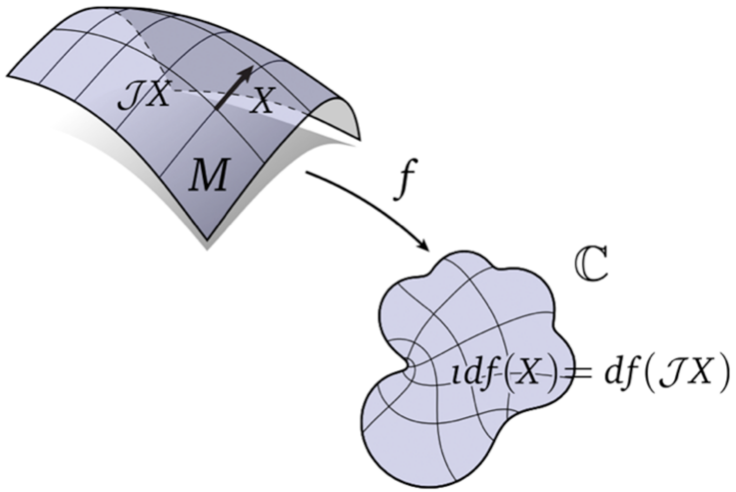




Complex Analysis - Motivation

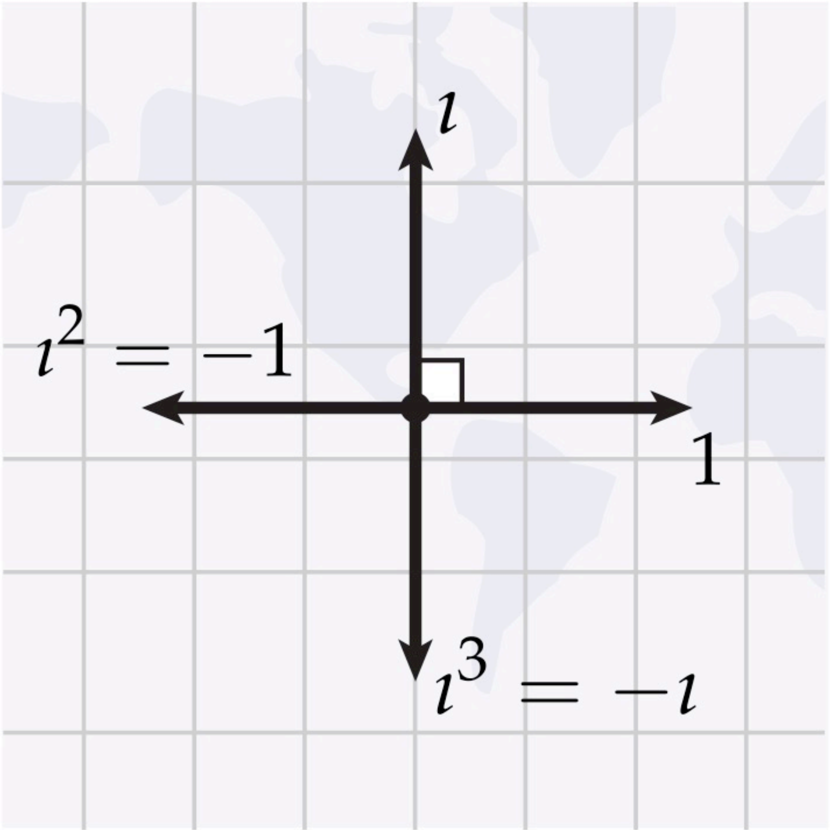
- Natural way to encode geometric transformations in 2D.
- Simplifies code/notation/debugging/thinking.
- Moderate reduction in computational cost/ bandwidth/storage.
- Fluency in complex analysis can lead to deeper/novel solutions to problems…





Truly: no good reason to use 2D vectors instead of complex numbers…

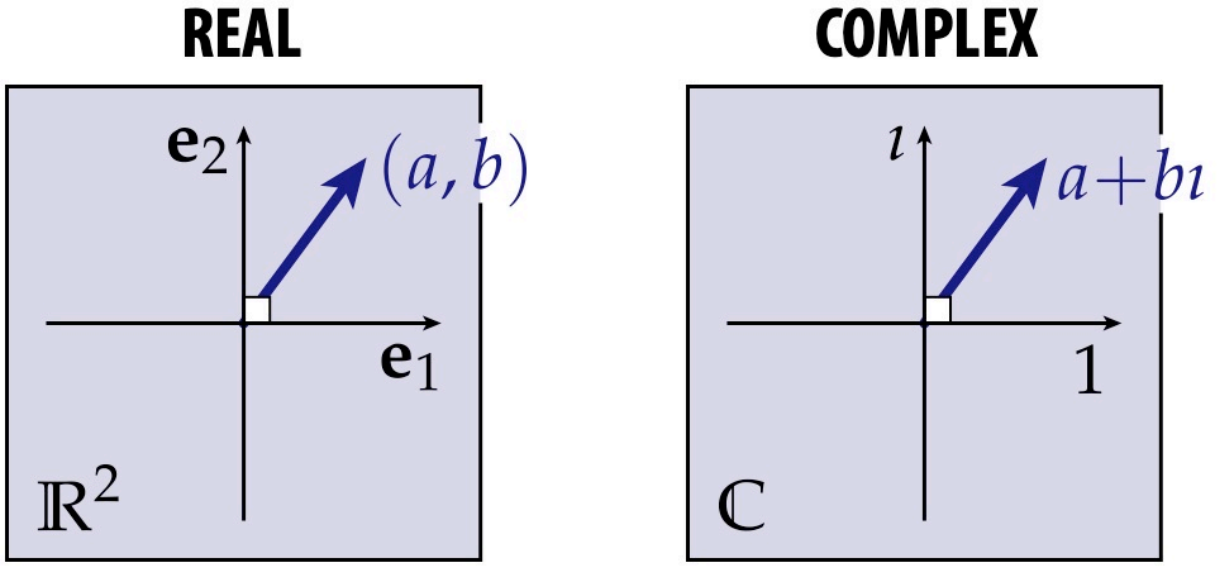
Imaginary units – Geometric description



Imaginary unit is just a quarter-turn in the counter-clockwise direction.

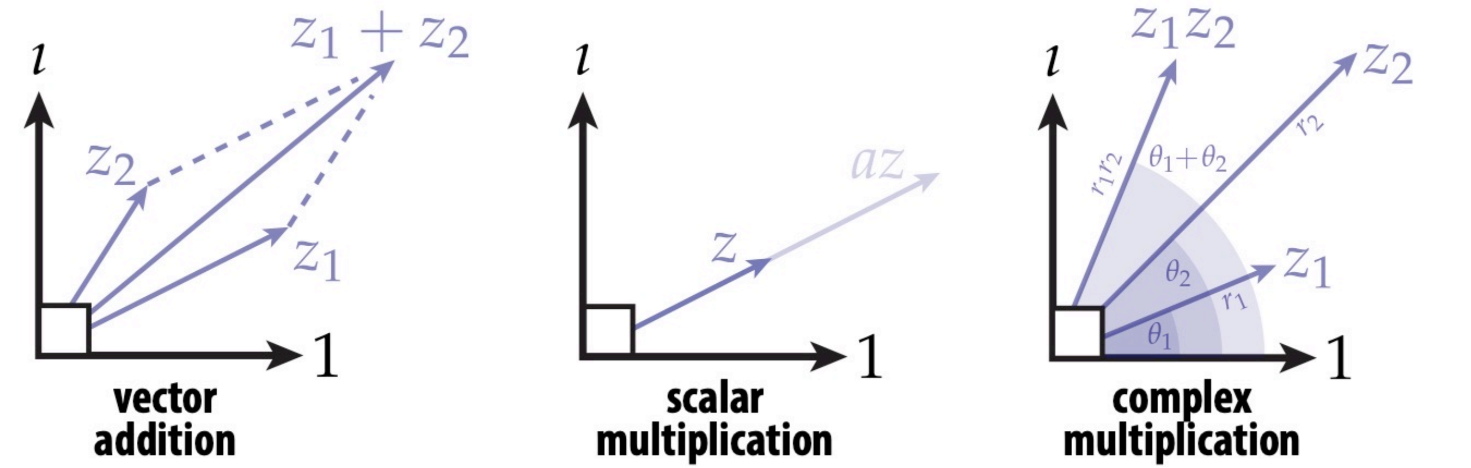
Complex Numbers

- Complex numbers are then just two vectors
- Instead of e_1, e_2 use "1" and "$\imath$" to denote two bases.
- Otherwise behaves like a 2D space
- … except that we are also going to get a very useful new notation of the *product* between the two vectors.



Complex Arithmetic

- Same operations as before, plus one more



- Complex multiplication:
  - Angles add
  - Magnitude multiplies

"POLAR FORM"*:

$$z_1 := (r_1, \theta_1)$$
$$z_2 := (r_2, \theta_2)$$
$$z_1 z_2 = (r_1 r_2, \theta_1 + \theta_2)$$

have to be more careful here!

***Not *quite* how it really works, but basic idea is right.**

Complex product – Rectangular form (1, $\imath$)

$$z_1 = (a + b\imath)$$

$$z_2 = (c + d\imath)$$

two quarter turns—
same as -1

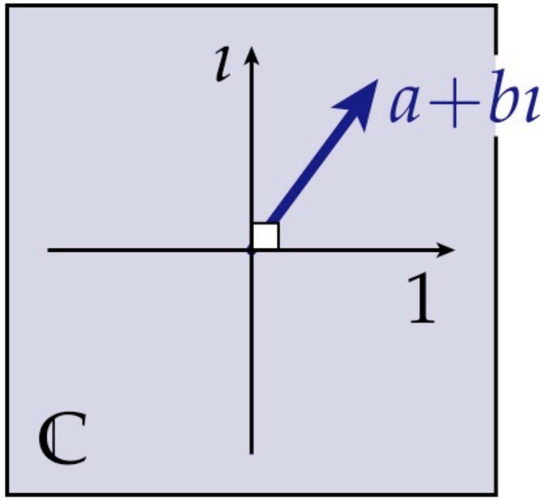
$$z_1 z_2 = ac + ad\imath + bc\imath + bd\imath^2 =$$

$$\boxed{(ac - bd) + (ad + bc)\imath.}$$

"real part" $\mathrm{Re}(z_1 z_2)$

"imaginary part" $\mathrm{Im}(z_1 z_2)$

- We used a lot of "rules" here. Can you justify them geometrically?
- Does this product agree with our geometric description (last slide)?



Complex product – Polar form

- Perhaps most beautiful identity in maths.

$$e^{\imath\pi} + 1 = 0$$

- Specialization of Euler's formula.

$$e^{\imath\theta} = \cos(\theta) + \imath\sin(\theta)$$

- Can use to implement complex product.

$$z_1 = ae^{\imath\theta}, \quad z_2 = be^{\imath\phi}$$

$$z_1 z_2 = abe^{\imath(\theta+\phi)}$$

(as with real exponentiation, exponents *add*)

Leonhard Euler (1707–1783)

2D rotations: Matrices vs. Complex

Suppose we want to rotate a vector u by an angle θ, then by an angle ϕ.

REAL / RECTANGULAR

$$\mathbf{u} = (x, y) \qquad \mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

$$\mathbf{Au} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

$$\mathbf{BAu} = \begin{bmatrix} (x\cos\theta - y\sin\theta)\cos\phi - (x\sin\theta + y\cos\theta)\sin\phi \\ (x\cos\theta - y\sin\theta)\sin\phi + (x\sin\theta + y\cos\theta)\cos\phi \end{bmatrix}$$

$$= \cdots \text{some trigonometry} \cdots =$$

$$\mathbf{BAu} = \begin{bmatrix} x\cos(\theta+\phi) - y\sin(\theta+\phi) \\ x\sin(\theta+\phi) + y\cos(\theta+\phi) \end{bmatrix}.$$

COMPLEX / POLAR

$$u = re^{\imath\alpha}$$

$$a = e^{\imath\theta}$$

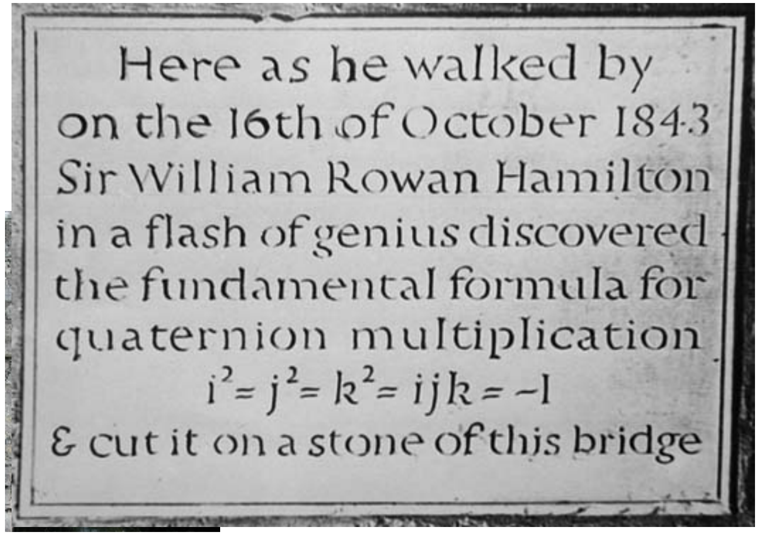
$$b = e^{\imath\phi}$$

$$abu = re^{\imath(\alpha+\theta+\phi)}$$

Quaternions generalize complex numbers

- TLDR: Kinda like complex numbers but for 3D rotations
- Weird situation: can't do 3D rotations w/ only 3 components!

William Rowan Hamilton
(1805-1865)



Quaternions

- A quaternion has 4 components:

$$\mathbf{q} = [q_w \ q_x \ q_y \ q_z]^T$$

- They generalize complex numbers

$$\mathbf{q} = q_w + q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$$

with additional properties: $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i} \cdot \mathbf{j} \cdot \mathbf{k} = -1$

- Unit length quaternions can be used to carry out rotations. The set they form is called S^3

Quaternions

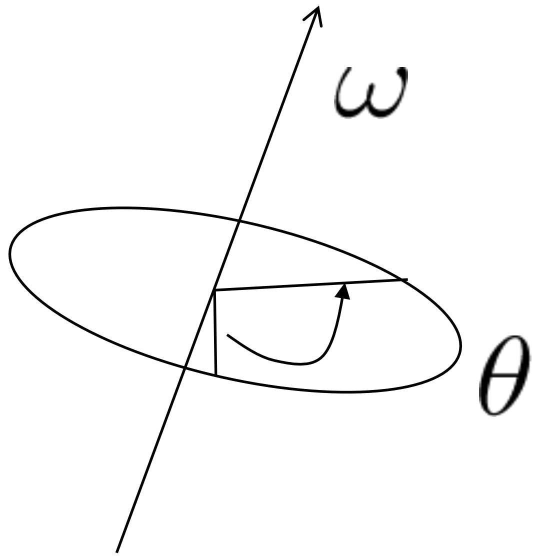
- Quaternions can also be interpreted as a **scalar** plus a **3-vector**

$$\mathbf{q} = [q_w \ \mathbf{v}]^T$$

- Where

$$q_w = \cos\frac{\theta}{2}$$

$$\mathbf{v} = \sin\frac{\theta}{2}\omega$$



- **Much easier to remember (and manipulate) than matrix!**

$$\begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z\sin\theta & u_x u_z(1-\cos\theta) + u_y\sin\theta \\ u_y u_x(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x\sin\theta \\ u_z u_x(1-\cos\theta) - u_y\sin\theta & u_z u_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

Quaternions

- Rotations can be carried away directly in parameter space via the quaternion product:
  - Concatenation of rotations:

$$\mathbf{q}_1 \circ \mathbf{q}_2 = \left(q_{w,1} q_{w,2} - \mathbf{v}_1 \cdot \mathbf{v}_2 \ , \ q_{w,1} \mathbf{v}_2 + q_{w,2} \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2\right)$$

- If we want to rotate a vector $\boldsymbol{a}$

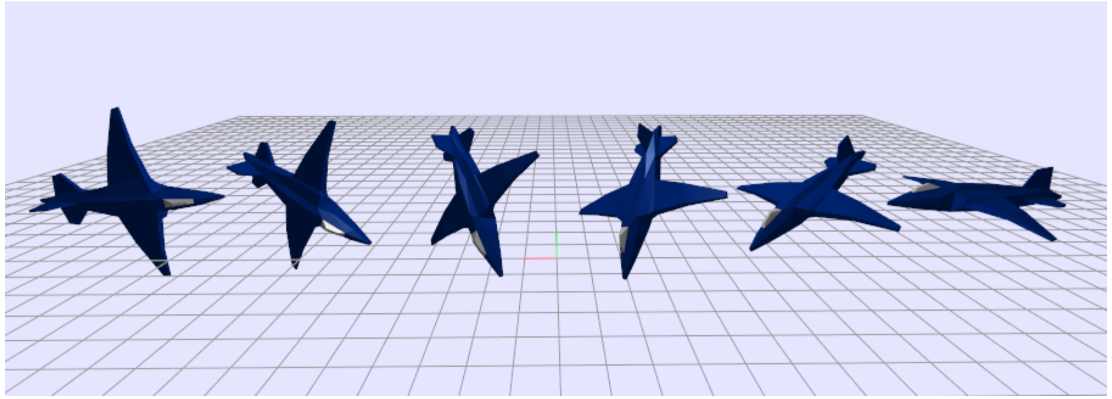
$$\boldsymbol{a}' = Rotate(\boldsymbol{a}) = \mathbf{q} \circ \tilde{\boldsymbol{a}} \circ \bar{\mathbf{q}}$$

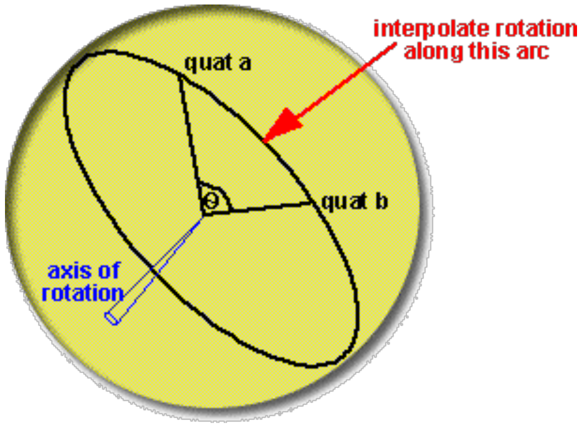
where $\bar{\mathbf{q}} = (q_w \ - \mathbf{v})$ is the quat conjugate.

Quaternions are ideal for interpolation

- Interpolating Euler angles can yield strange-looking paths, non-uniform rotation speed, …
- Simple solution with quaternions: "SLERP" (spherical linear interpolation):

$$\mathrm{Slerp}(q_0, q_1, t) = q_0 (q_0^{-1} q_1)^t, \quad t \in [0,1]$$





Quaternions

- ✓ Quaternions have no singularities
- ✓ Derivatives exist and are linearly independent
- ✓ Quaternion product allows to perform rotations
- ✓ Good for interpolation
- ✗ But all this comes at the expense of using 4 numbers instead of 3
- ✗ Enforce quadratic constraint $||\mathbf{q}||_2 = 1$

Axis-angle

For human motion modeling it is often needed to specify the axis of rotation of a joint

Any rotation about the origin can be expressed in terms of the axis of rotation $\omega \in \mathbb{R}^3$ and the angle of rotation θ with the **exponential map**

$$\boxed{\mathbf{R} = \exp(\theta \widehat{\omega})}$$

Lie Groups / Lie Algebras

Definition: A group is an *n*-dimensional *Lie-group*, if the set of its elements can be represented as a continuously differentiable manifold of dimension *n*, on which the group product and inverse are continuously differentiable functions as well

$$\begin{array}{c}\text{Lie Group}\\ M\end{array} \underset{\exp(\theta\xi)}{\overset{\theta\partial M\big|_0}{\rightleftarrows}} \begin{array}{c}\text{Lie algebra}\\ \theta\xi\end{array}$$

Axis-angle

- Given a vector ω the **skew symetric** matrix is

$$\theta\widehat{\omega} = \theta \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

You will also find it as $\omega_\times$

- It is the matrix form of the cross-product:

$$\omega \times \mathbf{p} = \hat{\omega}\,\mathbf{p}$$

Exponential map

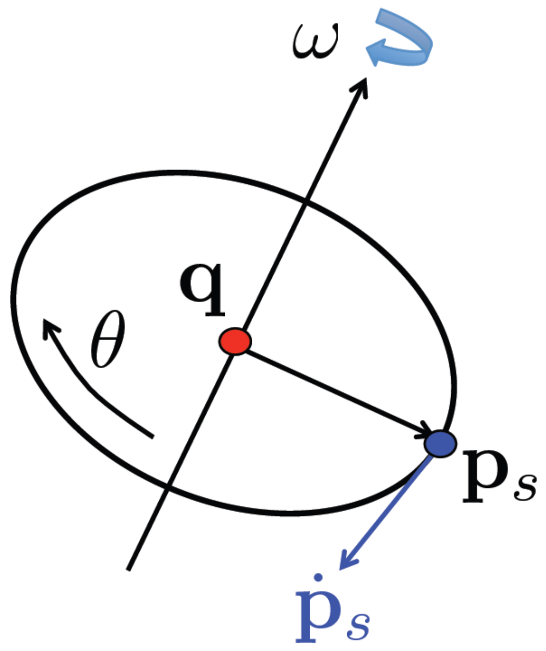
- The exponential map recovers the rotation matrix from the axis-angle representation (Lie-algebra)

$$\mathbf{R}(\theta, \omega) = \exp(\theta \widehat{\omega})$$

Exponential map

Proof: exponential map

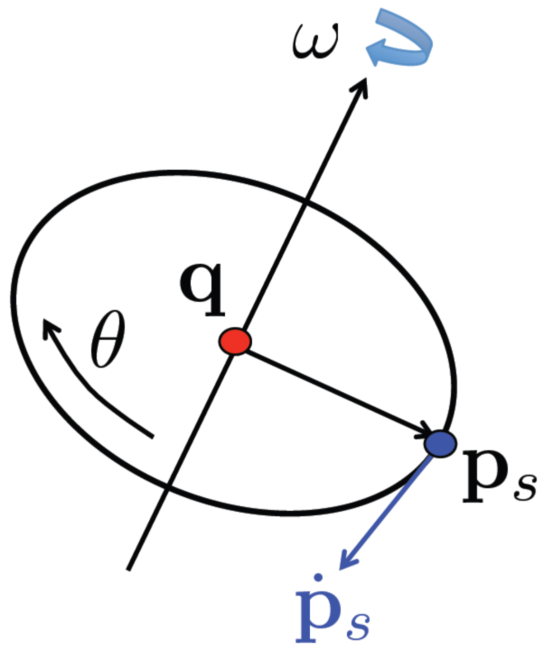
$$\dot{\mathbf{p}}(t) = \ ?$$



Exponential map

Proof: exponential map

$$\dot{\mathbf{p}}(t) = \boldsymbol{\omega} \times \mathbf{p}(t) = \widehat{\boldsymbol{\omega}}\mathbf{p}(t)$$



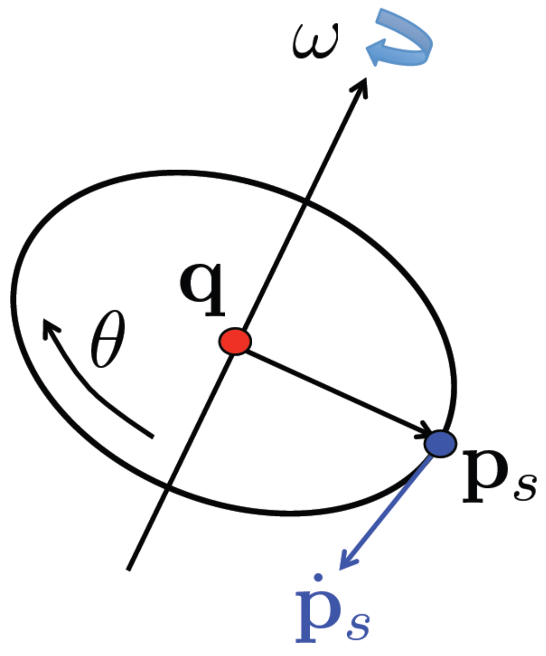
Exponential map

Proof: exponential map

$$\dot{\mathbf{p}}(t) = \boldsymbol{\omega} \times \mathbf{p}(t) = \widehat{\boldsymbol{\omega}}\mathbf{p}(t)$$

$$\Downarrow$$

$$\mathbf{p}(t) = \exp(\widehat{\boldsymbol{\omega}}t)\mathbf{p}(0)$$



Exponential map

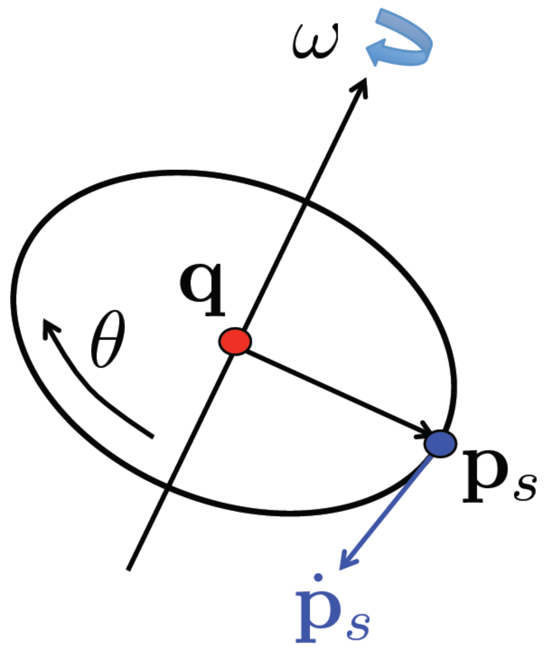
Proof: exponential map

$$\dot{\mathbf{p}}(t) = \boldsymbol{\omega} \times \mathbf{p}(t) = \widehat{\boldsymbol{\omega}}\mathbf{p}(t)$$

$$\mathbf{p}(t) = \exp(\widehat{\boldsymbol{\omega}}t)\mathbf{p}(0)$$

If we rotate θ units of time

$$\boxed{\mathbf{R}(\boldsymbol{\theta}, \boldsymbol{\omega}) = \exp(\boldsymbol{\theta}\,\widehat{\boldsymbol{\omega}})}$$



Exponential map

$$\exp(\theta\widehat{\omega}) = e^{(\theta\widehat{\omega})} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \dots$$

Exploiting the properties of skew symetric matrices

Rodriguez formula:

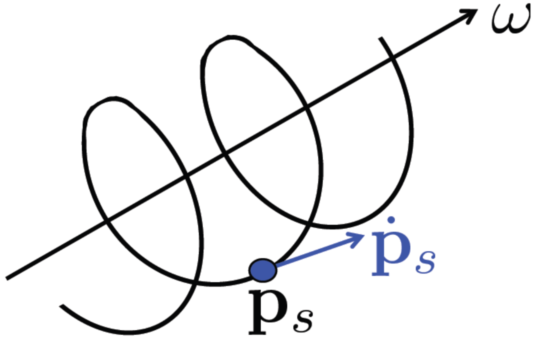
$$\boxed{\exp(\theta\widehat{\omega}) = I + \widehat{\omega}\sin(\theta) + \widehat{\omega}^2(1 - \cos(\theta))}$$

Closed form!

Twists

- What about translation ?
- The **twist coordinates** are defined as

$$\theta\xi = \theta(v_1, v_2, v_3, \omega_1, \omega_2, \omega_3)$$



- And the **twist** is defined as

$$[\theta\xi]^\wedge = \theta\hat{\xi} = \theta \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \boxed{\dot{\mathbf{p}} = \hat{\xi}\mathbf{p}}$$

Exponential map

- The rigid body motion can be computed in closed form as well

$$\mathbf{G}(\theta,\omega) = \begin{bmatrix} \mathbf{R}_{3\times 3} & \mathbf{t}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \exp(\theta\widehat{\xi})$$

- From the following formula

$$\exp(\theta\widehat{\xi}) = \begin{bmatrix} \exp(\theta\widehat{\omega}) & (I - \exp(\theta\widehat{\omega}))(\omega \times v + \omega\omega^T v\theta) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

Which representation should I use?

Number of parameters	Singularities	Human constraints	Concatenate motion	Optimization (derivatives)
Twists	Quaternions	Twists	Quaternions	Twists
Euler Angles	Twists	Quaternions	Twists	Euler Angles
Quaternions	Euler Angles	Euler Angles	Euler Angles	Quaternions

Ingredients to build a Virtual Human

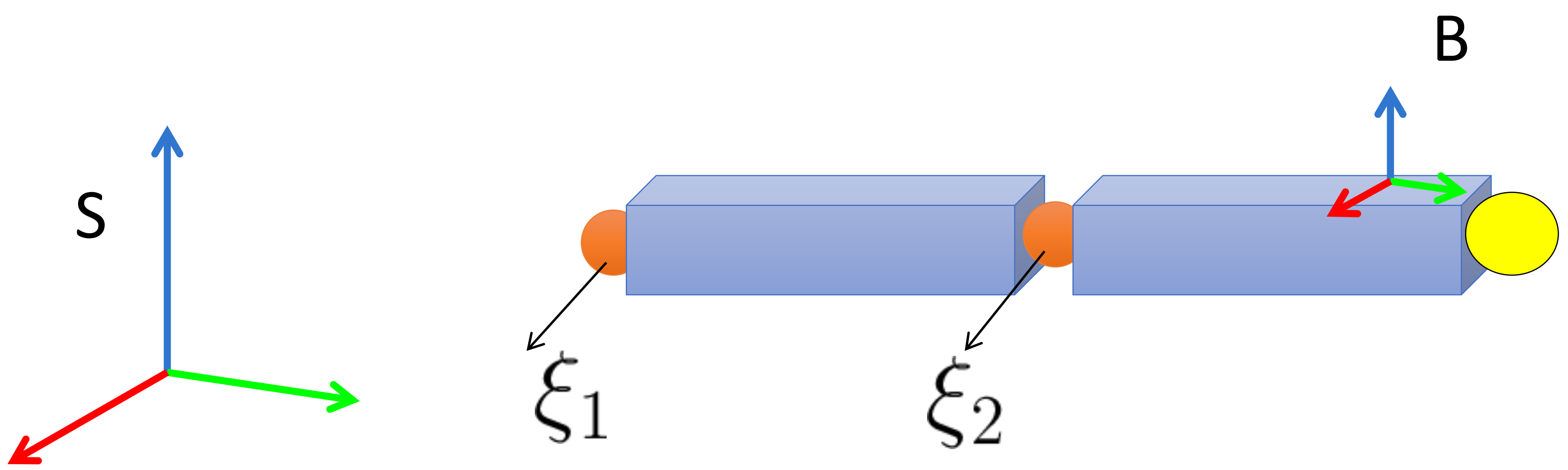
Building a human model

- Kinematic parameterization
  - Rotation Matrices
  - Euler Angles
  - Quaternions
  - Twists and Exponential maps
  - **Kinematic chains**
- Subject shape model
  - Geometric primitives
  - Detailed Body Scans
  - Human Shape models

Fitting model to observations

- Inference
  - Observation likelihood
  - Local optimization
  - Particle Based optimization
  - Directly regressing parameters

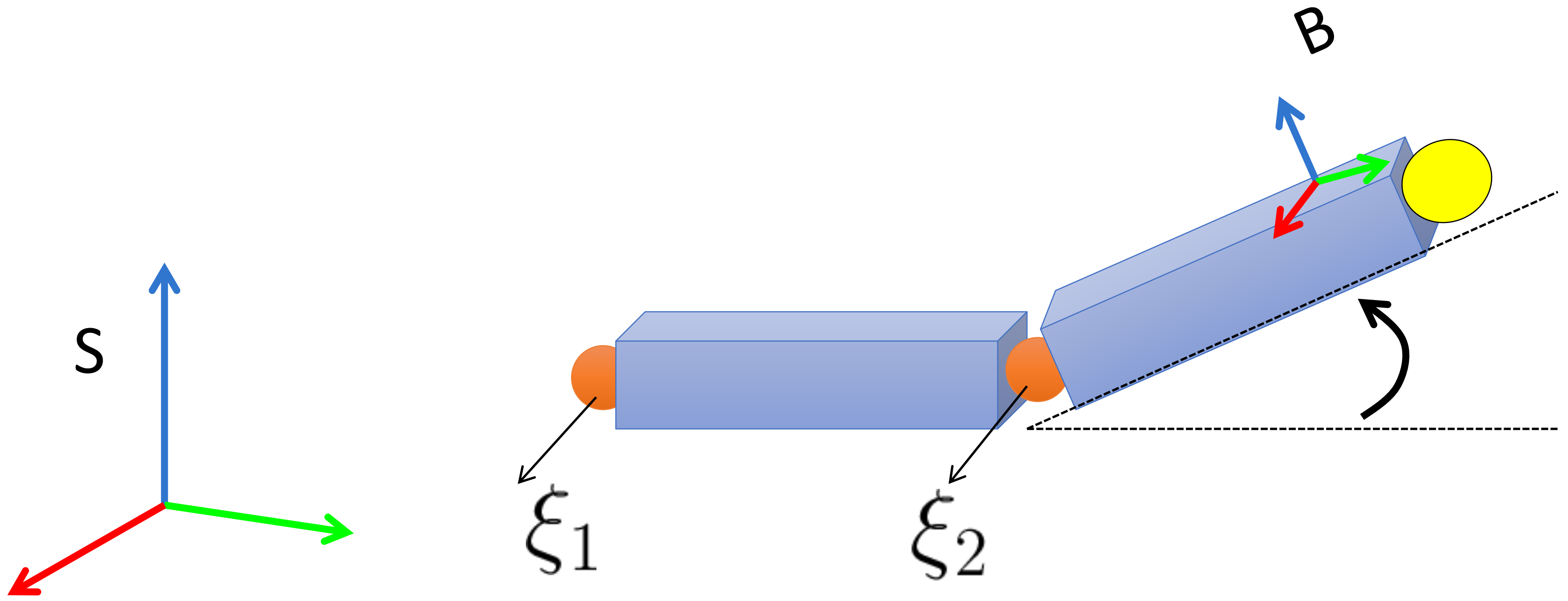
Articulation



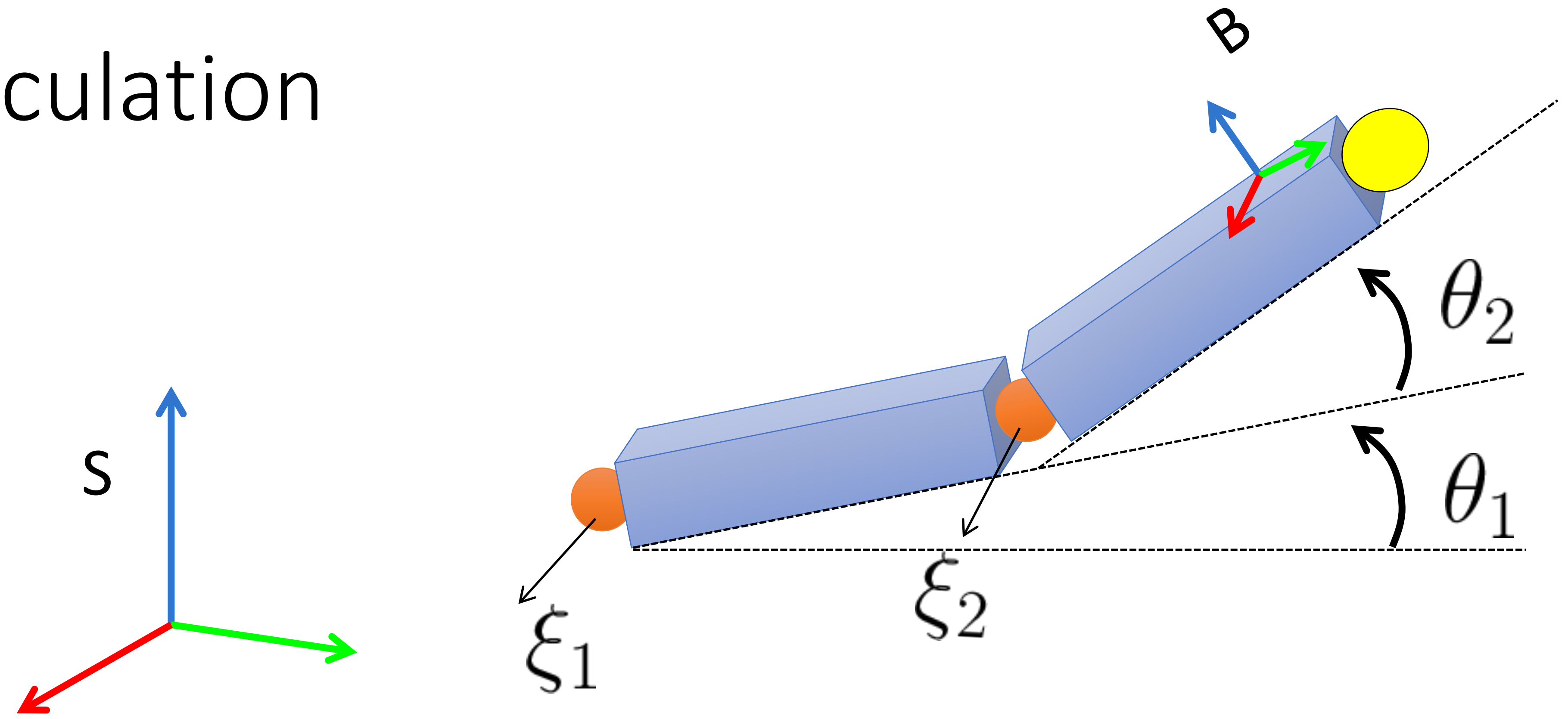
In a rest position we have

$$\mathbf{p}_s(0) = \mathbf{G}_{sb}\mathbf{p}_b$$

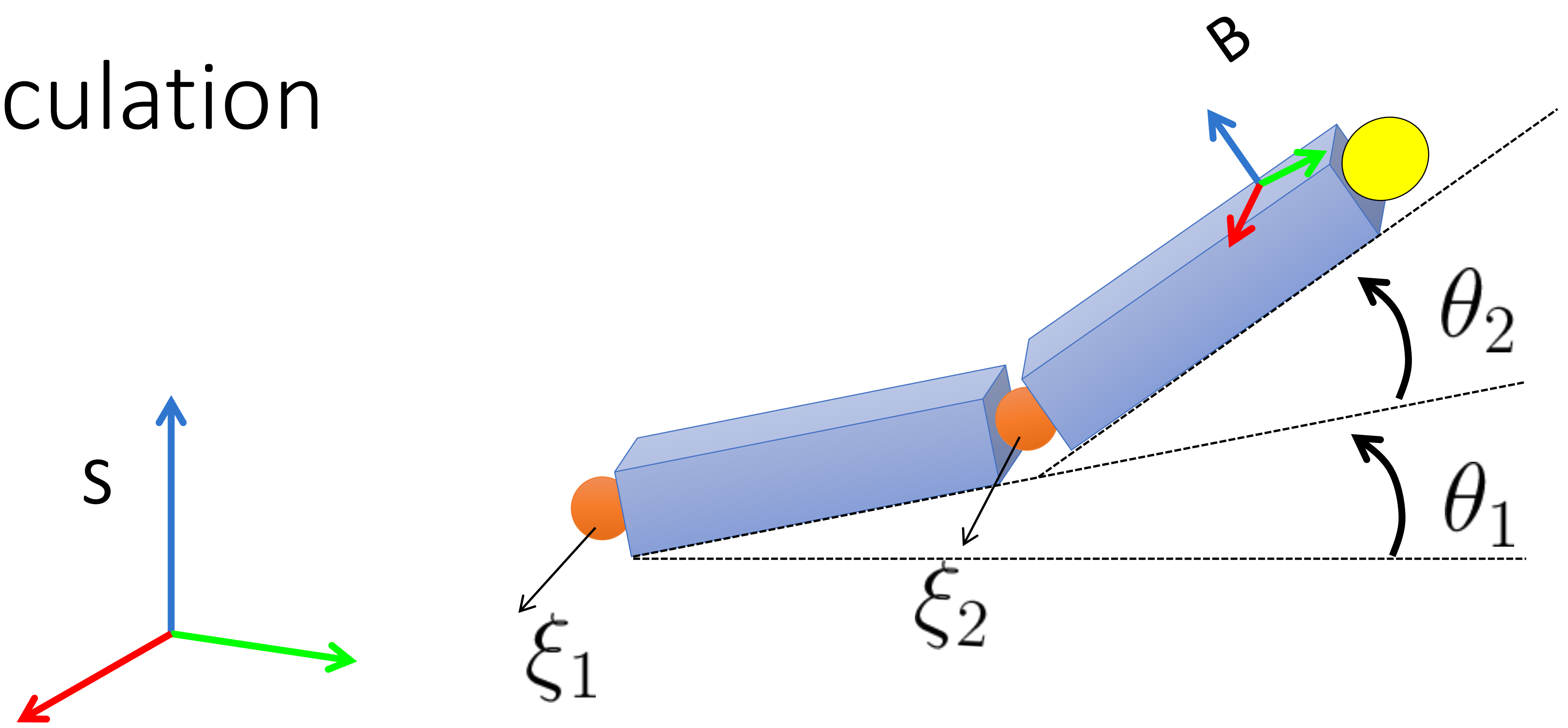
Articulation



Articulation



Articulation



The coordinates of the point in the spatial frame

$$\bar{\mathbf{p}}_s = \mathbf{G}_{sb}(\theta_1, \theta_2) = e^{\widehat{\xi}_1 \theta_1} e^{\widehat{\xi}_2 \theta_2} \mathbf{G}_{sb}(\mathbf{0}) \bar{\mathbf{p}}_b$$

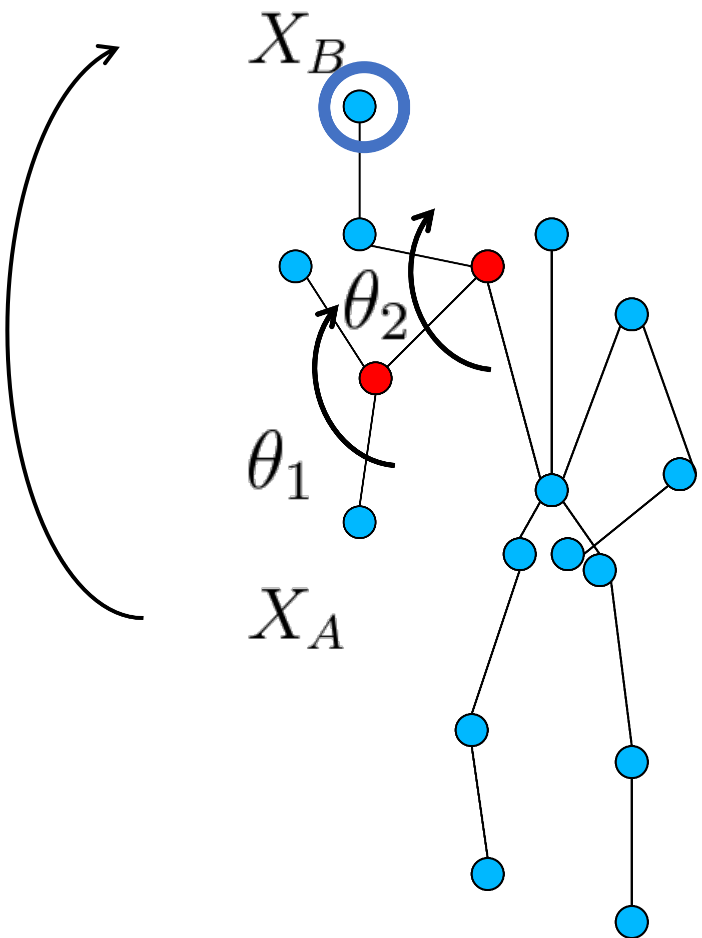
Product of exponentials

$$\boxed{\mathbf{G}_{sb}(\Theta) = e^{\widehat{\xi}_1 \theta_1} e^{\widehat{\xi}_2 \theta_2} \ldots e^{\widehat{\xi}_n \theta_n} \mathbf{G}_{sb}(\mathbf{0})}$$

- $\mathbf{G}_{sb}(\Theta)$ is the mapping from coordinate B to coordiante S
- BUT $\exp(\theta_i \widehat{\xi}_i)$ IS NOT the mapping from segment i+1 to segment i.
- Think of $\exp(\theta_i \widehat{\xi}_i)$ simply as the relative motion of that joint.

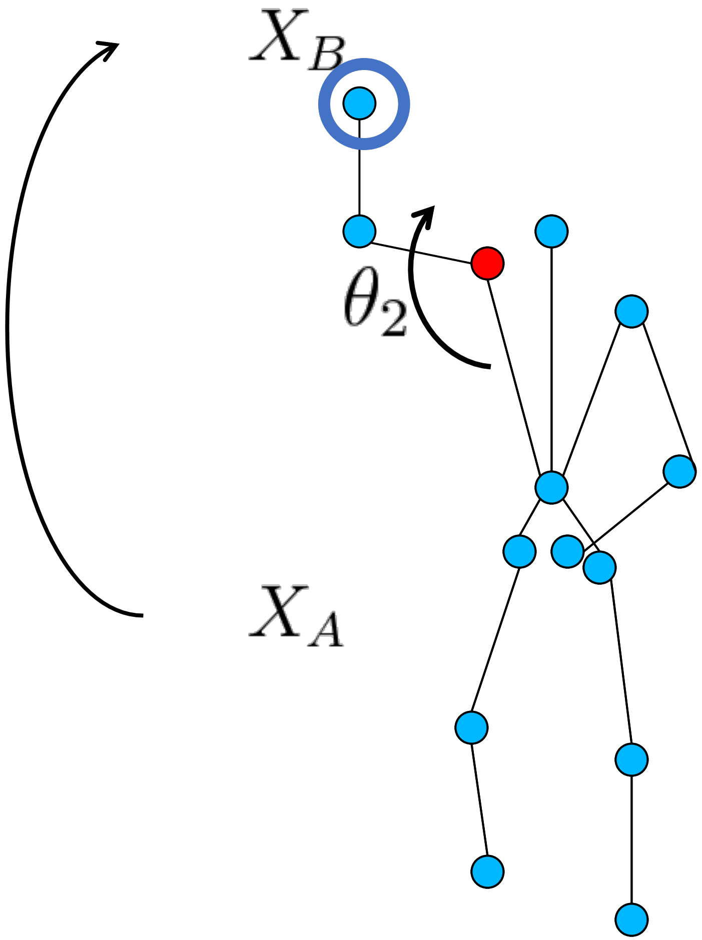
Inverse Kinematics

Supose we want to find the angles to reach a specific goal



Inverse Kinematics

Supose we want to find the angles to reach a specific goal



$$\arg\min_{\theta_1 \ldots \theta_n} \| \exp(\theta_1 \widehat{\xi}_1) \ldots \exp(\theta_n \widehat{\xi}_n) \mathbf{X}_A - \mathbf{X}_B \|^2$$

- The problem is non-linear

- Linearize with the articulated **Jacobian**

Articulated Jacobian

The **Jacobian** using twists is extremely simple and easy to compute

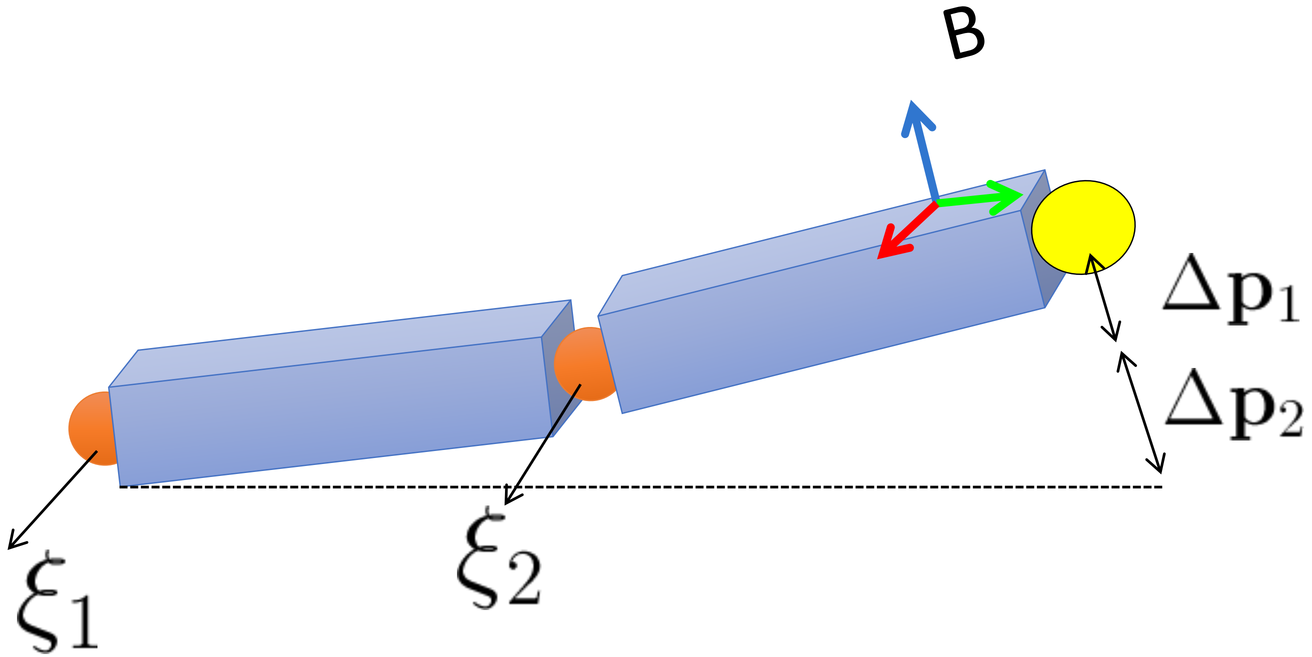
$$\boxed{\mathbf{J}_\Theta = [\xi_1 \quad \xi_2' \quad \dots \quad \xi_n']}$$

1) Every column corresponds to the contribution of i-th joint to the end-effector motion
2) Maps an increment of joint angles to the end-effector twist

$$\mathbf{J}_\Theta \Delta\Theta = \xi_T$$

Articulated Jacobian

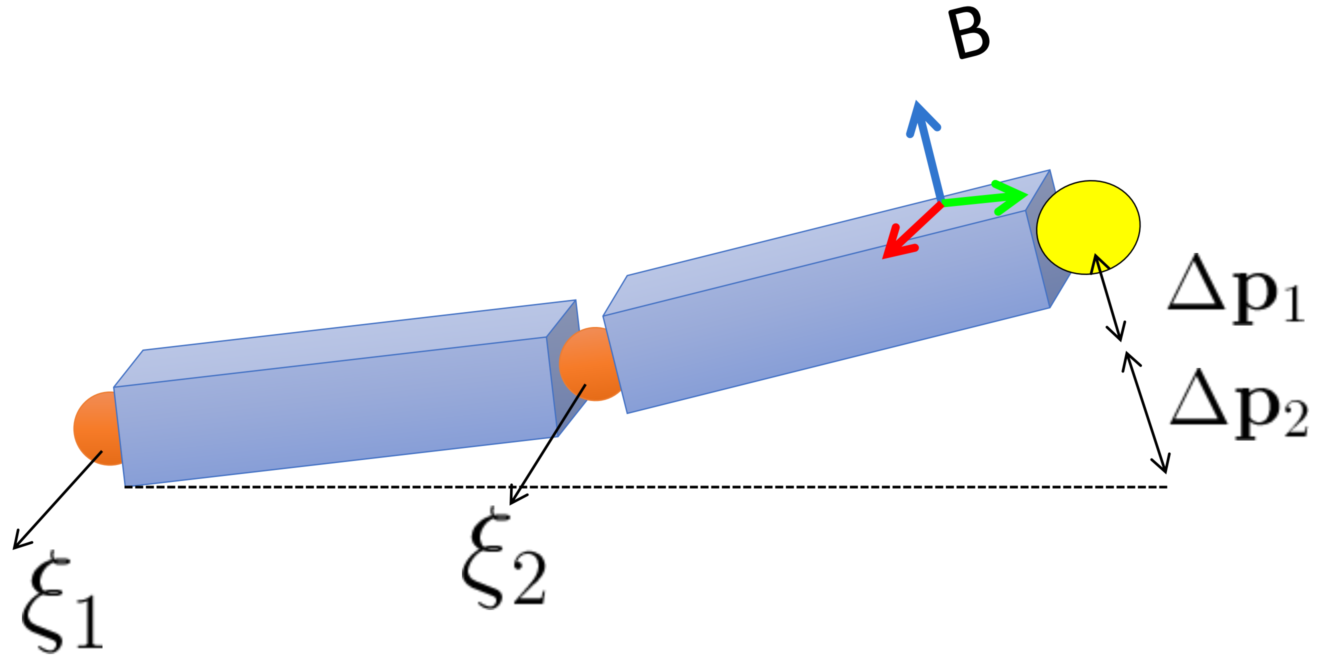
Intuition: Linear combination of twists



$$\Delta \bar{\mathbf{p}}_s = [\ \mathbf{J}_\Theta \ \cdot \Delta\Theta\ ]^\wedge\ \bar{\mathbf{p}}_s = [\xi_1 \Delta\theta_1 + \xi_2' \Delta\theta_2 + \ldots + \xi_n' \Delta\theta_n]^\wedge \bar{\mathbf{p}}_s$$

Articulated Jacobian

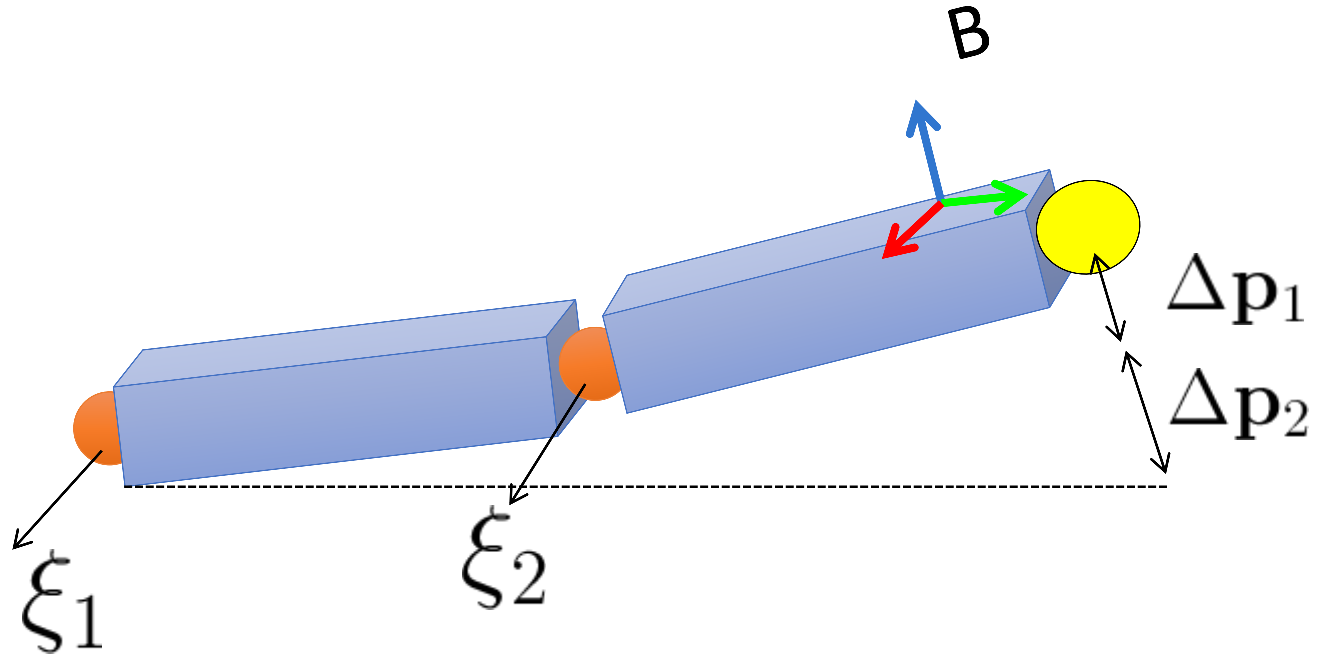
Intuition: Linear combination of twists



$$\Delta \bar{\mathbf{p}}_s = [\, \mathbf{J}_\Theta \;\cdot \Delta\Theta \,]^\wedge \, \bar{\mathbf{p}}_s = [\xi_1 \Delta\theta_1 + \boxed{\xi_2' \Delta\theta_2} + \ldots + \xi_n' \Delta\theta_n]^\wedge \bar{\mathbf{p}}_s$$

Articulated Jacobian

Intuition: Linear combination of twists

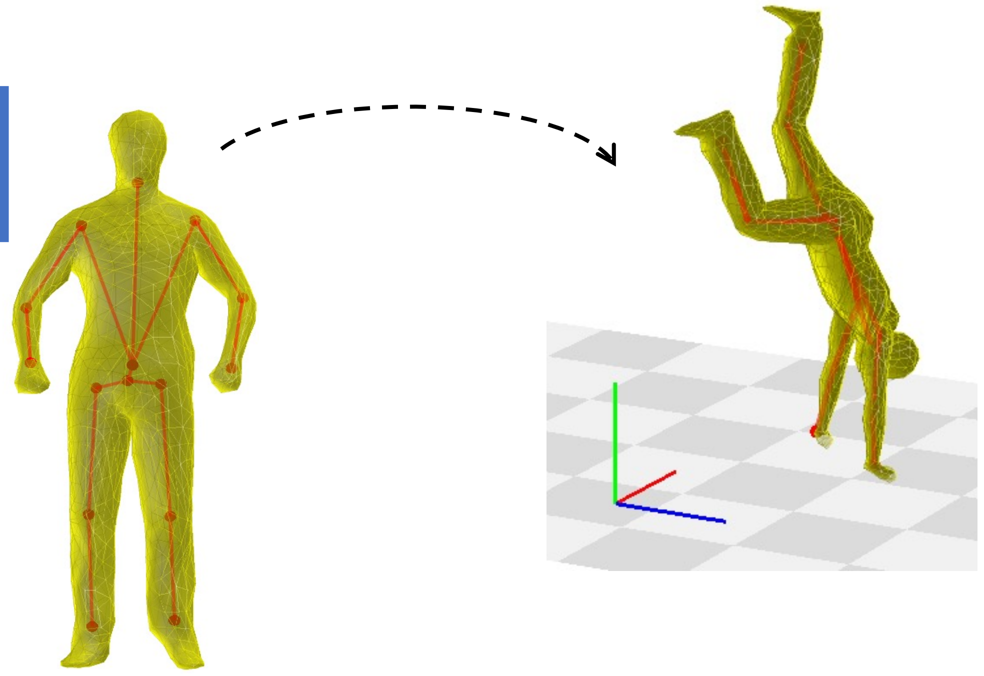


$$\Delta \bar{\mathbf{p}}_s = [\, \mathbf{J}_\Theta \cdot \Delta\Theta \,]^\wedge \, \bar{\mathbf{p}}_s = [\boxed{\xi_1 \Delta\theta_1} + \xi_2' \Delta\theta_2 + \ldots + \xi_n' \Delta\theta_n]^\wedge \bar{\mathbf{p}}_s$$

Pose Parameters

Pose parameters: root + joint angles

$$\boxed{\mathbf{x}_t = (\xi, \theta_1 \ldots \theta_n)}$$

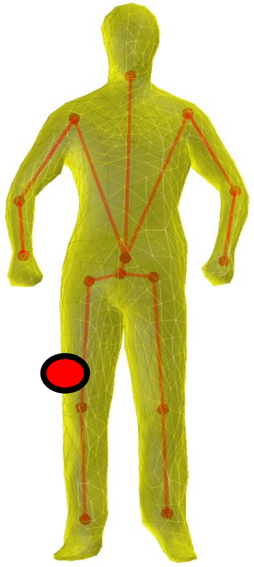


Pose Jacobian

Maps increments in the pose parameters to increments in end-effector position

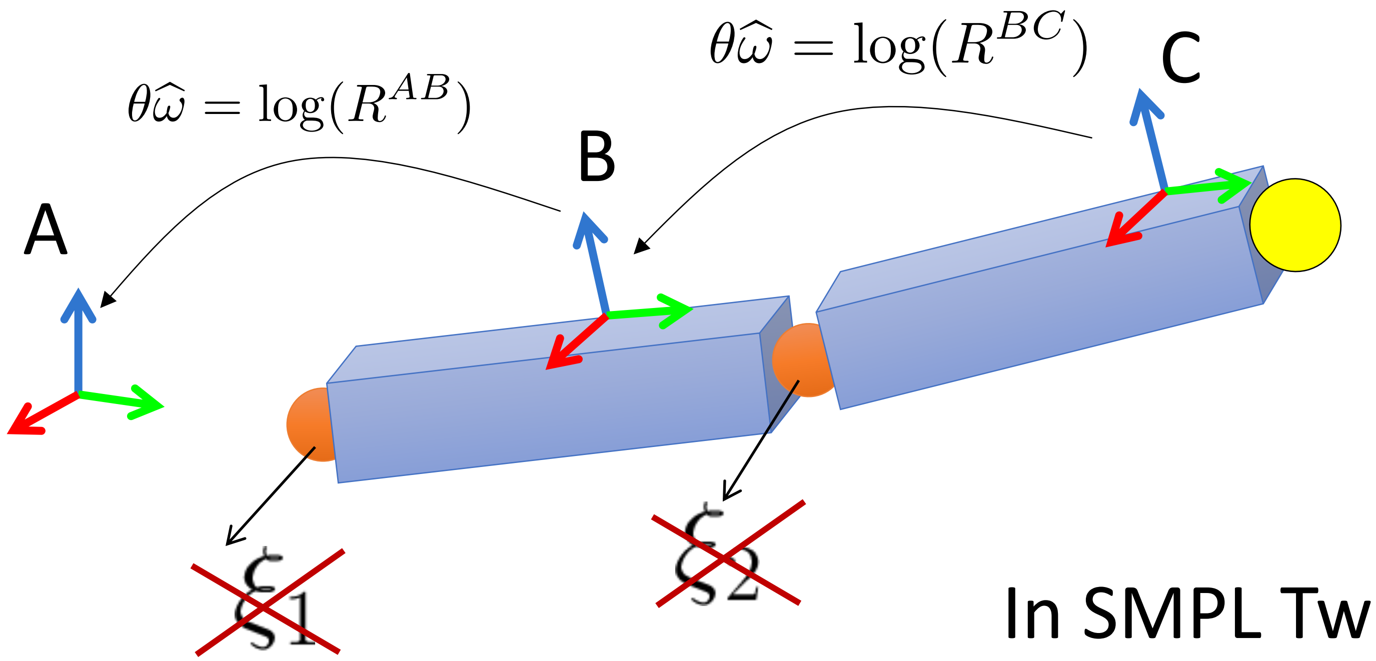
$$\boxed{\mathbf{J}_{\mathbf{x}} : \Delta\mathbf{x} \mapsto \Delta\mathbf{p}_s}$$

$$\mathbf{J}_{\mathbf{x}}(\mathbf{p}_s) = \left[\underbrace{\mathbf{I}_{[3\times3]} \quad -\mathbf{p}_s^{\wedge}}_{\text{6 columns of Root}} \quad \underbrace{\widehat{\xi}_1\,\bar{\mathbf{p}}_s \quad \widehat{\xi}'_2\,\bar{\mathbf{p}}_s \quad \ldots \quad \widehat{\xi}'_n\,\bar{\mathbf{p}}_s}_{\text{N columns for one per joint}} \right]$$



In SMPL (de-facto body model)

In SMPL rotations are local, from child to parent



In SMPL Twists are in the coordinates of the parent joint!

Slide credits and further reading

- Keenan Crane – Computer Graphics (slides on quaternions). CMU computer graphics lecture
- Pons-Moll & Rosehnan – ICCV’2011 Tutorial on Model Based Pose Estimation
  - Book chapter: model based human pose estimation available on pdf on my website.
- A Mathematical Introduction to Robotic Manipulation
  - excellent rigorous treatment of twists and exponential maps for articulated bodies

Slides below are originals for heavy editing

Ingredients to build a Virtual Human

1) Kinematic parameterization

- Rotation Matrices
- Euler Angles
- Quaternions
- Twists and Exponential maps
- Kinematic chains

2) Subject model

- Geometric primitives
- Detailed Body Scans
- Human Shape models

3) Inference

- Observation likelihood
- Local optimization
- Particle Based optimization

Ingredients to build a Virtual Human

1) Kinematic parameterization

- **Rotation Matrices**
- Euler Angles
- Quaternions
- Twists and Exponential maps
- Kinematic chains

2) Subject model

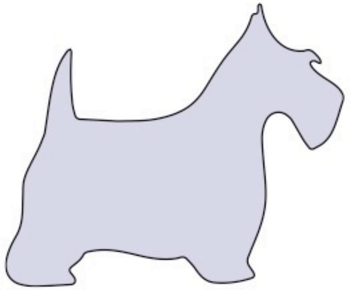
- Geometric primitives
- Detailed Body Scans
- Human Shape models

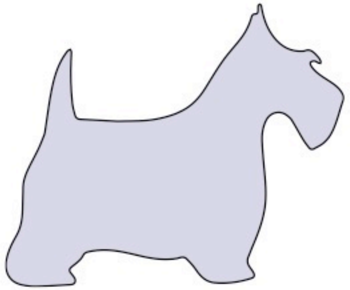
3) Inference

- Observation likelihood
- Local optimization
- Particle Based optimization

Commutativity of Rotations—2D

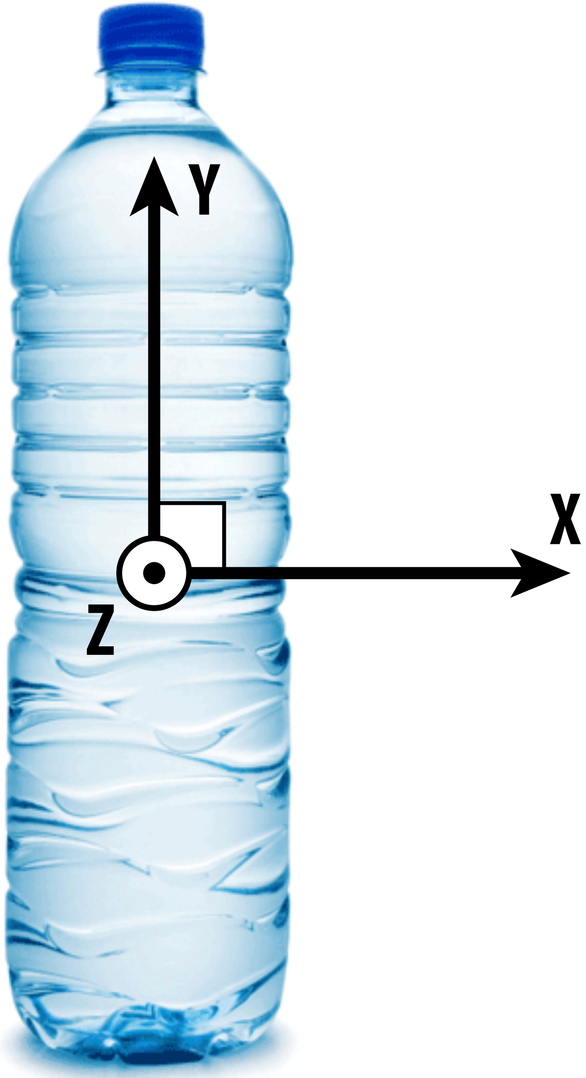
- **In 2D, order of rotations doesn't matter:**





Commutativity of Rotations—3D

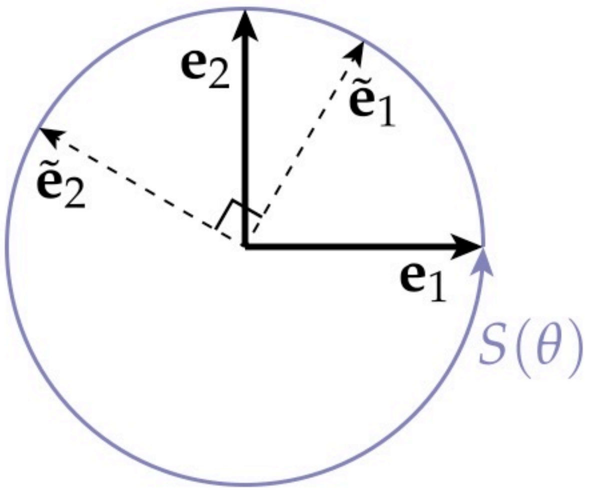
- What about in 3D?
- Try it at home—grab a water bottle!
  - Rotate 90° around Y, then 90° around Z, then 90° around X
  - Rotate 90° around Z, then 90° around Y, then 90° around X
  - (Was there any difference?)



CONCLUSION: bad things can happen if we're not careful about the order in which we apply rotations!

Representing Rotations—2D

- **First things first: how do we get a rotation matrix in 2D? (Don't just regurgitate the formula!)**
- **Suppose I have a function S(θ) that for a given angle θ gives me the point (x,y) around a circle (CCW).**
    - **Right now, I *do not care how this function is expressed!*****
- **What's e1 rotated by θ?**
- **What's e2 rotated by θ?**
- **How about** $\mathbf{u} := a\mathbf{e}_1 + b\mathbf{e}_2$ **?**



What then must the matrix look like?

***I.e., I don't yet care about sines and cosines and so forth.**

Gimbal Lock

- **When using Euler angles θ_x, θ_y, θ_z, may reach a configuration where there is *no way to rotate around one of the three axes!***
- **Recall rotation matrices around three axes:**

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \quad R_y = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \quad R_z = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- **Product of these matrices represents rotation by Euler angles:**

$$R_x R_y R_z = \begin{bmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \cos\theta_z \sin\theta_x \sin\theta_y + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_y \sin\theta_z & -\cos\theta_y \sin\theta_x \\ -\cos\theta_x \cos\theta_z \sin\theta_y + \sin\theta_x \sin\theta_z & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_y \sin\theta_z & \cos\theta_x \cos\theta_y \end{bmatrix}$$

- **Consider special case $\theta_y = \pi/2$ (so, $\cos\theta_y = 0$, $\sin\theta_y = 1$):**

$$\Longrightarrow \begin{bmatrix} 0 & 0 & 1 \\ \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_z & \cos\theta_x \cos\theta_z - \sin\theta_x \sin\theta_z & 0 \\ -\cos\theta_x \cos\theta_z + \sin\theta_x \sin\theta_z & \cos\theta_z \sin\theta_x + \cos\theta_x \sin\theta_z & 0 \end{bmatrix}$$

Product of exponentials

Product of exponentials formula

$$\boxed{\mathbf{G}_{sb}(\Theta) = e^{\widehat{\xi}_1 \theta_1} e^{\widehat{\xi}_2 \theta_2} \ldots e^{\widehat{\xi}_n \theta_n} \mathbf{G}_{sb}(\mathbf{0})}$$

$\mathbf{G}_{sb}(\Theta)$ is the mapping from coordinate B to coordiante S

BUT $\exp(\theta_i \widehat{\xi}_i)$ IS NOT the mapping from segment i+1 to segment i.

Think of $\exp(\theta_i \widehat{\xi}_i)$ simply as the relative motion of that joint

Overview

1) Kinematic parameterization

- Rotation Matrices
- Euler Angles
- Quaternions
- Twists and Exponential maps
- **Kinematic chains**

2) Subject model

- Geometric primitives
- Detailed Body Scans
- Human Shape models

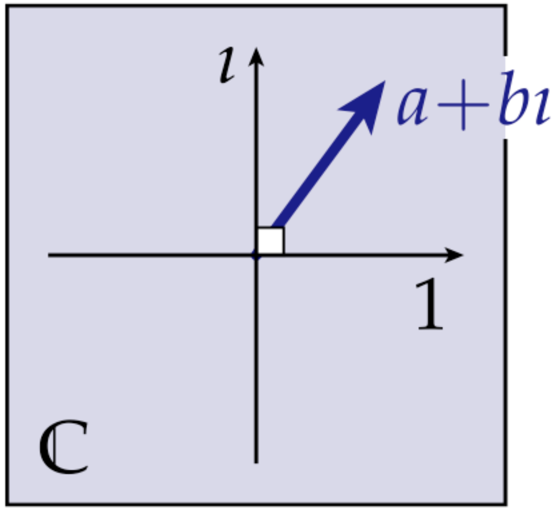
3) Inference

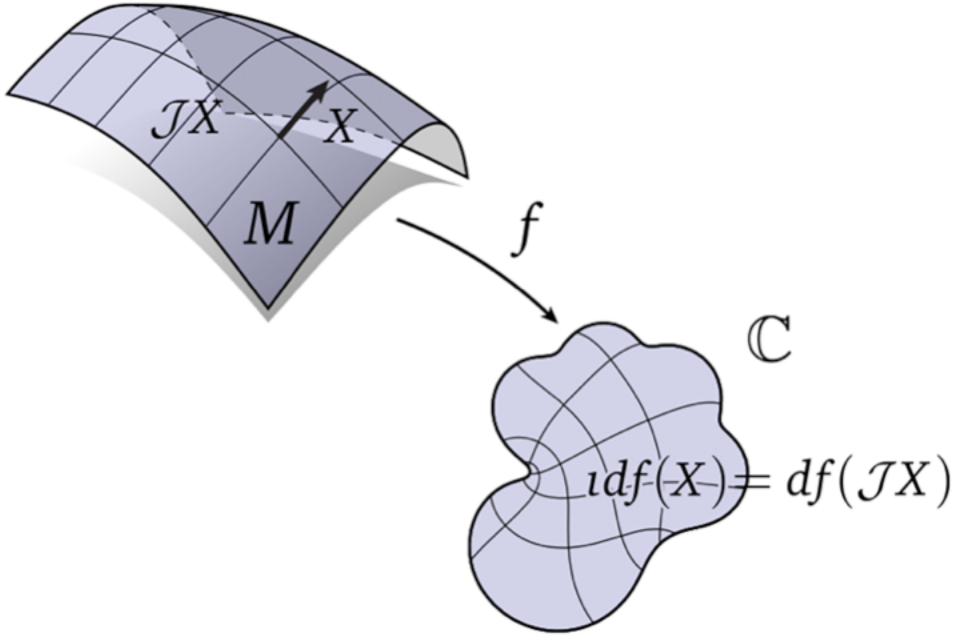
- Observation likelihood
- Local optimization
- Particle Based optimization

Complex Analysis—Motivation

- Natural way to encode geometric transformations in 2D
- Simplifies code / notation / debugging / *thinking*
- *Moderate* reduction in computational cost/bandwidth/storage
- Fluency with complex analysis can lead into deeper/novel solutions to problems…

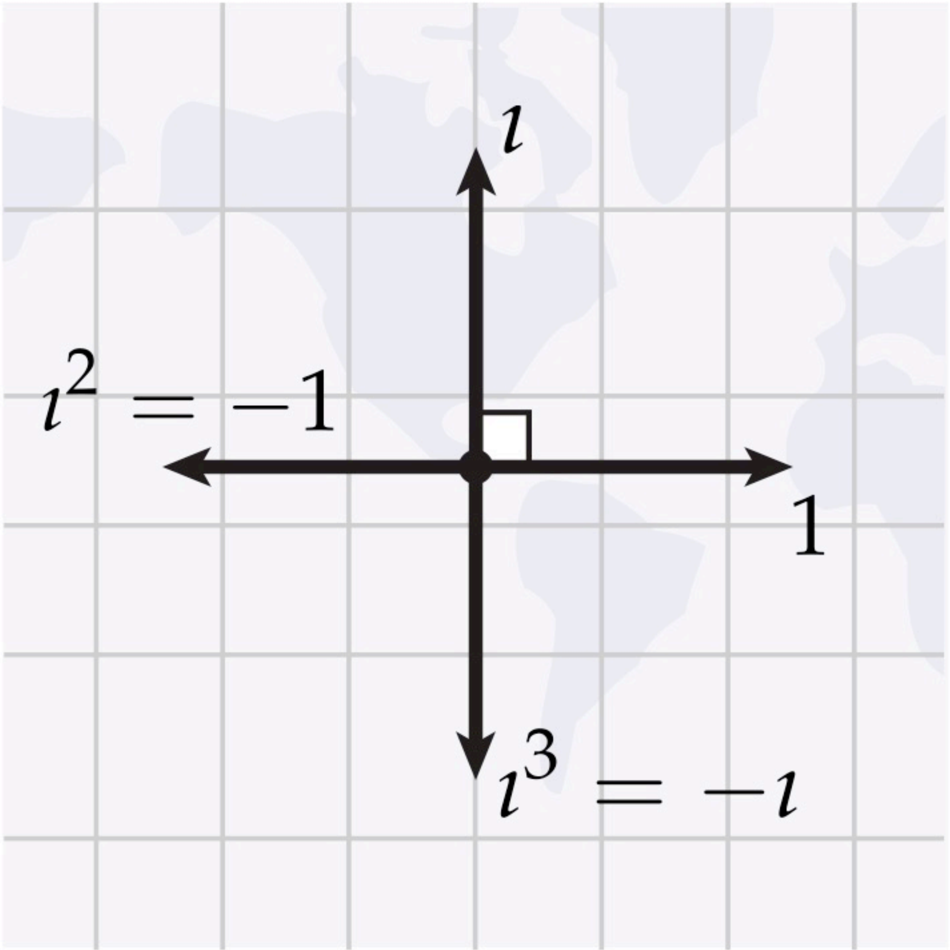
COMPLEX





Truly: no good reason to use 2D vectors instead of complex numbers…

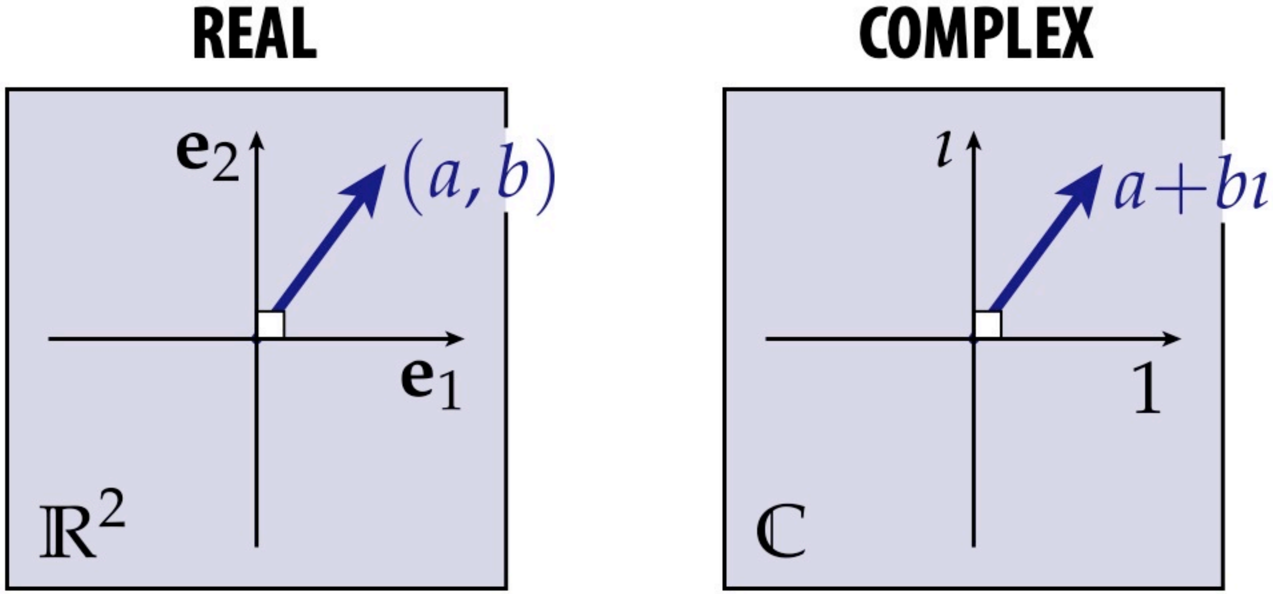
Imaginary Unit—Geometric Description



Imaginary unit is just a quarter-turn in the counter-clockwise direction.

Complex Numbers

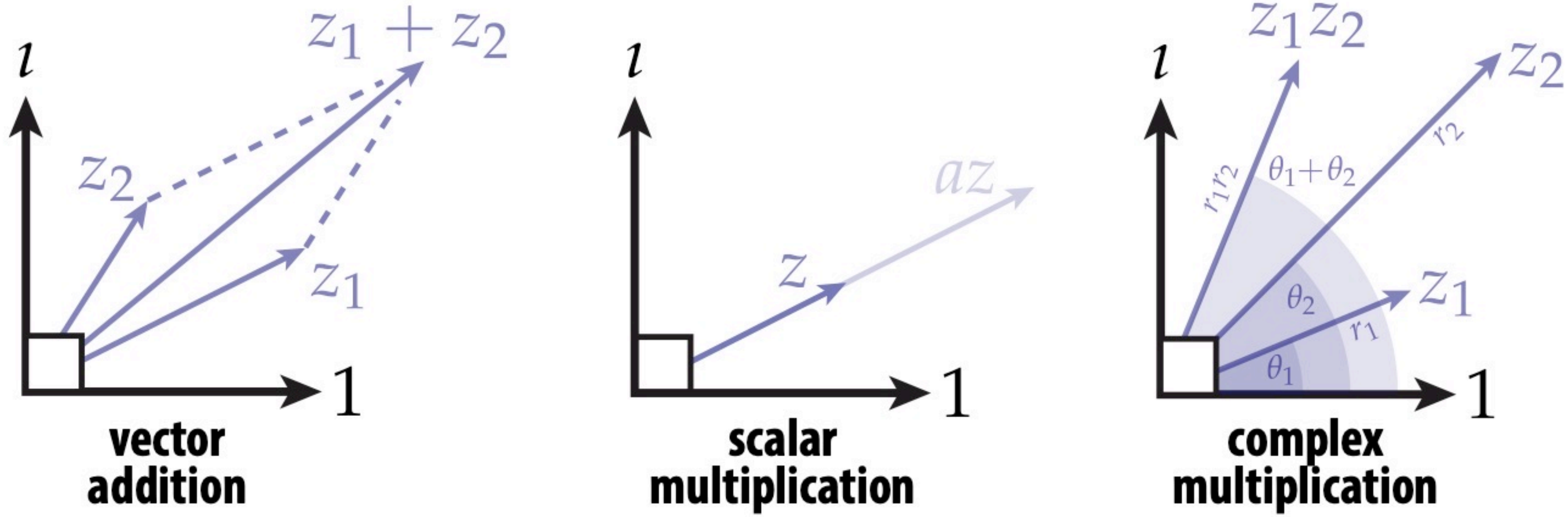
- Complex numbers are then just 2-vectors
- Instead of e_1,e_1, use "1" and "ι" to denote the two bases
- Otherwise, behaves exactly like a real 2-dimensional space



- …except that we're also going to get a very useful new notion of the *product* between two vectors.

Complex Arithmetic

- **Same operations as before, plus one more:**



- **Complex multiplication:**
  - **angles *add***
  - **magnitudes *multiply***

"POLAR FORM"*:

$$z_1 := (r_1, \theta_1)$$
$$z_2 := (r_2, \theta_2)$$
$$z_1 z_2 = (r_1 r_2, \theta_1 + \theta_2)$$

have to be more careful here!

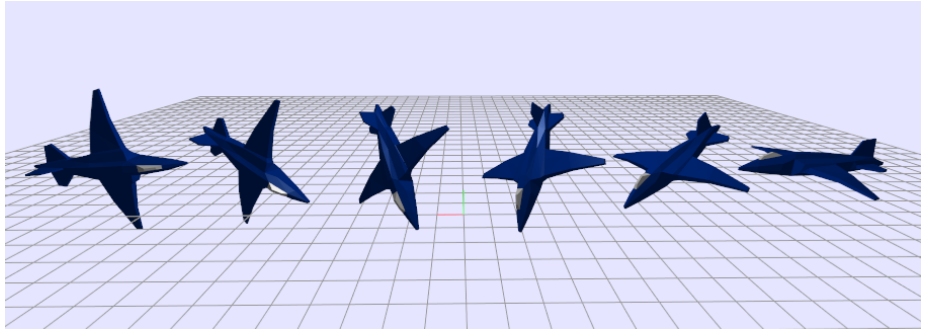
***Not *quite* how it really works, but basic idea is right.**

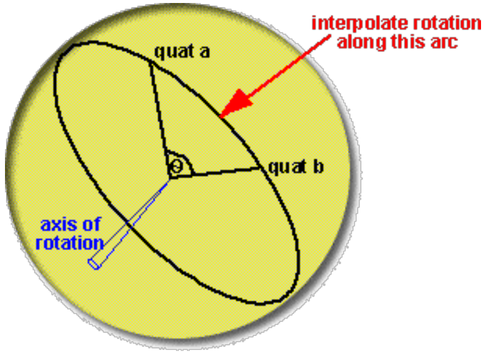
Quaternions are ideal for interpolation

Interpolating Rotations

- Suppose we want to smoothly interpolate between two rotations (e.g., orientations of an airplane)
- Interpolating Euler angles can yield strange-looking paths, non-uniform rotation speed, …
- Simple solution* w/ quaternions: "SLERP" (spherical linear interpolation):

$$\mathrm{Slerp}(q_0, q_1, t) = q_0(q_0^{-1}q_1)^t, \quad t \in [0,1]$$





*Shoemake 1985, "Animating Rotation with Quaternion Curves"

Complex Product—Rectangular Form

- **Complex product in "rectangular" coordinates (1, ı):**

$$z_1 = (a + b\imath)$$

$$z_2 = (c + d\imath)$$

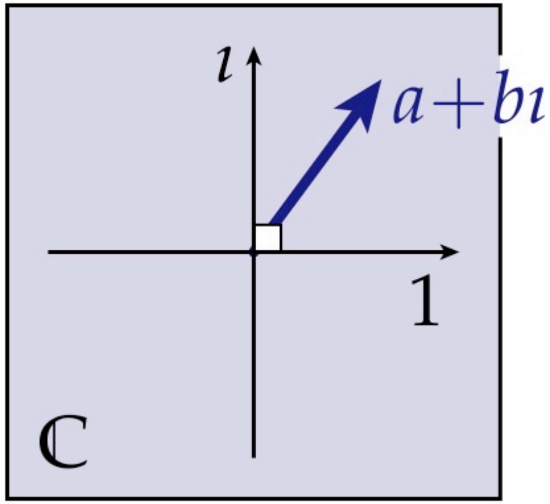
two quarter turns—
same as -1

$$z_1 z_2 = ac + ad\imath + bc\imath + bd\imath^2 =$$

$$\boxed{(ac - bd) + (ad + bc)\imath.}$$

"real part" $\mathrm{Re}(z_1 z_2)$

"imaginary part" $\mathrm{Im}(z_1 z_2)$



- **We used a lot of "rules" here. Can you justify them geometrically?**
- **Does this product agree with our geometric description (last slide)?**

Quaternions

- Rotations can be carried away directly in parameter space via the quaternion product:

  - Concatenation of rotations:

$$\mathbf{q}_1 \circ \mathbf{q}_2 = \left(q_{w,1} q_{w,2} - \mathbf{v}_1 \cdot \mathbf{v}_2 \ , \ q_{w,1} \mathbf{v}_2 + q_{w,2} \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2\right)$$

  - If we want to rotate a vector $\boldsymbol{a}$

$$\boldsymbol{a}' = Rotate(\boldsymbol{a}) = \mathbf{q} \circ \tilde{\boldsymbol{a}} \circ \bar{\mathbf{q}}$$

where $\bar{\mathbf{q}} = (q_w \ - \mathbf{v})$ is the quat conjugate

2D Rotations: Matrices vs. Complex

- **Suppose we want to rotate a vector u by an angle θ, then by an angle φ.**

REAL / RECTANGULAR	COMPLEX / POLAR
$\mathbf{u} = (x, y)$ $\quad \mathbf{A} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$	$u = re^{\imath\alpha}$
$\mathbf{B} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$	$a = e^{\imath\theta}$ $b = e^{\imath\phi}$
$\mathbf{Au} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$	$abu = re^{\imath(\alpha+\theta+\phi)}.$
$\mathbf{BAu} = \begin{bmatrix} (x\cos\theta - y\sin\theta)\cos\phi - (x\sin\theta + y\cos\theta)\sin\phi \\ (x\cos\theta - y\sin\theta)\sin\phi + (x\sin\theta + y\cos\theta)\cos\phi \end{bmatrix}$	
$= \cdots \text{some trigonometry} \cdots =$	
$\mathbf{BAu} = \begin{bmatrix} x\cos(\theta+\phi) - y\sin(\theta+\phi) \\ x\sin(\theta+\phi) + y\cos(\theta+\phi) \end{bmatrix}.$	

Complex Product—Polar Form

- **Perhaps most beautiful identity in math:**

$$e^{\imath\pi} + 1 = 0$$

- **Specialization of *Euler's formula*:**

$$e^{\imath\theta} = \cos(\theta) + \imath\sin(\theta)$$

- **Can use to "implement" complex product:**

$$z_1 = ae^{\imath\theta}, \quad z_2 = be^{\imath\phi}$$

$$z_1 z_2 = abe^{\imath(\theta+\phi)}$$

(as with real exponentiation, exponents *add*)

Leonhard Euler (1707–1783)