Mathematical Foundations of
Computer Graphics and Vision

Variational Methods I

Luca Ballan

Half of the course…

	# variables	domain
▪ classic optimization	finite	dense $\mathbb{R}$
▪ discrete optimization	finite	discrete
▪ optimization on ∞-dimensional spaces	∞	dense $\mathbb{R}$
▪ optimization on manifolds	finite	dense $\mathbb{R}$ but highly constraint

Why Optimization?

- Why optimization is important for applied sciences?
- Where did the concept of “algorithm” go?

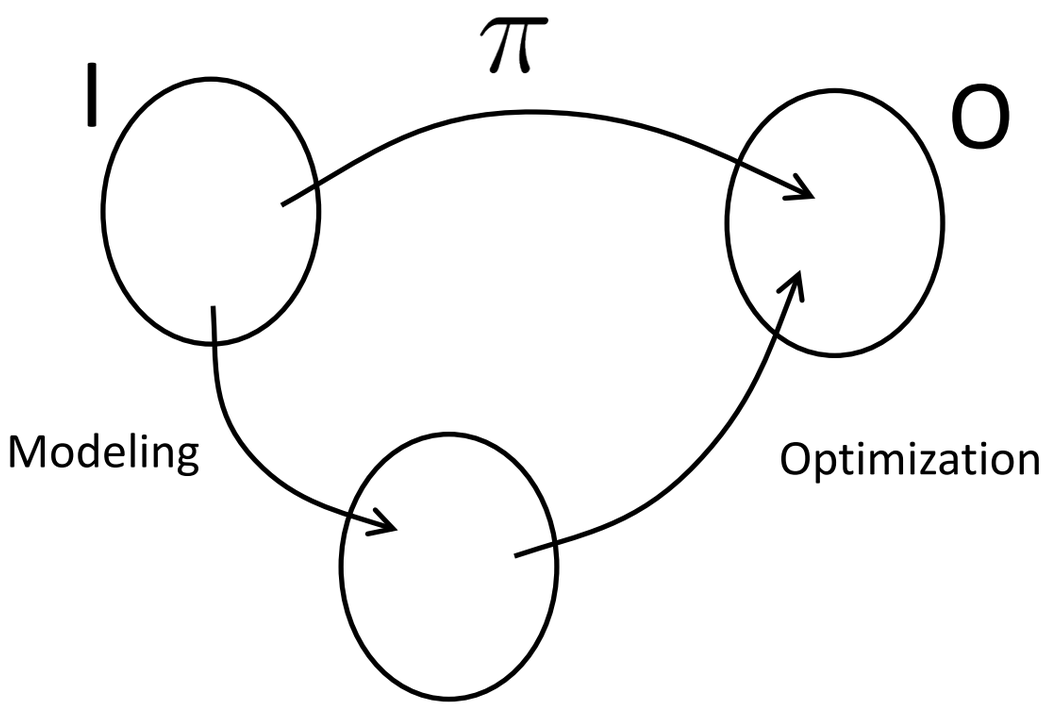
- Formulating a real-world problem as an optimization problem is just a **problem solving paradigm**. It is not the best method, it is just one of the possible ones.

- What is the formal definition of a problem?

Why Optimization?

- Why optimization is important for applied sciences?
- Where did the concept of “algorithm” go?

- Formulating a real-world problem as an optimization problem is just a **problem solving paradigm**. It is not the best method, it is just one of the possible ones.

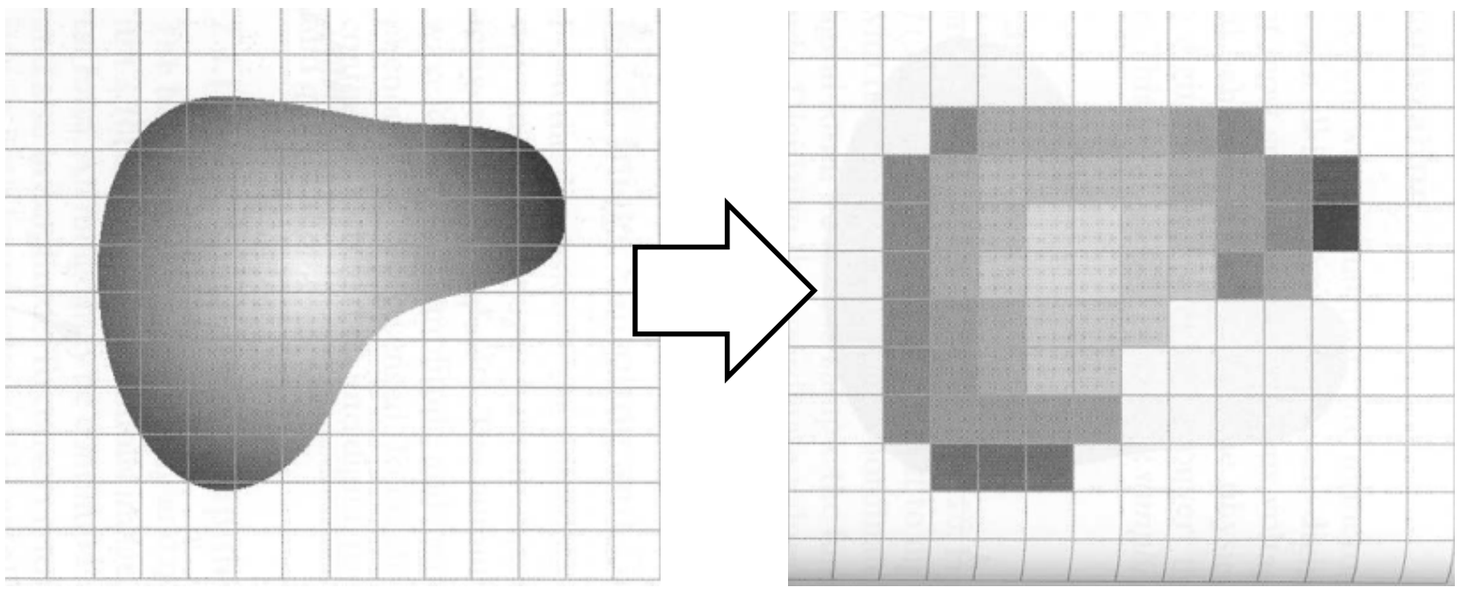


- Optimization problems are well studied.
- Reducing the problem to something that somebody else has already solved.
- If the optimization can be solved perfectly, can we say that we will be able to solve the problem perfectly?

Why Infinite-dimensional Spaces?

- Digital images and videos are discrete (numerical signal).
  - Discrete in color or brightness space (q**uantization**)
  - Discrete in the physical space (**space sampling**)
  - (videos) Discrete in time (**time sampling**)

- We are used to consider them discrete, because of the limitations of our processing units



“Infinite-Dimensional” Representation

Finite-Dimensional Representation

Why Infinite-dimensional Spaces?

- We are used to discrete representations, because:
  - digital objects are discrete, and their processing in a computer will ultimately require a discretization
  - No numerical approximations in modeling the transition from discrete to continuous
  - For various problems there exist efficient algorithms from discrete optimization

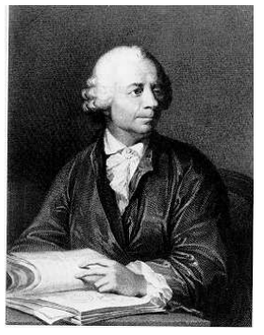
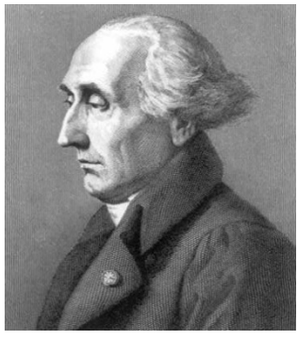
- But continuous representations have some advantages:
  - **The world is continuous** ergo **the images should be treated as continuous functions**
  - **There exists a huge mathematical theory for continuous functions** (functional analysis, differential geometry, partial differential equations, etc...)
  - Certain properties (e.g. rotational invariance) are easier to model in a continuous way
  - Finally, continuous models correspond to the limit of infinitely fine discretization

*

Calculus of Variations

- **Calculus of variations** is a classical topic in mathematics and in physics:
  in fact, in mechanics, it forms the basis for the **least action principle** which says that the motion of a particle lies on a stationary (minimum) point of a functional (the action).

- e.g., the Fermat's principle (the principle of least time): the path taken between two points by a ray of light is the path that can be traversed in the least time
  (the geodesic of the space)


Reference book: **Gelfand Fomin, Calculus of Variations, Prentice Hall, 1963**

Simple Optimization

- Given $\mathbb{R}^k$ (**Vector space** of dimension k over the **field** $\mathbb{R}$)
- Given $L \in C^1(\mathbb{R}^k, \mathbb{R})$ (**Loss functional** of class C^1 over $\mathbb{R}^k$)
- Find x^* such that

$$x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$$ (**unconstrained minimization problem**)

- How do we solve for it?

There are many different ways to solve it!!

*

A possible solution

$$x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$$

- Compute the gradient of $L \longrightarrow \nabla L : \mathbb{R}^k \to \mathbb{R}^k$
- Compute the set of stationary points

$$S_L = \{x \in \mathbb{R}^k \mid \nabla L(x) = 0_k\}$$

- we know that, the point we are looking for, x^*, belongs to S_L

**

- therefore, we can solve for this new problem

$$x^* = \arg\min_{x \in S_L} L(x)$$

it is guarantee that this coincides with the solution of the original problem $x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$

If the cardinality of S_L is finite

- This optimization is relatively easy.
- It becomes an optimization over a discrete domain.
- One can use brute force, or some heuristics, or …)

A possible solution

- Given $x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$

Real Scenario:

- Compute $\nabla L : \mathbb{R}^k \to \mathbb{R}^k$

- maybe we cannot have analytical expression of ∇L
- maybe because we neither have an analytical expression of L

- Compute $S_L = \{x \in \mathbb{R}^k \mid \nabla L(x) = 0_k\}$

- To find S_L we need to solve for an equation

$$\nabla L(x) = 0_k$$

- which can be very difficult to solve!

- Solve $x^* = \arg\min_{x \in S_L} L(x)$

- if S_L is not finite, this optimization is still not an easy problem
- we still have to discriminate between minimum, saddle, and maximum, local and global
- if the problem is convex and x^* is a minimum -> we know that x^* is the global minimum

Another solution

$$x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$$

- That was what we are used to do in mathematical analysis
- But if we have a computer, we might prefer an “iterative” approach
- **Descent techniques**: “one starts from one point in $\mathbb{R}^k$ and you go straight down the hill until he/she hits a local minima”

*

- **Gradient descent** is a particular descent technique which always uses as descent direction the opposite of the gradient
  (i.e. the direction in which the functional decreases most)

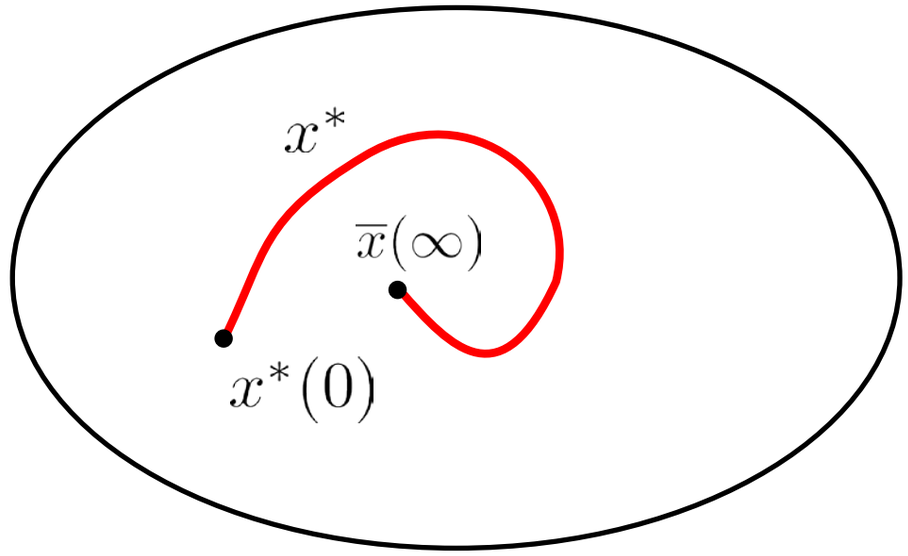
Another solution

$$x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$$

- This corresponds to adding a time dimension to our solution x^*
- i.e., x^* becomes a curve in our solution space $\mathbb{R}^k$

$$x^*(t) \in \mathbb{R}^k$$

- which start from an initial solution $x^*(0) = x_0$
- and it evolves according to a PDE

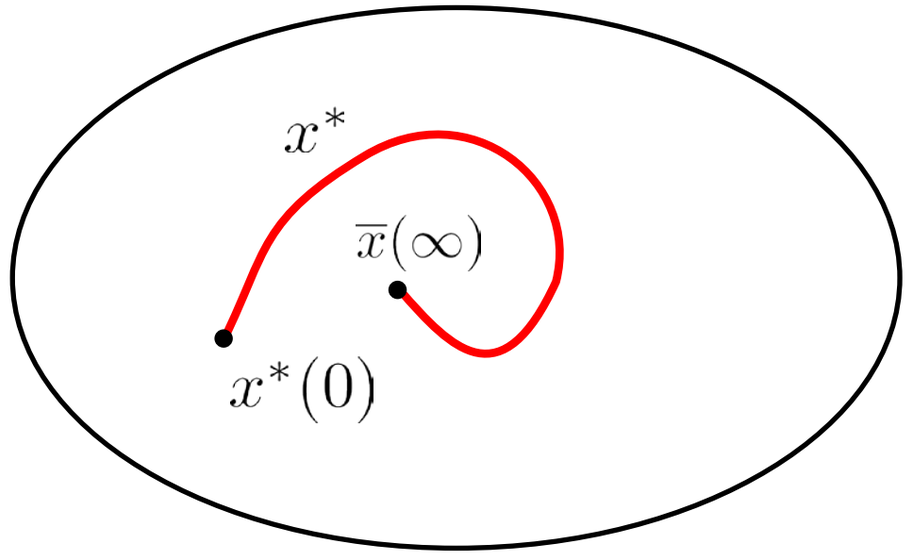


Another solution

$$x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$$

- This corresponds to adding a time dimension to our solution x^*
- i.e., x^* becomes a curve in our solution space $\mathbb{R}^k$

$$x^*(t) \in \mathbb{R}^k$$



- which start from an initial solution $x^*(0) = x_0$
- and it evolves according to a PDE

$$\begin{cases} x^*(0) = x_0 \\ \dfrac{\partial x^*}{\partial t}(t) = -\nabla L(x^*(t)) \end{cases} \qquad \text{(PDE)}$$

Dinamic System

$$x^* = \lim_{t \to \infty} x^*(t)$$

We hope that this dynamical system converges to the solution of our problem (in a finite time better)

A More Complex Optimization Problem

- Given $\mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$ =
  - the set of all the functions of type $\mathbb{R}^k \to \mathbb{R}^m$
  - **vector space** (infinite dimensional) over the **field** $\mathbb{R}$

- Given $L \in C^1(\ \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)\ , \mathbb{R})$ (**Loss functional** of class C^1 over $\mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$)

- Find u^* such that

$$u^* = \underset{u \in \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)}{\arg\min}\ L(u)$$

(**unconstrained minimization problem** with functions as domain)

- How do we solve for it?

A More Complex Optimization Problem

$$L \in C^1(\ \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)\ , \mathbb{R})$$

- How does the gradient of a functional of functions look like?
- Can it look like

$$\nabla L = \left(\frac{\partial L}{\partial x_1}, \ldots, \frac{\partial L}{\partial x_k} \right) \ \textbf{?}$$

  - What are $(x_1, \ldots, x_k)$ in this case?

- How does it look like the derivative? $\dfrac{\partial L}{\partial ?}$
  - what do I need to place here?

- a lot of confusion!!!

Intuitively

$$L : \mathbb{R}^k \to \mathbb{R} \quad \Rightarrow \quad \nabla L = \left(\frac{\partial L}{\partial x_1}, \ldots, \frac{\partial L}{\partial x_k} \right)$$

the domain of optimization has dimension k

the gradient length is k

$$L : \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m) \to \mathbb{R} \quad \Rightarrow \quad \nabla L$$

the domain of optimization is infinite dimensional

the gradient length is infinite, so maybe, the gradient should be a function

Calculus of Variations

- Calculus of variation extends the concept of gradient and derivative to all the functional defined on a generic topological vector space (either finite or infinite dimensional)

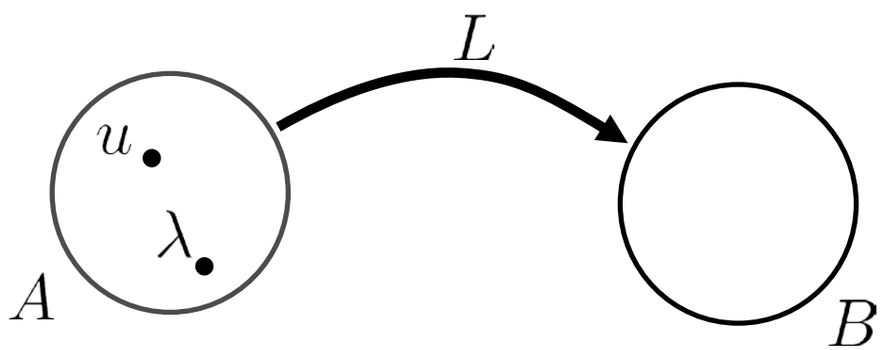
- We start describing this topic with the definition of the concept of **Directional Derivative**

Directional Derivative (of a function)

- Given $A,\ B$

Vector spaces over a same **field** (e.g. $\mathbb{R}$) with **topology**

- Given $L : A \to B$
- and $u, \lambda \in A$



- There exists an object

$$\frac{\partial L}{\partial \lambda}(u) = \lim_{\epsilon \to 0} \frac{L(u + \epsilon \cdot \lambda) - L(u)}{\epsilon}$$

$\in A$ $\in B$

Directional Derivative of L with direction λ evaluated in u

- The directional derivate is a function

$$\frac{\partial L}{\partial \lambda} : A \to B$$

*

Gradient (of a functional)

- If $B = \mathbb{R}$ (vector space with topology)
- If A is a general topological vector space with an **inner product**

$$\langle \cdot, \cdot \rangle_A : A \times A \to \mathbb{R}$$

- Given $L : A \to \mathbb{R}$ (**functional**), there might exists an object

$$\nabla L : A \to A$$

Gradient of L

informally, indicating the "direction" of maximal increase of L

- For each point $u \in A$, the gradient of L is formally defined as the unique element of A such that

$$\underbrace{\frac{\partial L}{\partial \lambda}(u)}_{\in \mathbb{R}} = \underbrace{\langle \underbrace{\nabla L(u)}_{\in A}, \underbrace{\lambda}_{\in A} \rangle_A}_{\in \mathbb{R}}$$

Properties

$$\frac{\partial L}{\partial \lambda}(u) = \langle \nabla L(u), \lambda \rangle_A$$

- If we restrict to all the directions with norm 1, the directional derivative has its maximum where the direction of λ is parallel to the gradient $\nabla L(u)$

$$\|\lambda\|_A = \sqrt{\langle \lambda, \lambda \rangle_A} = 1$$

Norm inherited by the inner product

- **Class C1**: If the gradient $\nabla L(u)$ exists and it is continuous for all $u \in A$, then L is said to be of class $C^1(A, \mathbb{R})$

- If $L \in C^1(A, \mathbb{R})$, $\frac{\partial L}{\partial \lambda}(u)$ is linear in both L and λ

Stationary points of L

- If $L \in C^1(A, \mathbb{R})$, the set of stationary point is defined as

$$S_L = \{u \in A \mid \nabla L(u) = 0_A\}$$

$$= \left\{ u \in A \mid \frac{\partial L}{\partial \lambda}(u) = 0_{\mathbb{R}}, \forall \lambda \in A \right\}$$

- L is locally flat
- there is no direction of maximum variation

*

- if u is a minimum for L then any small variation of u (along any direction λ) would cause no effect on the value returned by L

*

Summary

- Given A a topological vector space with an inner product
- and given a functional of type $L : A \to \mathbb{R}$ sufficiently regular, i.e. $\in C^1(A, \mathbb{R})$

- it is defined an object called **directional derivative**

$$\frac{\partial L}{\partial \lambda} : A \to \mathbb{R}$$

(which, for every couple of elements in A, returns a value in $\mathbb{R}$)

- it is defined an object called **gradient**

$$\nabla L : A \to A$$

(which, for every elements of A, returns another element of A)

- it is defined the set of **stationary points** as

$$S_L = \{u \in A \mid \nabla L(u) = 0_A\}$$
$$\Big| $$
$$= \left\{ u \in A \mid \frac{\partial L}{\partial \lambda}(u) = 0_{\mathbb{R}}, \forall \lambda \in A \right\}$$

Let's pick $A = \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$

- $A = \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$ = the set of all the functions of type $\mathbb{R}^k \to \mathbb{R}^m$
- $(\mathbb{F}(\mathbb{R}^k, \mathbb{R}^m), +, \cdot_e)$ is a **vector space** over the field $\mathbb{R}$ (infinite dimensional)
- It admits an **inner product**, defined as

$$\langle f, g \rangle = \int_{\mathbb{R}^k} \langle f(x), g(x) \rangle_{\mathbb{R}^m} \, dx$$

- Therefore, it admits a **norm** $\|f\| = \sqrt{\langle f, f \rangle}$ (inherited from the inner product)
- and it admits a **metric** $d(f, g) = \|f - g\|$ (inherited from the norm)

- Therefore, it is a **topological vector space** ⇒ everything defined before should be defined also for this space!

PS: this space is called the **Lebesgue space of order 2** (L^2). It has the structure of an **Hilbert space** and it is a very important for the theory of the Fourier Transform and the theory of probability.

The functional $L : A = \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m) \to \mathbb{R}$

- Given

$$L : \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m) \to \mathbb{R}$$

- the **directional derivative** is defined (as before)

$$\frac{\partial L}{\partial \lambda}(u) = \lim_{\epsilon \to 0} \frac{L(u + \epsilon \cdot \lambda) - L(u)}{\epsilon}$$

(Gâteaux derivative)

where $u \in \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$
$\lambda \in \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$

*

- the **gradient** is defined (as before)

$$\nabla L : \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m) \to \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)$$

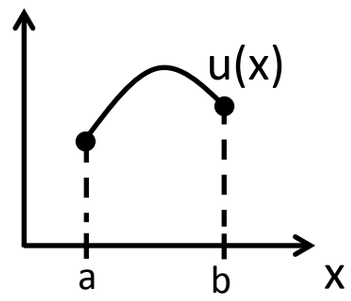
$$\frac{\partial L}{\partial \lambda}(u) = \langle \nabla L(u), \lambda \rangle$$

*

- **The gradient is unlikely to exist**

A simple Loss Functional

- Given $L : C^2([a,b],\mathbb{R}) \to \mathbb{R}$



$$L(u) = \int_a^b \psi(x, u(x), \dot{u}(x))dx$$

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left(\frac{\partial \psi}{\partial u}(x, u(x), \dot{u}(x)) - \frac{\partial}{\partial x}\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x)) \right) \lambda(x)dx + \left[\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x))\lambda(x) \right]_a^b$$

$$\nabla L(u)(x) = \frac{\partial \psi}{\partial u}(x, u(x), \dot{u}(x)) - \frac{\partial}{\partial x}\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x))$$

Mathematical Foundations of
Computer Graphics and Vision

Variational Methods II

Luca Ballan

Last Lecture

- If we have a topological vector space A with an inner product
- and functionals of type $L : A \to \mathbb{R}$ sufficiently regular

- it is possible to define an object called **directional derivative**

$$\frac{\partial L}{\partial \lambda}(u) = \lim_{\epsilon \to 0} \frac{L(u + \epsilon \cdot \lambda) - L(u)}{\epsilon}$$

(that not always exists)

- But when this exists for all possible directions, and there exist a function

$$\nabla L : A \to A$$

such that

$$\frac{\partial L}{\partial \lambda}(u) = \langle \nabla L(u), \lambda \rangle_A$$

- then this function is called **gradient**

(and it exists very rarely)

Last Lecture

- If the directional derivative for L exists for all **points and directions** in A then the set of stationary point is defined as

$$S_L = \left\{ u \in A \mid \frac{\partial L}{\partial \lambda}(u) = 0_{\mathbb{R}}, \forall \lambda \in A \right\}$$

The set of points in A where small variations in A coincide with zero variations in L

- If the gradient for L exists for all points in A then this set can be equivalently defined as

$$S_L = \{ u \in A \mid \nabla L(u) = 0_A \}$$

The Space L^2

- Given

$$A \subseteq \mathbb{F}(\Omega \subseteq \mathbb{R}^k, \mathbb{R}^m)$$

such that

$$A = \left\{ \quad \forall f : (\Omega \subseteq \mathbb{R}^k) \to \mathbb{R}^m \quad | \quad \int_\Omega \|f(x)\|^2 \, dx \quad \text{exists} \right\}$$

*

- Therefore for any functional of type

$$L : L^2(\Omega \subseteq \mathbb{R}^k, \mathbb{R}^m) \to \mathbb{R}$$

the directional derivative and gradient can be defined (when they exist).

The functional $L : A \to \mathbb{R}$

- the **directional derivative**

$$\frac{\partial L}{\partial \lambda}(u) = \lim_{\epsilon \to 0} \frac{L(u + \epsilon \cdot \lambda) - L(u)}{\epsilon}$$

$$\Big| \quad \Big|$$

$$= \lim_{\epsilon \to 0} \frac{\partial}{\partial \epsilon} L(u + \epsilon \cdot \lambda)$$

(Gâteaux derivative)

where

$$u \in \mathbb{F}(\Omega \subseteq \mathbb{R}^k, \mathbb{R}^m)$$
$$\lambda \in \mathbb{F}(\Omega \subseteq \mathbb{R}^k, \mathbb{R}^m)$$

- and the **gradient** is defined is that object of type $\nabla L : \mathbb{F}(\Omega \subseteq \mathbb{R}^k, \mathbb{R}^m) \to \mathbb{F}(\Omega \subseteq \mathbb{R}^k, \mathbb{R}^m)$ that satisfies this relationship (if it exists)

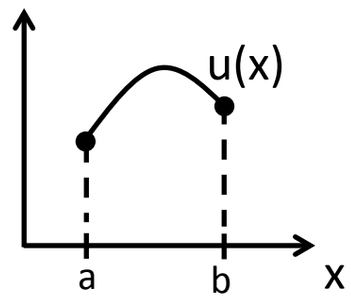
$$\frac{\partial L}{\partial \lambda}(u) = \langle \nabla L(u), \lambda \rangle$$

$$\Big| \quad \Big|$$

$$= \int_\Omega \langle \nabla L(u)(x), \lambda(x) \rangle_{\mathbb{R}^m} \, dx$$

A simple Loss Functional

- Given $L : C^2([a,b],\mathbb{R}) \to \mathbb{R}$



$$L(u) = \int_a^b \psi(x, u(x), \dot{u}(x))dx$$

*

- The directional derivative:

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left(\frac{\partial \psi}{\partial u}(x, u(x), \dot{u}(x)) - \frac{\partial}{\partial x}\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x)) \right) \lambda(x)dx + \left[\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x))\lambda(x) \right]_a^b$$

- or shortly:

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \psi}{\partial \dot{u}} \right) \lambda dx + \left[\right]_a^b$$

In some situations, this second term is zero.

A simple Loss Functional

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial x} \frac{\partial \psi}{\partial \dot{u}} \right) \lambda dx$$

Directional derivative

- In these situations the gradient of L exists (sufficient and necessary condition for its existence)

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial x} \frac{\partial \psi}{\partial \dot{u}} \right)$$

Gradient $\in C^0([a,b], \mathbb{R})$

*

- u is a stationary point/function for L if and only if

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial x} \frac{\partial \psi}{\partial \dot{u}} \right) = \mathbf{0}$$

(PDE)

has to be equal to the null function!!

Euler-Lagrange equation
(necessary condition for optimality, not in general sufficient)

*

Boundary Conditions

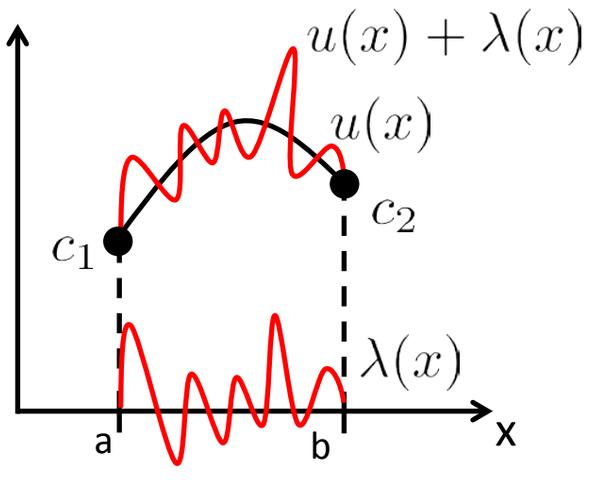
- Sufficient and necessary condition for the gradient to exists is that the second term of the directional derivative is zero

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \psi}{\partial \dot{u}} \right) \lambda dx + \left[\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x))\lambda(x) \right]_a^b$$

- If the problem to minimize has a constraint of type

$$\begin{cases} u(a) = c_1 \\ u(b) = c_2 \end{cases}$$

- it is logical to consider only variations/directions λ which maintain the extremes of the curve



$$\begin{cases} \lambda(a) = 0 \\ \lambda(b) = 0 \end{cases}$$

Dirichlet boundary condition

$$\left[\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x))\lambda(x) \right]_a^b = 0$$

Boundary Conditions

$$\left[\frac{\partial\psi}{\partial\dot{u}}(x,u(x),\dot{u}(x))\lambda(x)\right]_a^b = 0$$

$$\Rightarrow \quad \frac{\partial L}{\partial\lambda}(u) = \int_a^b \left(\frac{\partial\psi}{\partial u} - \frac{\partial}{\partial x}\frac{\partial\psi}{\partial\dot{u}}\right)\lambda dx$$

$$\begin{cases} \lambda(a) = 0 \\ \lambda(b) = 0 \end{cases}$$

Dirichlet boundary condition

$$\begin{cases} \dfrac{\partial\psi}{\partial\dot{u}}(a,u(a),\dot{u}(a)) = 0 \\ \dfrac{\partial\psi}{\partial\dot{u}}(b,u(b),\dot{u}(b)) = 0 \end{cases}$$

Von Neumann boundary condition

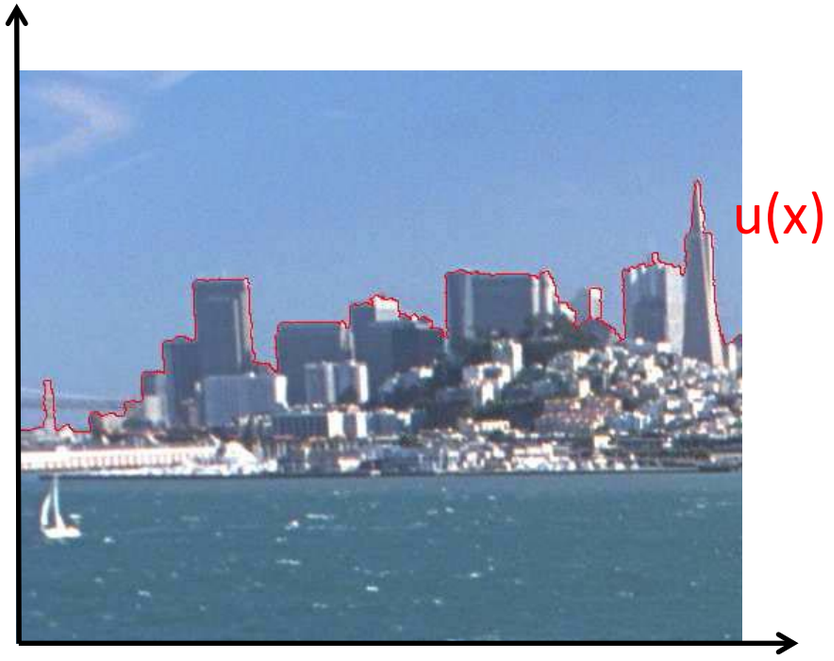
$$\begin{cases} u(a) = u(b) \\ \dot{u}(a) = \dot{u}(b) \\ \dfrac{\partial\psi}{\partial\dot{u}} \quad \text{does not depend on x} \\ \lambda(a) = \lambda(b) \end{cases}$$

(no name) boundary condition
(but it is the most useful)

*

Application Example: Sky-Line detection

Input

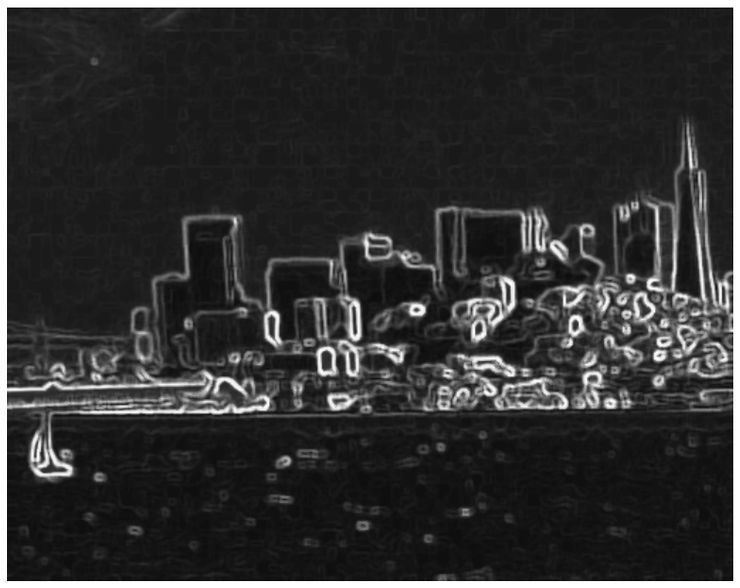


Segmented image

$$L(u) = \int_0^1 \underbrace{-E(x, u(x))^2} + \dot{u}(x)^2 dx$$

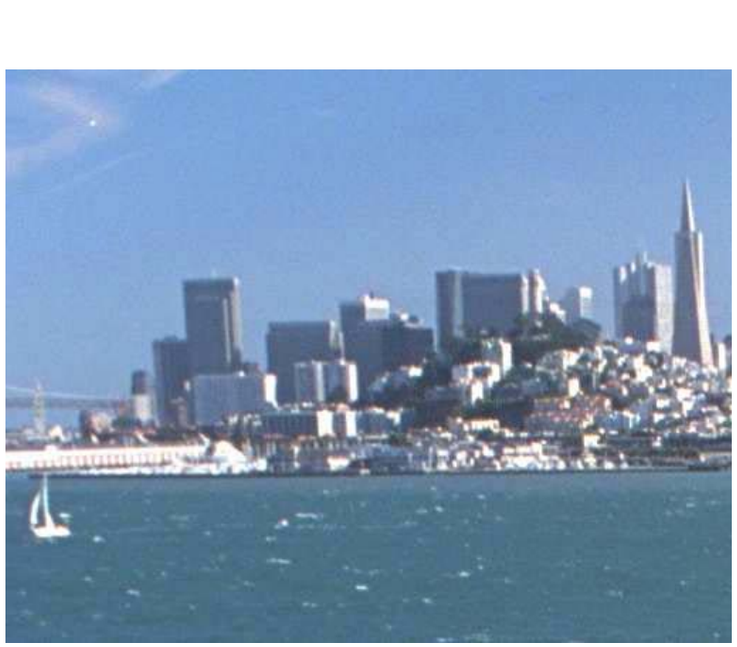
$$E(x, y) = \frac{\partial I}{\partial y}(x, y)$$

it penalizes curves passing through non-edge pixels

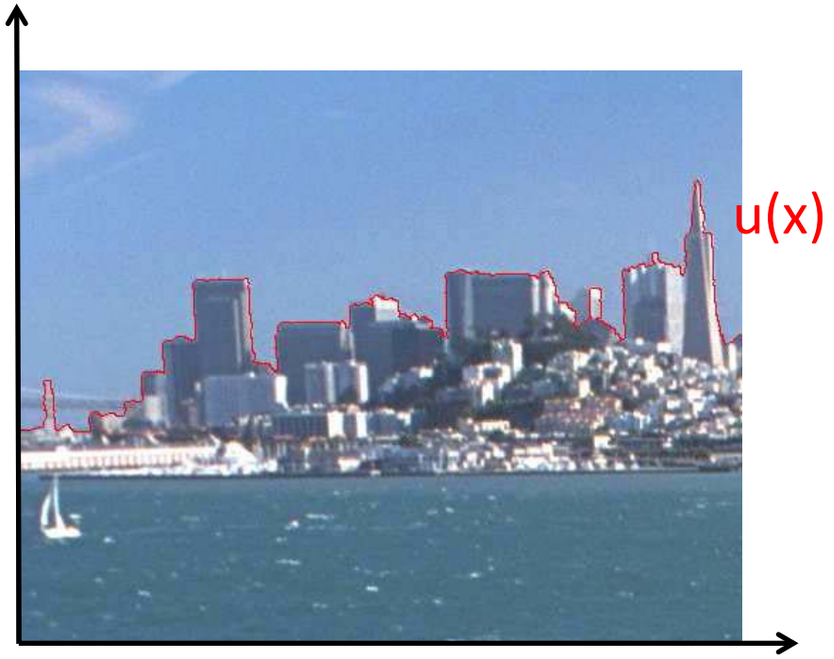


Edge Map
(it is an image)

Application Example: Sky-Line detection



Input



Segmented image

$$L(u) = \int_0^1 \underbrace{-E(x, u(x))^2}_{} + \underbrace{\dot{u}(x)^2}_{} dx$$

Snakes model, or **Active Contour Model**
[Kass, Witkin, Terzopoulos IJCV'88]

$$E(x, y) = \frac{\partial I}{\partial y}(x, y)$$

it penalizes curves passing through non-edge pixels

it penalizes curves which are non-smooth

Application Example: Sky-Line detection

- To solve this problem, first compute the edge map $E(x,y) = \frac{\partial I}{\partial y}(x,y)$

- then solve the following minimization problem

$$u^* = \underset{u \in \mathbb{C}^2([0,1],\mathbb{R})}{\arg\min} \int_0^1 -E(x,u(x))^2 + \dot{u}(x)^2 dx$$

- compute the Euler-Lagrange Equation

$$\nabla L(u) = \frac{\partial \psi}{\partial u} 2E\left(\frac{\partial}{\partial x}, \frac{\partial \psi}{\partial \dot{u}}\right) \frac{\partial E}{\partial y}(x,u(x)) - 2\ddot{u}(x) = 0 \qquad *$$

- This is still a PDE, but the solution is not obvious, since we do not have a close formula for $E(x,y)$

- But, we can always resort to a **gradient descent** approach.

Gradient Descent

- First we need to add a time dimension to our problem $\longrightarrow$ $u^*(t,x)$

- Solve the dynamical system

$$\begin{cases} u^*(0,x) = u_0(x) \\ \dfrac{\partial u^*}{\partial t}(t,x) = -\nabla L(u^*(t,x)) \end{cases}$$

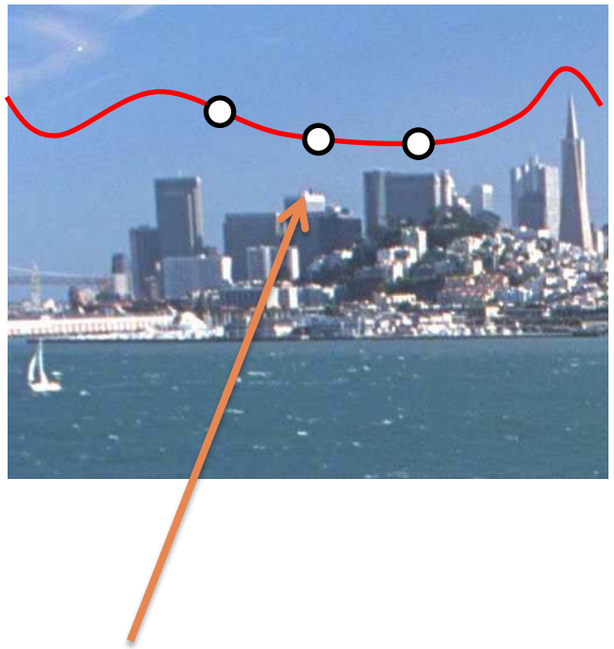
$u_0(x)$

$$= 2E(x, u^*(t,x)) \frac{\partial E}{\partial y}(x, u^*(t,x)) + 2\ddot{u}^*(t,x)$$

- We know that the solution of our problem is (if it converges)

$$u^*(x) = \lim_{t \to \infty} u^*(t,x)$$

Gradient Descent

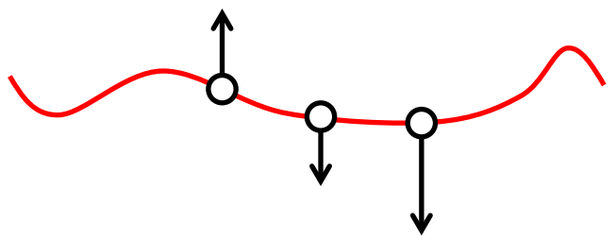


$u^*(t, x)$

- Let fix an $\overline{x}$, $u^*(t, \overline{x})$ represents the y-position of a particle (the particle $\overline{x}$) evolving over time. (something falling down)

- This particle starts from $u_0(\overline{x})$ at time $t = 0$

- It then falls down in the image with speed $\dfrac{\partial u^*}{\partial t}(t, \overline{x})$



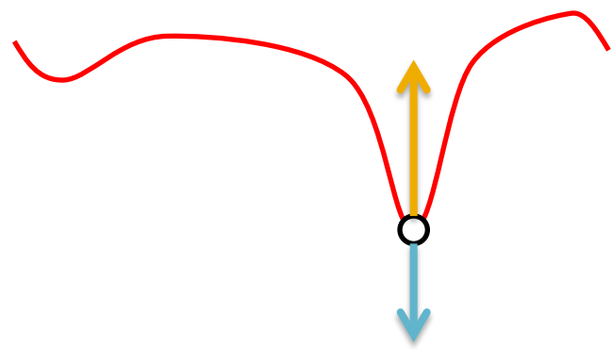
Gradient Descent

- The speed of this particle

$$\frac{\partial u^*}{\partial t}(t,\overline{x}) = \underbrace{2E(\overline{x},u^*(t,\overline{x}))\frac{\partial E}{\partial y}(\overline{x},u^*(t,\overline{x}))}_{\text{depends by the edge map E}} + \underbrace{2\ddot{u}^*(t,\overline{x})}_{\text{and by the second derivative of the curve in } \overline{x}}$$

- These two terms can be thought as

$$\frac{\partial u^*}{\partial t}(t,\overline{x}) = \text{External force on } \overline{x} \;+\; \text{Internal force on } \overline{x}$$



Numerical Implementation

$$\begin{cases} u^*(0,x) = u_0(x) \\ \dfrac{\partial u^*}{\partial t}(t,x) = -\nabla L(u^*(t,x)) \end{cases}$$

- The curve $u^*(t,x)$ needs to be discretized in space and time

$$\begin{cases} u^*(0,x) = u_0(x) \\ u^*(t+1,x) = u^*(t,x) - \beta \nabla L(u^*(t,x)) \end{cases}$$

time discretization

$$x \in \{0, \tfrac{1}{n}, \tfrac{2}{n}, \ldots, 1\}$$

space discretization

- β is a constant guiding the convergence of the dynamical system: a too high β makes the system unstable; a too small β makes the convergence too slow. From the Finite Element theory, the choice β depends (also) on the used spatial parameterization/sampling $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$

- β can be computed in each step using the line search algorithm
(no more a curve evolving but a pure optimization technique)

*

Numerical Implementation

$$\begin{cases} u^*(0,x) = u_0(x) \\ \dfrac{\partial u^*}{\partial t}(t,x) = -\nabla L(u^*(t,x)) \end{cases}$$

- The curve $u^*(t,x)$ needs to be discretized in space and time

$$\begin{cases} u^*(0,x) = u_0(x) \\ u^*(t+1,x) = u^*(t,x) - \beta \nabla L(u^*(t,x)) \end{cases}$$

time discretization

$$x \in \{0, \tfrac{1}{n}, \tfrac{2}{n}, \ldots, 1\}$$

space discretization

- for a sufficiently high t , $u^*(t,x)$ will lie on a **local minima** of L

Gradient Evaluation

$$\nabla L(u^*(t,x)) = \underbrace{-2E(x,u^*(t,x))\frac{\partial E}{\partial y}(x,u^*(t,x))}_{\text{external force}} - 2\ddot{u}^*(t,x)$$

- To compute the external force for a point in the curve (i.e. a particle), we just need to evaluate a point in the function (floating point image)

$$E(x,y)\frac{\partial E}{\partial y}(x,y) \qquad \text{where} \qquad E(x,y) = \frac{\partial I}{\partial y}(x,y)$$

- this function/image can be pre-computed before running the optimization (e.g. using Sobel filters).
- Btw, better smooth I before otherwise $\frac{\partial I}{\partial y}$ is full of Dirac deltas (which are useless for a continuous evolution)
- the evaluation is independent for each particle (to estimate the external force of one particle, we do not need to know the state of the other particles)
- therefore, this operation is parallelizable

Gradient Evaluation

$$\nabla L(u^*(t,x)) = -2E(x,u^*(t,x))\frac{\partial E}{\partial y}(x,u^*(t,x)) - \underbrace{2\ddot{u}^*(t,x)}_{\text{internal force}}$$

- To compute the internal force for a point in the curve (i.e. a particle), we can approximate the second derivative of the curve using **finite differences**

$$\ddot{u}^*(t,u) = \frac{u^*(t,x+h) - 2u^*(t,x) + u^*(t,x-h)}{h^2}$$

- the evaluation is dependent only on the state of neighboring particles $x \pm h$
- therefore, this operation is easily parallelizable on a grid (e.g. on GPU)
- (Variational approaches are very suitable for GPU implementations)

Boundary conditions

$$L(u) = \int_0^1 -E(x, u(x))^2 + \dot{u}(x)^2 dx$$

- We need to make sure that the boundary conditions are satisfied

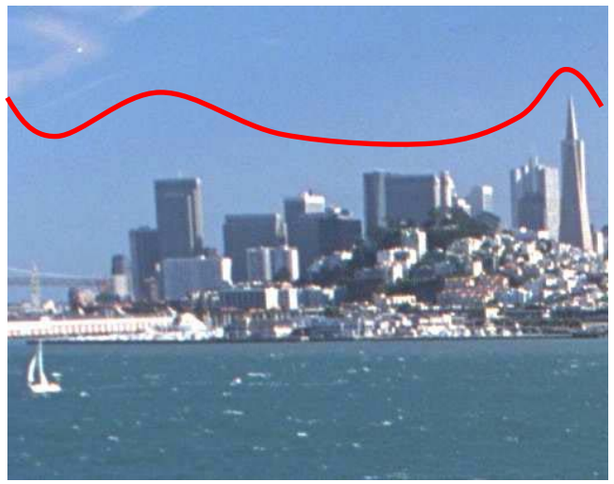
$$\left[\frac{\partial \psi}{\partial \dot{u}}(x, u(x), \dot{u}(x))\lambda(x)\right]_a^b = 0$$

$$\Updownarrow$$

$$[2\dot{u}(x)\lambda(x)]_0^1 = 0$$

$$\Updownarrow$$

$$2\dot{u}(1)\lambda(1) - 2\dot{u}(0)\lambda(0) = 0$$



Dirichlet boundary condition? $\lambda(1) = \lambda(0) = 0$

Fix the extremes at the starting position: bad

Von Neumann boundary condition? $\dot{u}(1) = \dot{u}(0) = 0$

Fix the steepness of the extremes: ok

No name boundary condition? $u(1) = u(0)$ …. Etc…

Both the extremes has to be at the same height: bad

Boundary conditions

$$L(u) = \int_0^1 -E(x, u(x))^2 + \dot{u}(x)^2 dx$$

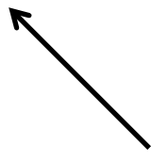
- What if the boundary conditions are not satisfied?

- So, what will this object be?

$$\nabla L(u)(x) = -2E(x, u(x)) \frac{\partial E}{\partial y}(x, u(x)) - 2\ddot{u}(x)$$

- Will it work the same?

- Yes, if $\nabla L(u)$ is still a descent direction for L
  (instead of a gradient descent technique we will have a descent technique)



One has to verify this.. And the verification depends from problem to problem…

Image Segmentation

- The goal of image segmentation is to partition an image into “meaningful” components.

- What is “meaningful” depends on the application. In case of an image representing a real scene, one may want that each of the segmented components corresponds to different physical objects.

Input

Segmented image

Image Segmentation

Typical approaches:

- **Edge-based methods**: running an edge detector to identify the contours on the color or on the texture. Then group the output into connected curves.

- **Region-based methods:** identify regions in the image for which some criterion is more or less uniform (brightness, color, textures,...) (e.g. by thresholding or clustering). Then use region growing, region merging, connected components etc.. to refine the segmentation.

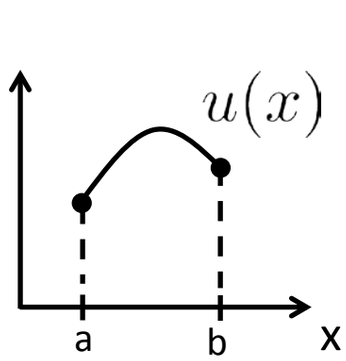
Image Segmentation: a Variational Appr.

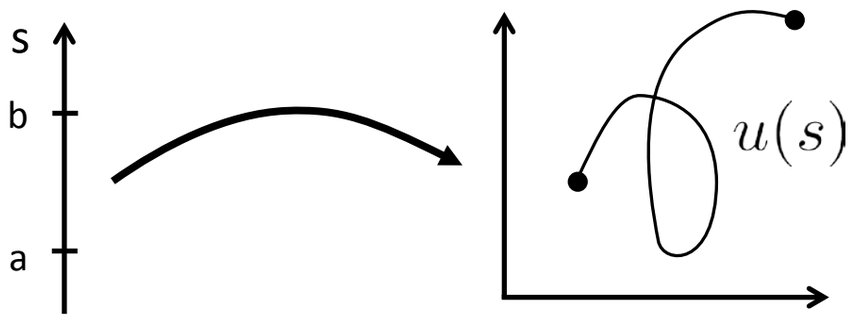
D. Mumford, J. Shah ’89

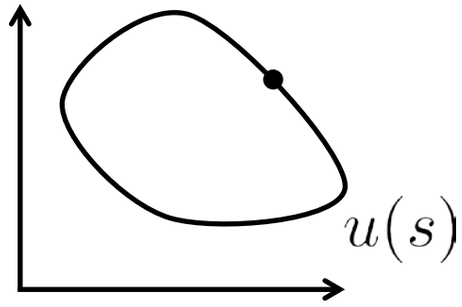
- To get these segmentations we need something more than a curve of type $u \in C^2([0,1], \mathbb{R})$

$$\Downarrow$$

$$u \in C^2([0,1], \mathbb{R}^2)$$





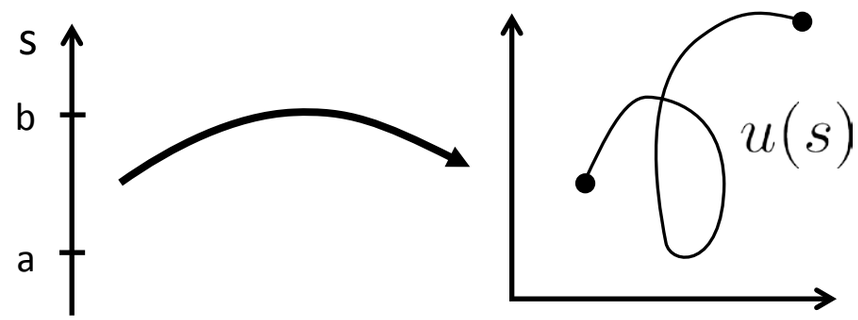


possibly closed

Snake Model v2.0

$$u \in C^2([0,1], \mathbb{R}^2)$$

$$L : C^2([0,1], \mathbb{R}^2) \to \mathbb{R}$$



$$u^* = \underset{u \in \mathbb{C}^2([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 ds$$

- The edge map $E(x,y) = \|\nabla I(x,y)\|$

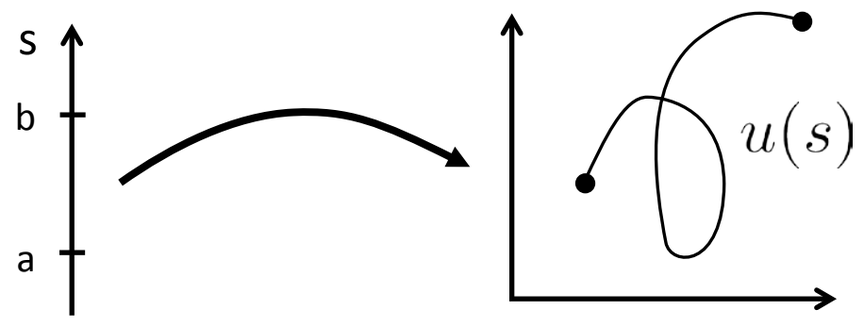
$$L(u) = \int_0^1 -E(x,u(x))^2 + \dot{u}(x)^2 dx$$

Snake 1.0

Snake Model v2.0

$$u \in C^2([0,1], \mathbb{R}^2)$$

$$L : C^2([0,1], \mathbb{R}^2) \to \mathbb{R}$$

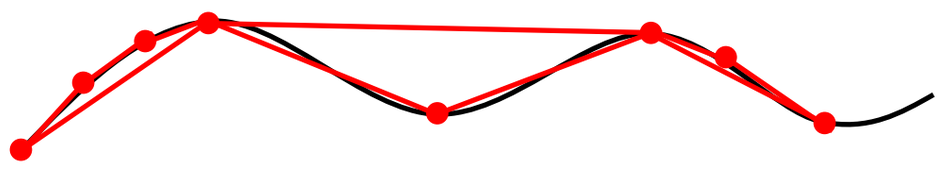


$$u^* = \underset{u \in \mathbb{C}^2([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 ds$$

$$L(u) = \int_0^1 -E(x, u(x))^2 + \dot{u}(x)^2 dx$$

Snake 1.0

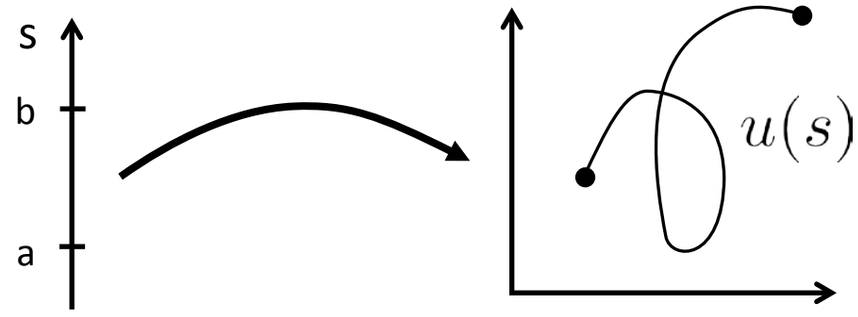
- The edge map $E(x,y) = \|\nabla I(x,y)\|$
- $\dot{u}(s)$ in this case does not enforce the curve to be smooth but, instead, it forces the parameterization to be smooth.
- if $u(s)$ is discretized, the term $\|\dot{u}(s)\|^2$ would penalize non-uniform sampling of the curves



Snake Model v2.1

$$u \in C^2([0,1], \mathbb{R}^2)$$

$$L : C^2([0,1], \mathbb{R}^2) \to \mathbb{R}$$

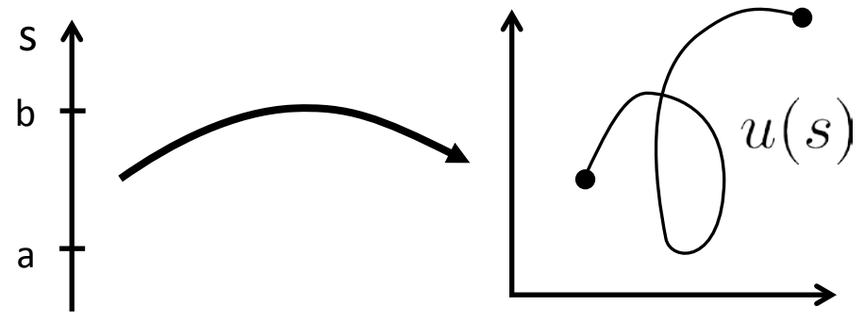


$$u^* = \underset{u \in \mathbb{C}^2([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 ds$$

Snake Model v2.1

$$u \in C^4([0,1], \mathbb{R}^2)$$

$$L : C^4([0,1], \mathbb{R}^2) \to \mathbb{R}$$



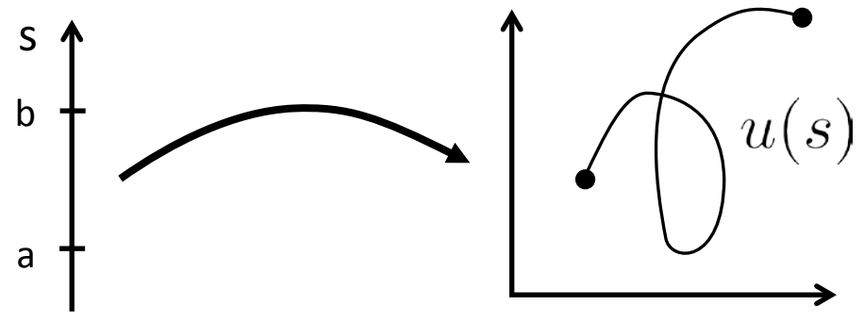
$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

- the term $\|\ddot{u}(s)\|$ is equal to the curvature of the curve, in case of uniform parameterization

Snake Model v2.1

$$u \in C^4([0,1],\mathbb{R}^2)$$

$$L : C^4([0,1],\mathbb{R}^2) \to \mathbb{R}$$



$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

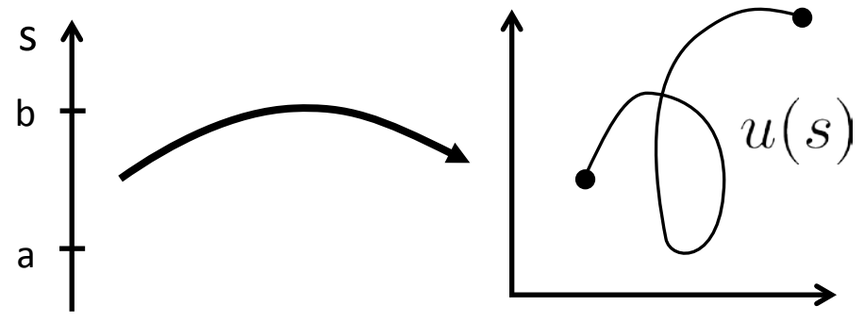
- the term $\|\ddot{u}(s)\|$ is equal to the curvature of the curve, in case of uniform parameterization

- The functional we have now is more complex than the one before

$$L(u) = \int_a^b \psi(s, u(s), \dot{u}(s), \ddot{u}(s)) ds \qquad u : [0,1] \to \mathbb{R}^2$$

Another Loss Functional #2

- Given $L : C^2([a,b], \mathbb{R}^m) \to \mathbb{R}$



$$L(u) = \int_a^b \psi(s, u_1(s), \ldots, u_m(s), \dot{u}_1(s), \ldots, \dot{u}_m(s)) ds$$

- The directional derivative is in this case

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left\langle \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s} \frac{\partial \psi}{\partial \dot{u}} \right)(s), \lambda(s) \right\rangle_{\mathbb{R}^m} ds + \left[\text{⟨ ⟩} \right]_a^b$$

$[a,b] \to \mathbb{R}^m$ $\qquad$ $[a,b] \to \mathbb{R}^m$

Another Loss Functional #2: Summary

$$L(u) = \int_a^b \psi(s, u(s), \dot{u}(s)) ds$$

Functional $L : C^2([a,b], \mathbb{R}^m) \to \mathbb{R}$

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left\langle \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s} \frac{\partial \psi}{\partial \dot{u}} \right)(s), \lambda(s) \right\rangle_{\mathbb{R}^m} ds$$

Directional derivative

$$\frac{\partial L}{\partial \lambda}(u) = \langle \nabla L(u), \lambda \rangle = \int_a^b \langle \nabla L(u)(s), \lambda(s) \rangle_{\mathbb{R}^m} ds$$

Another Loss Functional #2: Summary

$$L(u) = \int_a^b \psi(s, u(s), \dot{u}(s)) ds$$

Functional $L : C^2([a,b], \mathbb{R}^m) \to \mathbb{R}$

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left\langle \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s} \frac{\partial \psi}{\partial \dot{u}} \right)(s), \lambda(s) \right\rangle_{\mathbb{R}^m} ds$$

Directional derivative

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s} \frac{\partial \psi}{\partial \dot{u}} \right)$$

Gradient $\in C^0([a,b], \mathbb{R}^m)$

- u is a stationary point/function for L if and only if

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s} \frac{\partial \psi}{\partial \dot{u}} \right) = \mathbf{0}$$

Euler-Lagrange equation

this is the null function in $C^2([a,b], \mathbb{R}^m)$

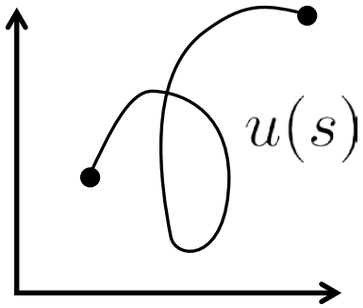
Boundary Conditions

$$\left[\left\langle \frac{\partial \psi}{\partial \dot{u}}, \lambda \right\rangle\right]_a^b = 0 \quad \Longleftrightarrow \quad \left[\sum_{i=1}^{m} \frac{\partial \psi}{\partial \dot{u}_i}(s, u_1(s), \ldots, u_m(s), \dot{u}_1(s), \ldots, \dot{u}_m(s))\lambda_i(s)\right]_a^b = 0$$

$$\begin{cases} \lambda(a) = 0 \\ \lambda(b) = 0 \end{cases}$$

Dirichlet boundary condition

(lambda is just noise on the curve u(t) which does not affect the extremes u(a) and u(b))

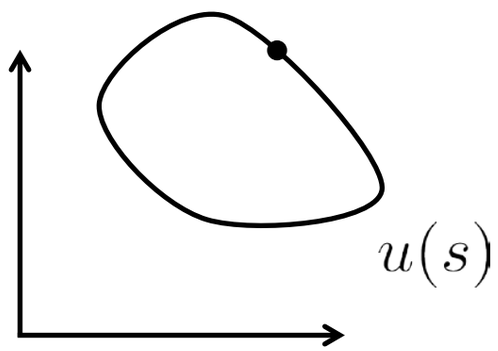


$$\begin{cases} u(a) = u(b) \\ \dot{u}(a) = \dot{u}(b) \\ \dfrac{\partial \psi}{\partial \dot{u}} \\ \lambda(a) = \lambda(b) \end{cases}$$

does not depend on s

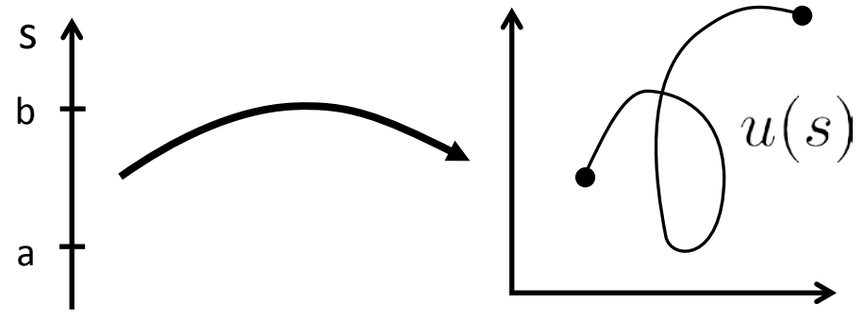
(no name) boundary condition

The curve must be close and with continuous first derivative at the closure point



Another Loss Functional #3

- Given $L : C^4([a,b], \mathbb{R}^m) \to \mathbb{R}$



$$L(u) = \int_a^b \psi(s, u(s), \dot{u}(s), \ddot{u}(s)) ds$$

- The directional derivative is in this case

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left\langle \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \dot{u}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \ddot{u}} \right), \lambda \right\rangle ds + \left[\quad \right]_a^b + \left[\quad \right]_a^b - \left[\quad \right]_a^b$$

Another Loss Functional #2: Summary

$$L(u) = \int_a^b \psi(s, u(s), \dot{u}(s), \ddot{u}(s)) ds$$

Functional $L : C^4([a,b], \mathbb{R}^m) \to \mathbb{R}$

$$\frac{\partial L}{\partial \lambda}(u) = \int_a^b \left\langle \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \dot{u}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \ddot{u}} \right), \lambda \right\rangle ds$$

Directional derivative

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \dot{u}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \ddot{u}} \right)$$

Gradient $\in C^0([a,b], \mathbb{R}^m)$

- u is a stationary point/function for L if and only if

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \dot{u}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \ddot{u}} \right) = \mathbf{0}$$

Euler-Lagrange equation

Boundary Conditions

$$\left[\left\langle \frac{\partial \psi}{\partial \dot{u}}, \lambda \right\rangle\right]_a^b + \left[\left\langle \frac{\partial \psi}{\partial \ddot{u}}, \dot{\lambda} \right\rangle\right]_a^b - \left[\left\langle \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \ddot{u}}\left(s, u(s), \dot{u}(s), \ddot{u}(s)\right), \lambda \right\rangle\right]_a^b = 0$$

$$\begin{cases} \lambda(a) = 0 \\ \lambda(b) = 0 \\ \dot{\lambda}(a) = 0 \\ \dot{\lambda}(b) = 0 \end{cases}$$

Dirichlet boundary condition

(lambda is just noise on the curve u(t) which does not affect the extremes u(a) and u(b) and their first derivatives)

(bad for our case)

$$\begin{cases} u(a) = u(b) \\ \dot{u}(a) = \dot{u}(b) \\ \ddot{u}(a) = \ddot{u}(b) \\ \dddot{u}(a) = \dddot{u}(b) \end{cases}$$

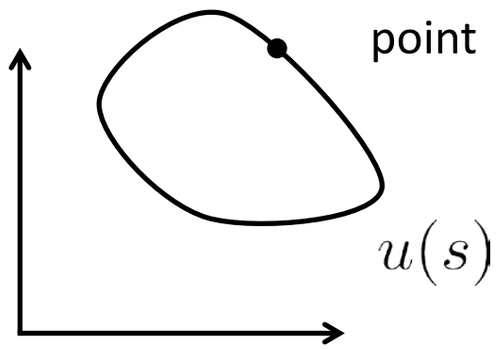
(no name) boundary condition

The curve must be close and with continuous first, second and third derivatives at the closure point

$$\frac{\partial \psi}{\partial \dot{u}}$$ does not depend on s

$$\frac{\partial \psi}{\partial \ddot{u}}$$ does not depend on s

$$\lambda(a) = \lambda(b) \quad \dot{\lambda}(a) = \dot{\lambda}(b)$$



Snake Model v2.1

- Now we can solve for the snake model

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

$$\begin{aligned} \text{subject to} \quad & u(0) = u(1) \\ & \dot{u}(0) = \dot{u}(1) \\ & \ddot{u}(0) = \ddot{u}(1) \\ & \dddot{u}(0) = \dddot{u}(1) \end{aligned}$$

- The gradient is $\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \dot{u}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \ddot{u}} \right)$ *

$$\nabla L(u) = -2E(u(s))\nabla E(u(s)) - \alpha\ddot{u}(s) + \beta\ddddot{u}(s)$$

$$E(x,y) = \|\nabla I(x,y)\|$$

Snake Model v2.1

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

$$\text{subject to} \quad \begin{aligned} u(0) &= u(1) \\ \dot{u}(0) &= \dot{u}(1) \\ \ddot{u}(0) &= \ddot{u}(1) \\ \dddot{u}(0) &= \dddot{u}(1) \end{aligned}$$

$$\begin{cases} u(a) = u(b) \\ \dot{u}(a) = \dot{u}(b) \\ \ddot{u}(a) = \ddot{u}(b) \\ \dddot{u}(a) = \dddot{u}(b) \\ \dfrac{\partial \psi}{\partial \dot{u}} \quad \text{does not depend on s} \\ \dfrac{\partial \psi}{\partial \ddot{u}} \quad \text{does not depend on s} \\ \lambda(a) = \lambda(b) \quad \dot{\lambda}(a) = \dot{\lambda}(b) \end{cases}$$

⇨ **(no name) boundary condition**

Snake Model v2.1

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

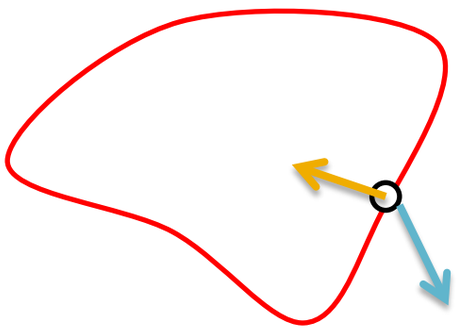
$$\text{subject to} \quad \begin{aligned} u(0) &= u(1) \\ \dot{u}(0) &= \dot{u}(1) \\ \ddot{u}(0) &= \ddot{u}(1) \\ \dddot{u}(0) &= \dddot{u}(1) \end{aligned}$$

⇩

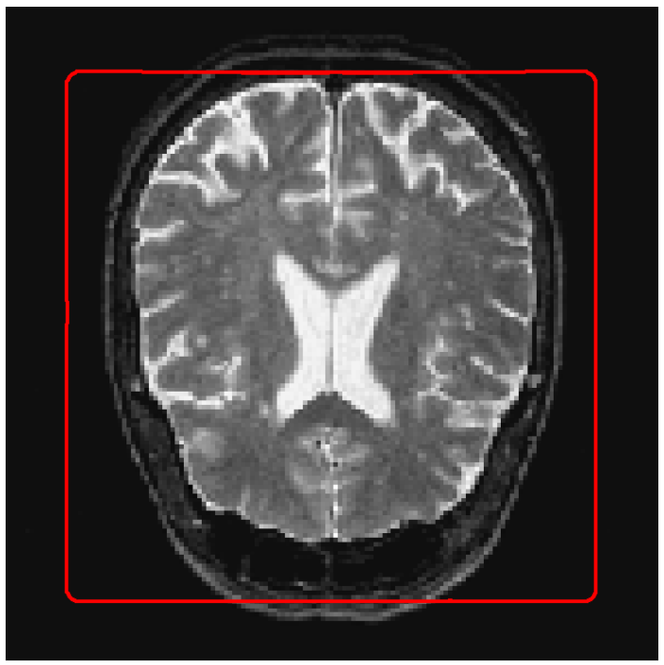
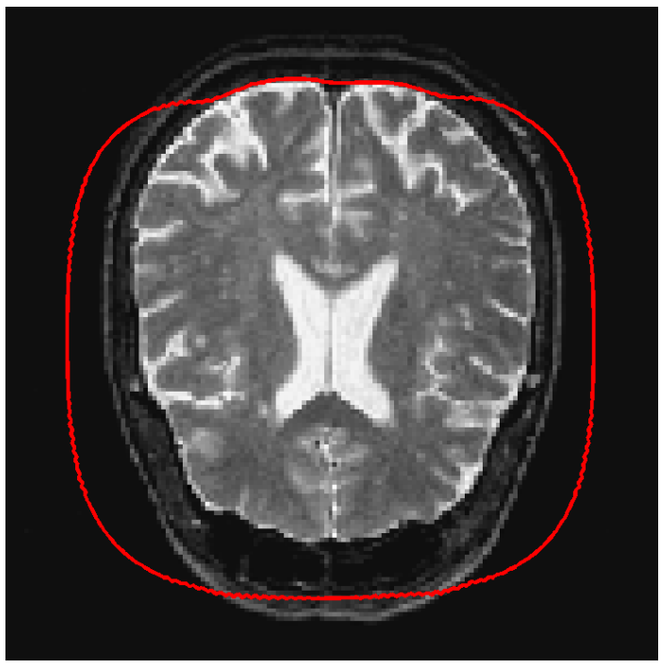
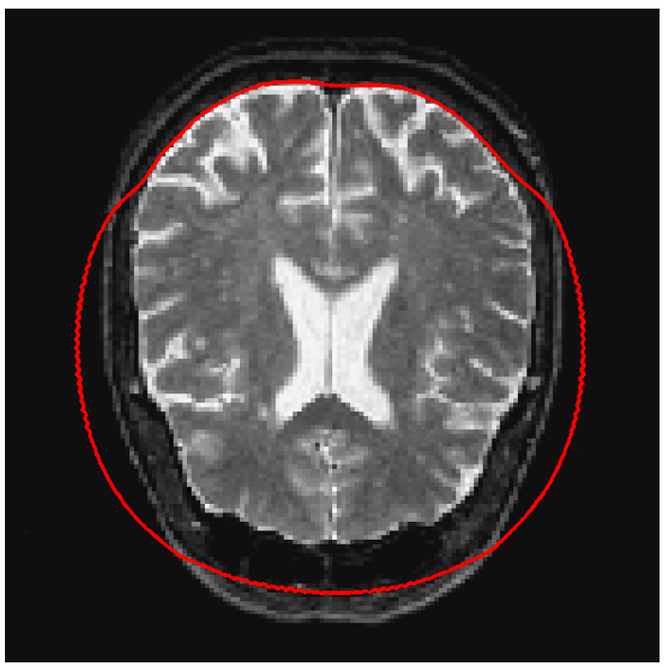
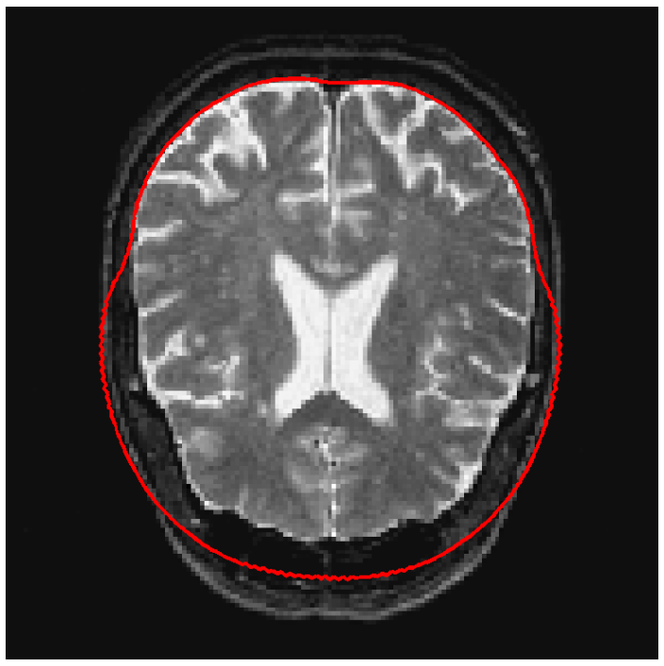
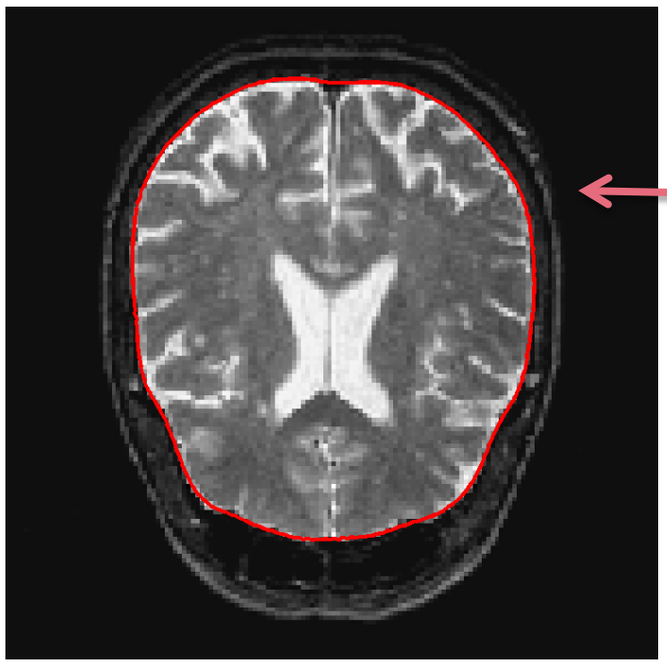
$$\nabla L(u) = \underbrace{-2E(u(s))\nabla E(u(s))}_{\text{External “force”}} \underbrace{- \alpha\ddot{u}(s) + \beta\ddddot{u}(s)}_{\text{Internal “force”}}$$

$\in \mathbb{R}^2$ $\in \mathbb{R}^2$

2D vector field in the image space



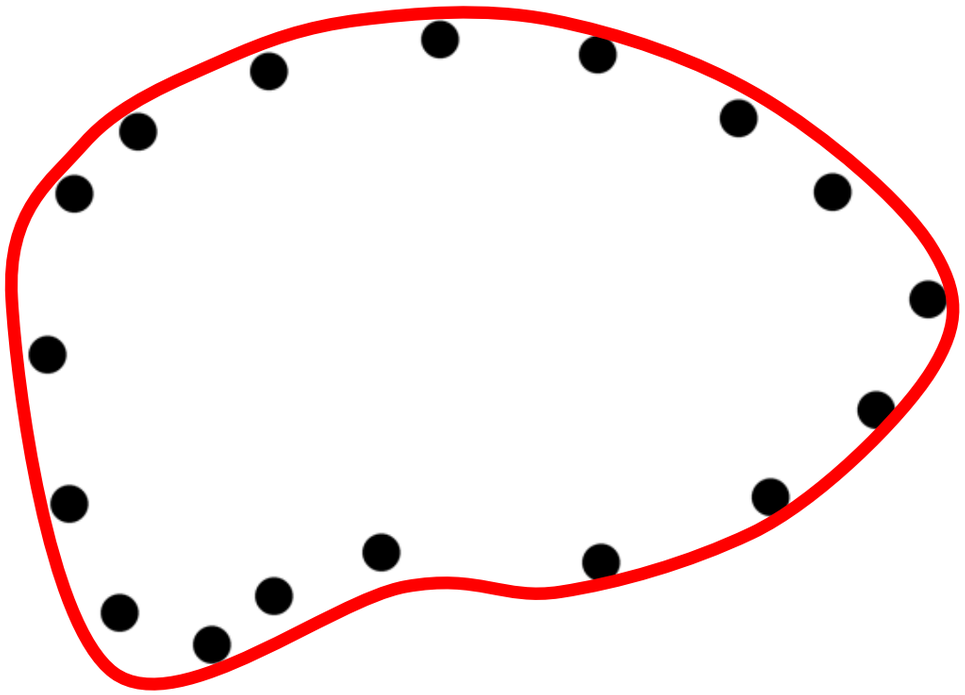
Snake Model v2.1: Evolution example

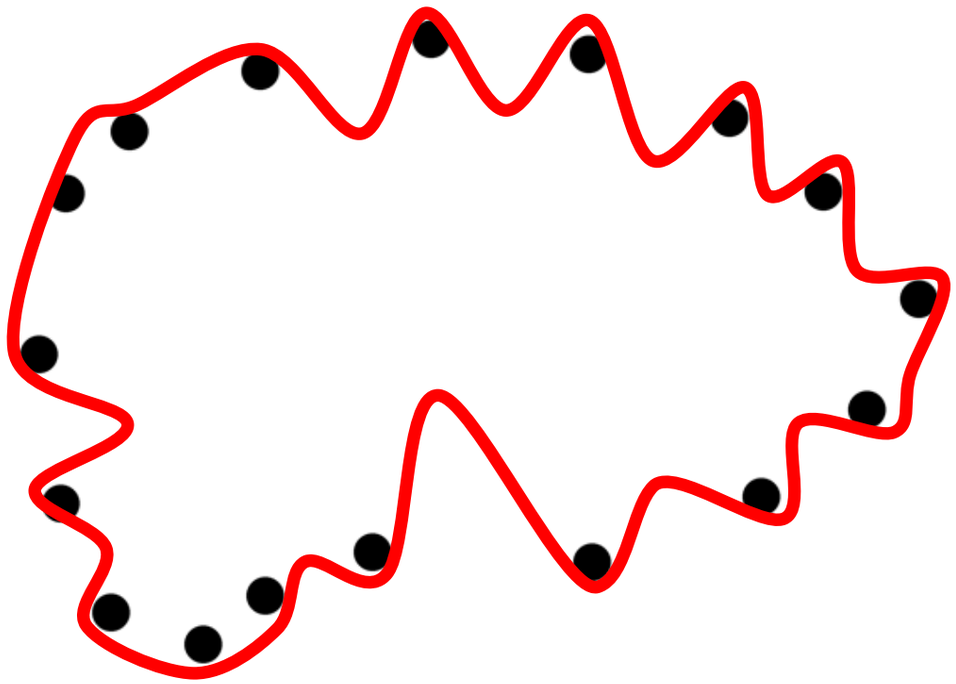






- **Robust** to noise

- α and β shrink the snake till a strong edge is found
- The higher α and β are, the stronger has to be the edge (these have to be tuned)

Lack of information

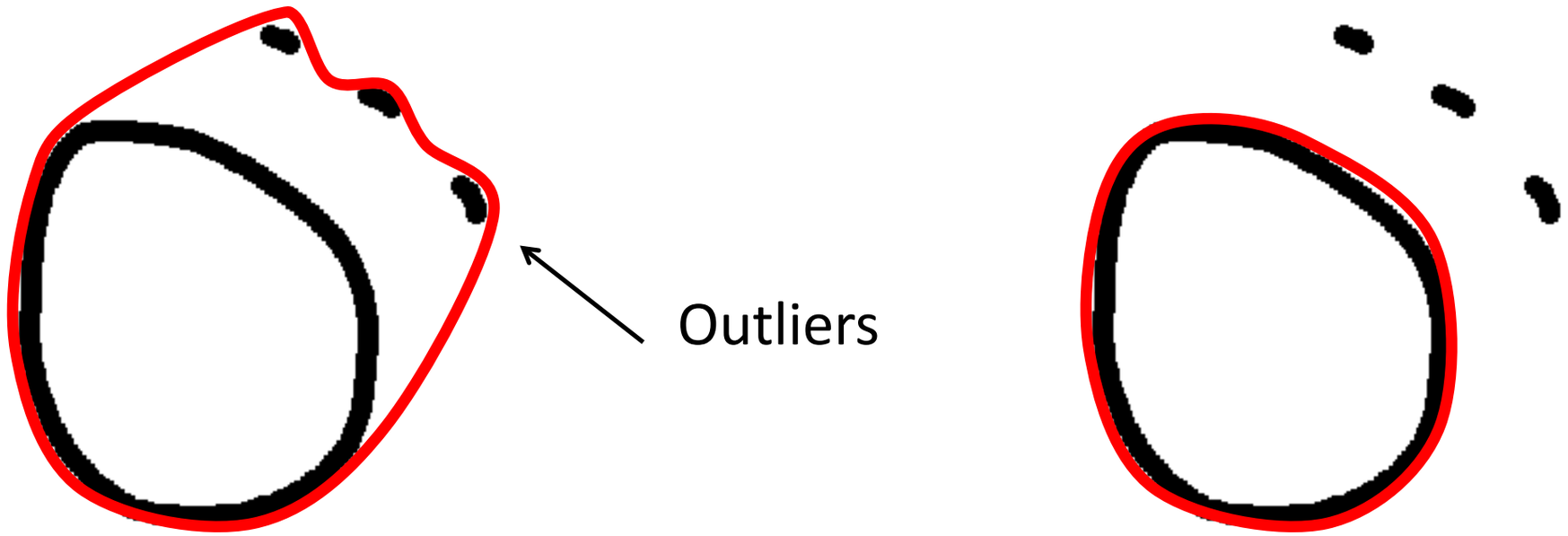




The result higher depends on the chosen prior, i.e. the values of α and β

The higher they are, the more robust the method is to lack of information

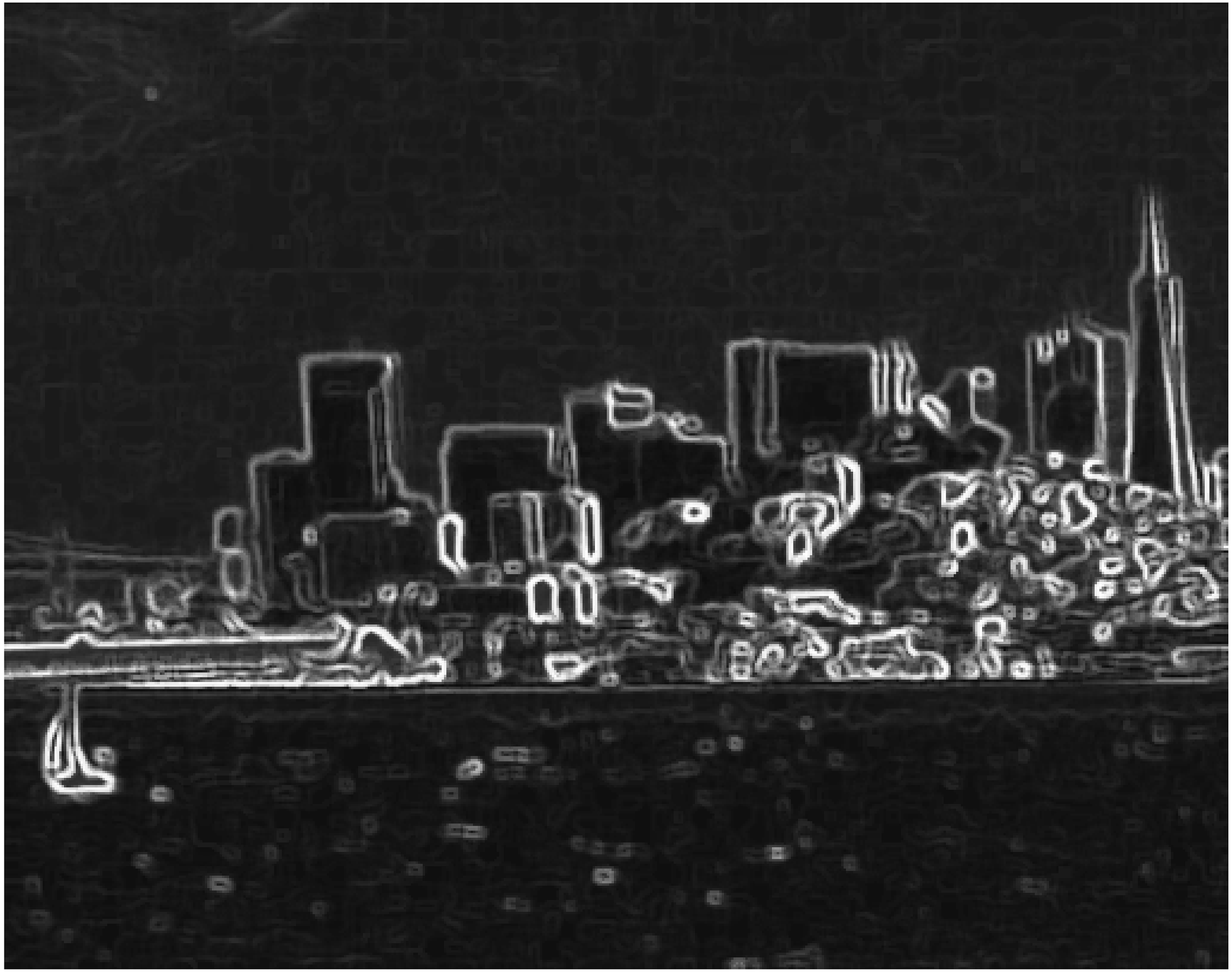
Outliers



Again, the result depends on the choice of α and β

The higher they are, the more robust the method is to outliers

Outliers, Noise, Lack of Information



Outliers, Noise, Lack of Information

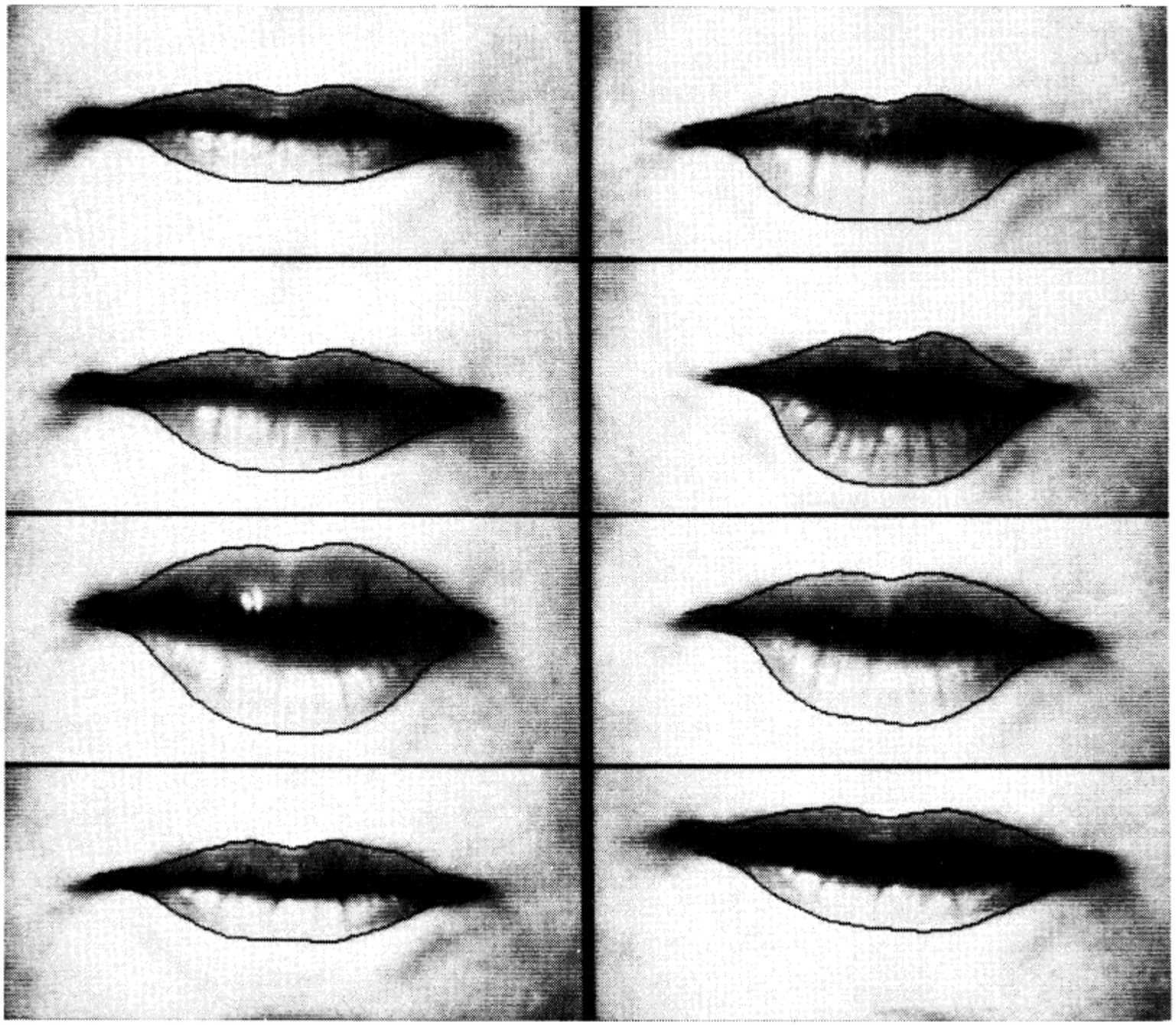
- Smooth curves make the approach robust to noise, “outliers” (still an L2), and lack of information

- Too smooth!! is this what we want? No more details

- real detail vs. outlier/noise (L2 smoothness term)

Tracking dynamic object

- Very fast to track dynamic elements, using the previous frame as initialization of the snake
- with very few iterations it will converge



Problems

- It inherits all the problems of a gradient descent technique
  - it converges to a local minima
  - these local minima can densely populate the space of solution
    - especially for high dimensional problems
    - therefore, once we get out from one local minimum it will fall right after into another one.
  - convergence can slow down in case of flat areas of the functional (line search) *

Problems

- It inherits all the problems of a gradient descent technique
  - it converges to a local minima
  - these local minima can densely populate the space of solution
    - especially for high dimensional problems
    - therefore, once we get out from one local minimum it will fall right after into another one.
  - convergence can slow down in case of flat areas of the functional (line search)

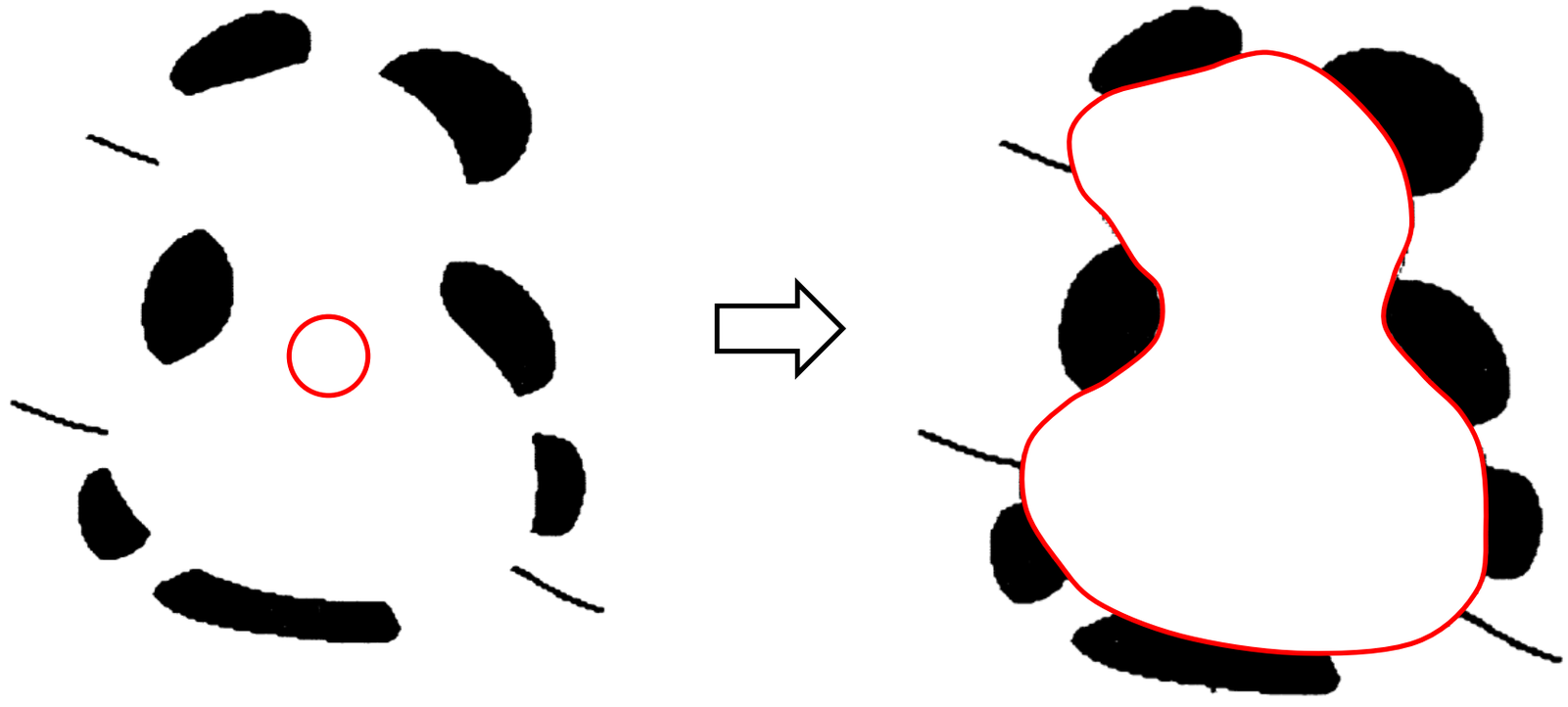
- The way the problem is formulated can introduce
  - multiple global minima
  - The term $-E(u(s))^2$ introduces a lot of flat areas in the functional (gradient information is only local not global) [Resort to descent technique, not gradient] *
  - α and β have to be decided a priori
    - internal forces tend to shrink the snake into a single point (the higher are these constant the higher is this effect)

Problems

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

$$\begin{aligned} \text{subject to} \quad & u(0) = u(1) \\ & \dot{u}(0) = \dot{u}(1) \\ & \ddot{u}(0) = \ddot{u}(1) \\ & \dddot{u}(0) = \dddot{u}(1) \\ & u \neq \emptyset \end{aligned}$$

- The global/local minimum of this functional is the **empty set** -> L(u) = 0
- Our gradient descent approach implicitly excluded it as a possible solution by just stopping to the first local minimum, but



Mathematical Foundations of
Computer Graphics and Vision

Variational Methods III

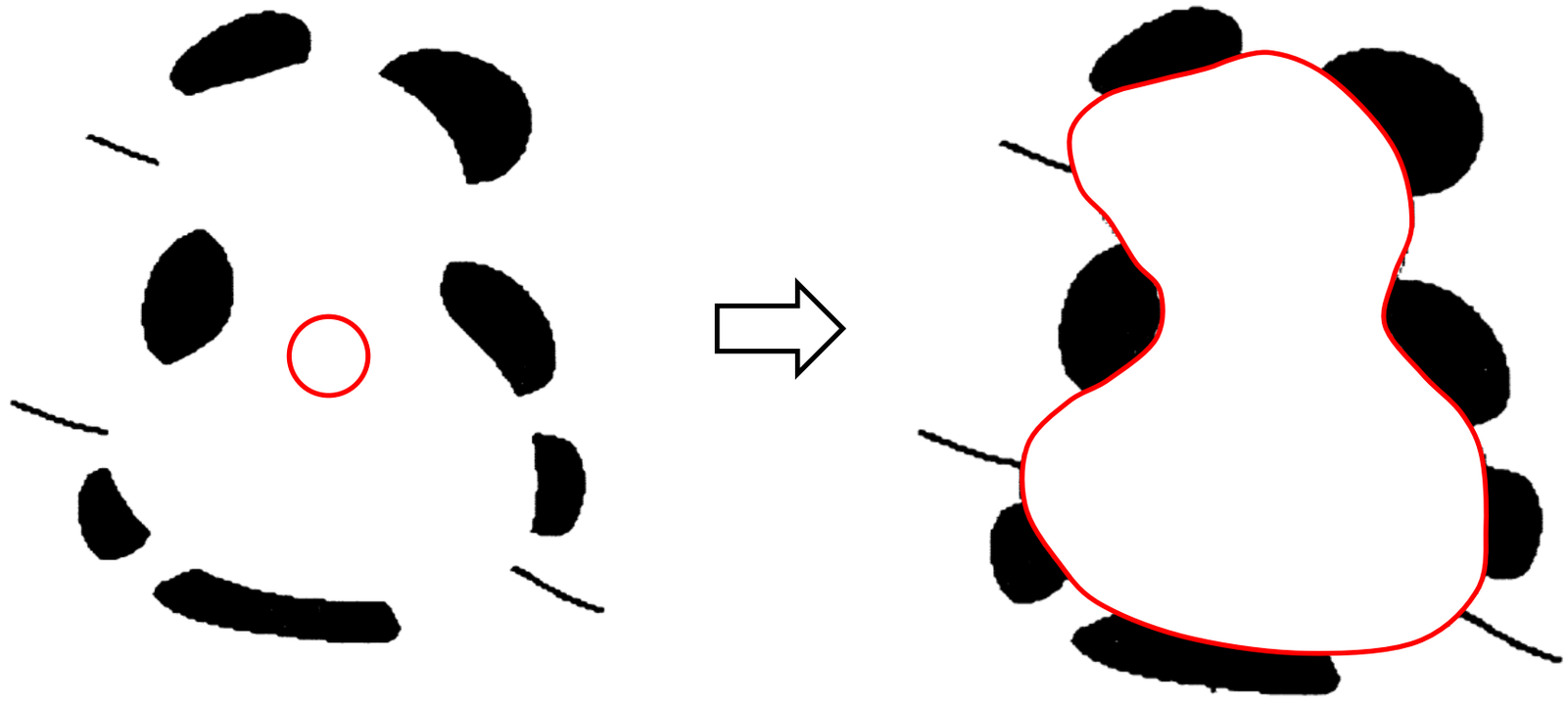
Luca Ballan

Problems

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 -E(u(s))^2 + \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

$$\begin{aligned} \text{subject to} \quad & u(0) = u(1) \\ & \dot{u}(0) = \dot{u}(1) \\ & \ddot{u}(0) = \ddot{u}(1) \\ & \dddot{u}(0) = \dddot{u}(1) \\ & u \neq \emptyset \end{aligned}$$

- The global/local minimum of this functional is the **empty set** -> L(u) = 0
- Our gradient descent approach implicitly excluded it as a possible solution by just stopping to the first local minimum, but



The Balloon Term

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 \dots ds + \underbrace{\gamma \int_{int(u)} dp}$$

Balloon term

- It measures the area inside the curve u (integral over all the internal points of u)

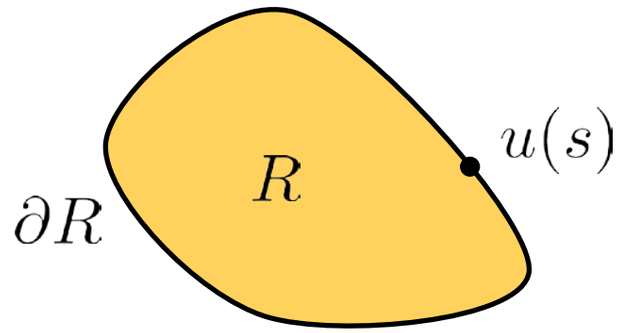
- $\gamma > 0$ penalizes big areas (force it too be small), $\gamma < 0$ penalizes small areas (force it too be big and also non-null)

- It is an integral over the interior points: Euler-Lagrange equation is not applicable

The Balloon Term

The **Green's Theorem** (on the Cartesian plane)

$$\int_R (\nabla \times v)\, dp = \int_{\partial R} v \cdot dp$$



$v : \mathbb{R}^2 \to \mathbb{R}^2$

Vector field

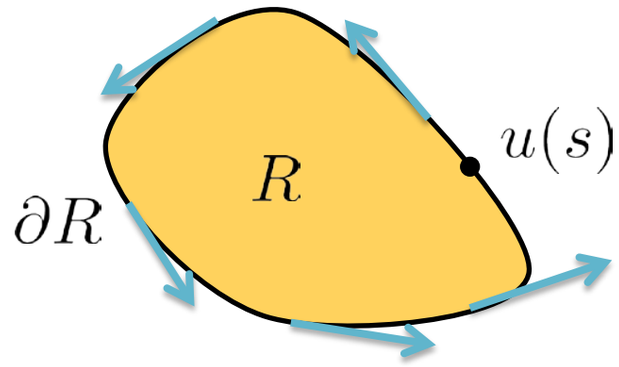
The curl of a vector field is a scalar field

$$\nabla \times v = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}$$

The Balloon Term

The **Green's Theorem** (on the Cartesian plane)

$$\int_R (\nabla \times v)\, dp = \underbrace{\int_{\partial R} v \cdot dp}$$



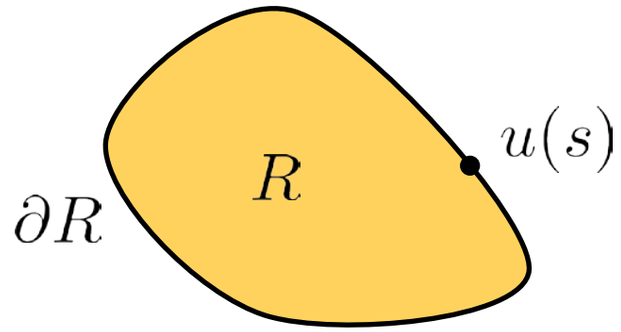
$v : \mathbb{R}^2 \to \mathbb{R}^2$

Vector field

Oriented integration on an oriented curve (counterclockwise)

The Balloon Term

The **Green's Theorem** (on the Cartesian plane)



$$v : \mathbb{R}^2 \to \mathbb{R}^2$$

$$\int_R (\nabla \times v)\, dp = \int_{\partial R} v \cdot dp$$

$$\int_R \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \right) dxdy = \int_{\partial R} v_x dx + v_y dy$$

$$= \int_0^1 v(u(s)) \cdot \dot{u}(s) \quad ds$$

$$\int_{int(u)} dp \quad \Leftarrow \quad \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} = 1$$

The Balloon Term 

$$v(x,y) = \left(-\frac{y}{2}, \frac{x}{2}\right) \quad \Rightarrow \quad \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} = 1$$

$$\Downarrow$$

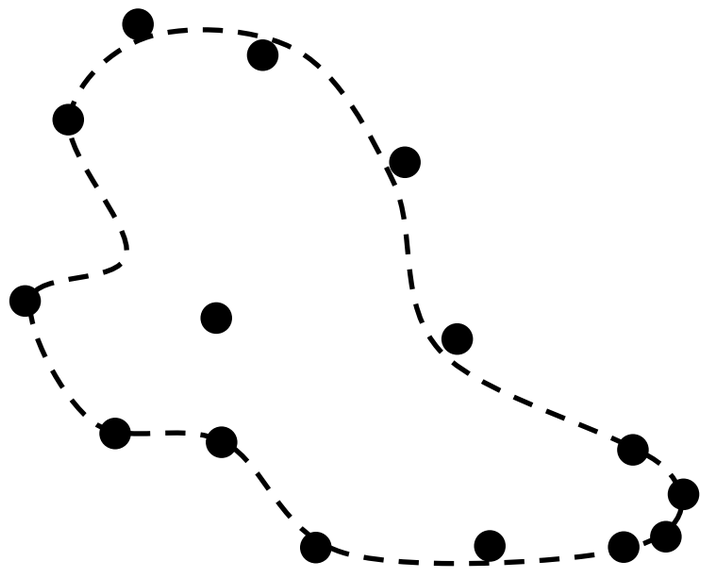
$$\int_{int(u)} dp = \int_0^1 v(u(s)) \cdot \dot{u}(s)\ ds = \frac{1}{2} \int_0^1 (-u_y(s), u_x(s)) \cdot \dot{u}(s)\ ds$$

Green's Theorem

Euler-Lagrange can be applied on this functional

Magically it will result in a force pushing the contour in/our along its normal

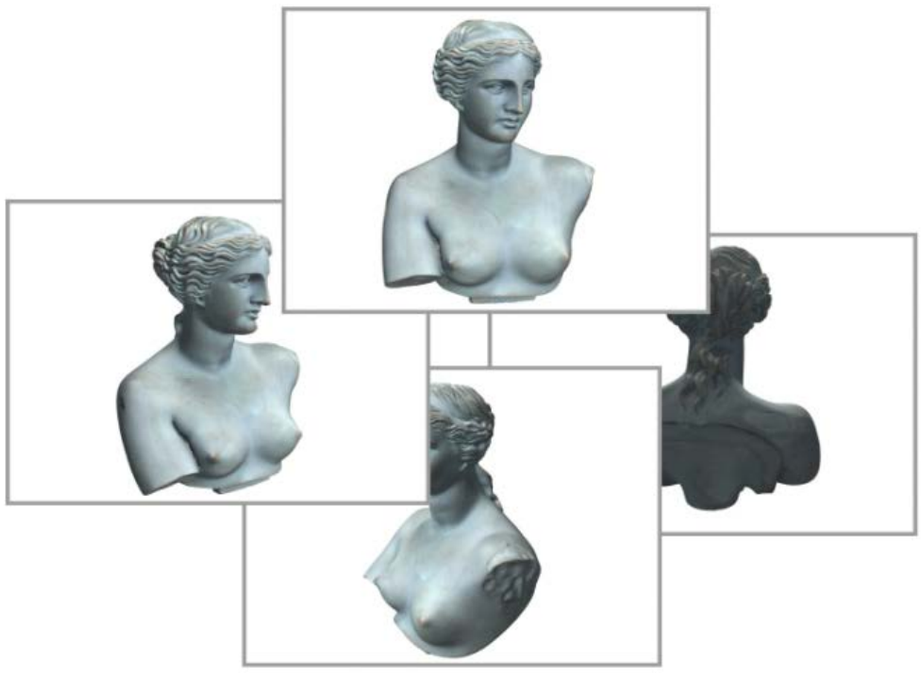
Curve Fitting



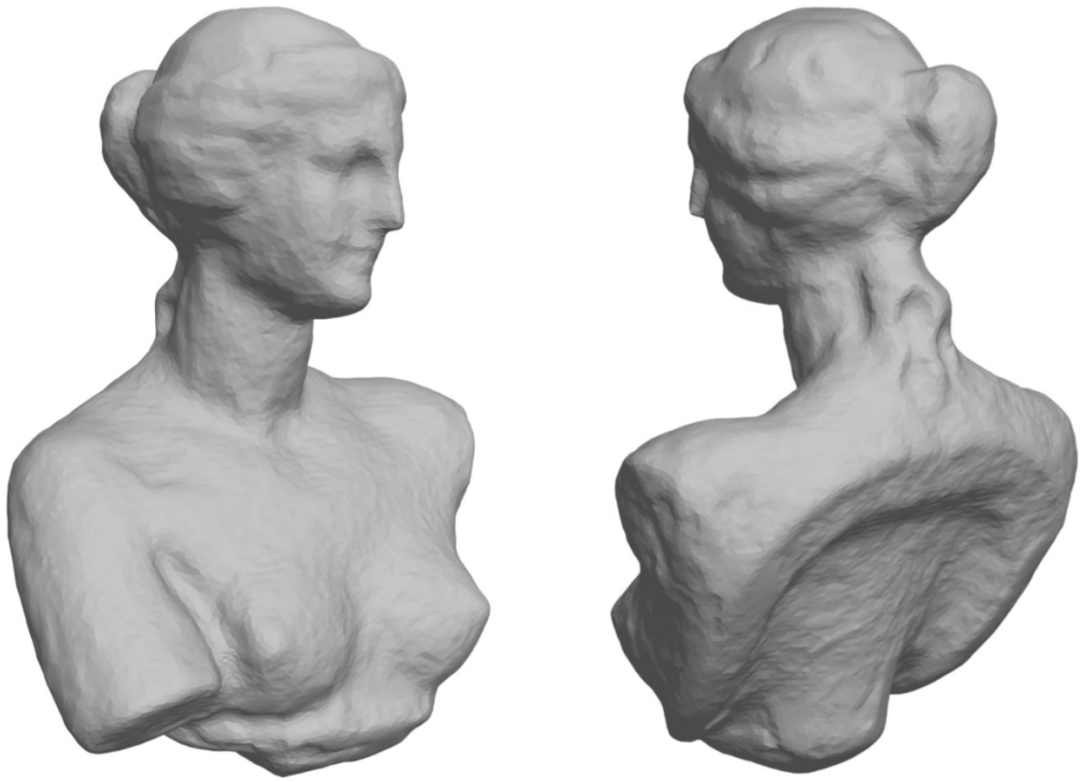
- In: Set of points $= \{q_i\}_i = \{(x_i, y_i)\}_i$

$$u^* = \underset{u \in \mathbb{C}^4([0,1],\mathbb{R}^2)}{\arg\min} \int_0^1 \frac{\alpha}{2}\|\dot{u}(s)\|^2 + \frac{\beta}{2}\|\ddot{u}(s)\|^2 ds$$

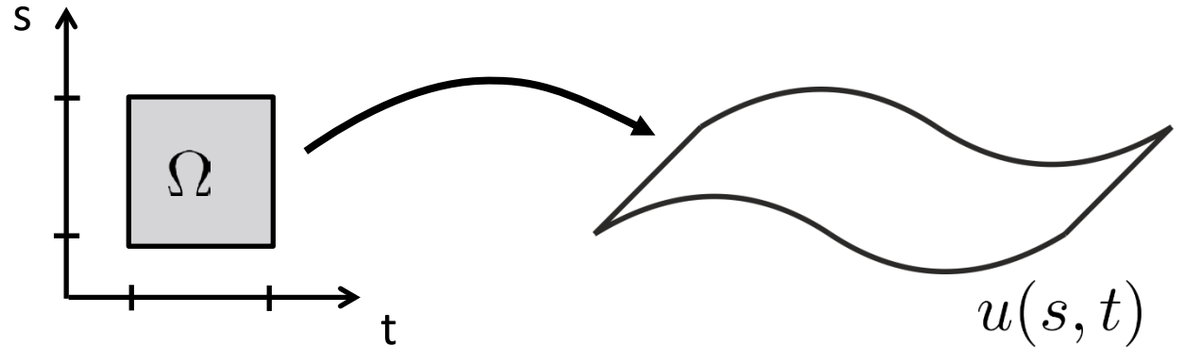
3D Surface Reconstruction



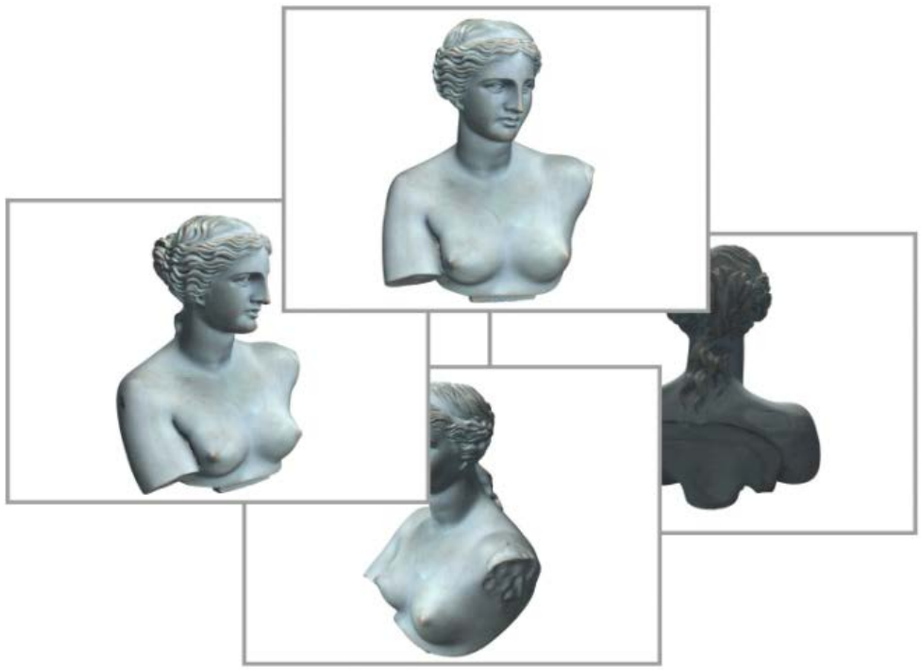
Input images

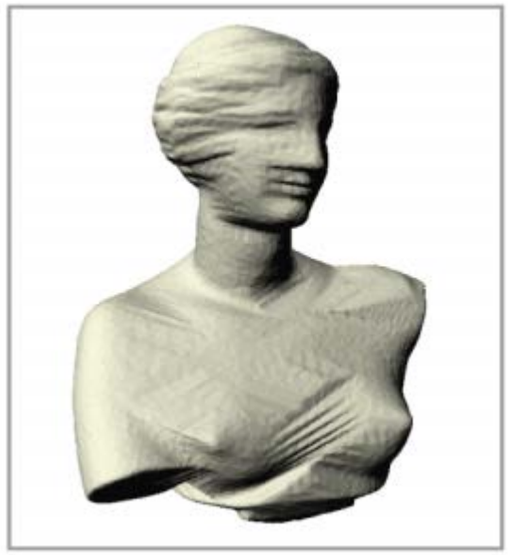


3D Surface



3D Surface Reconstruction

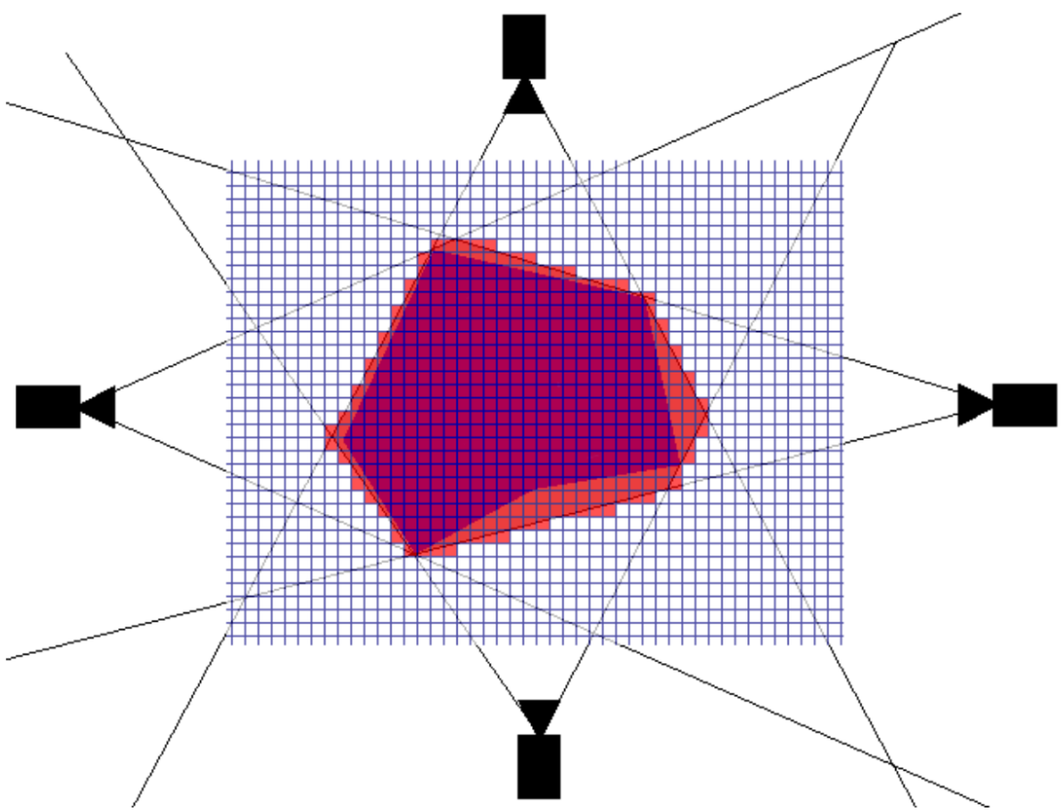




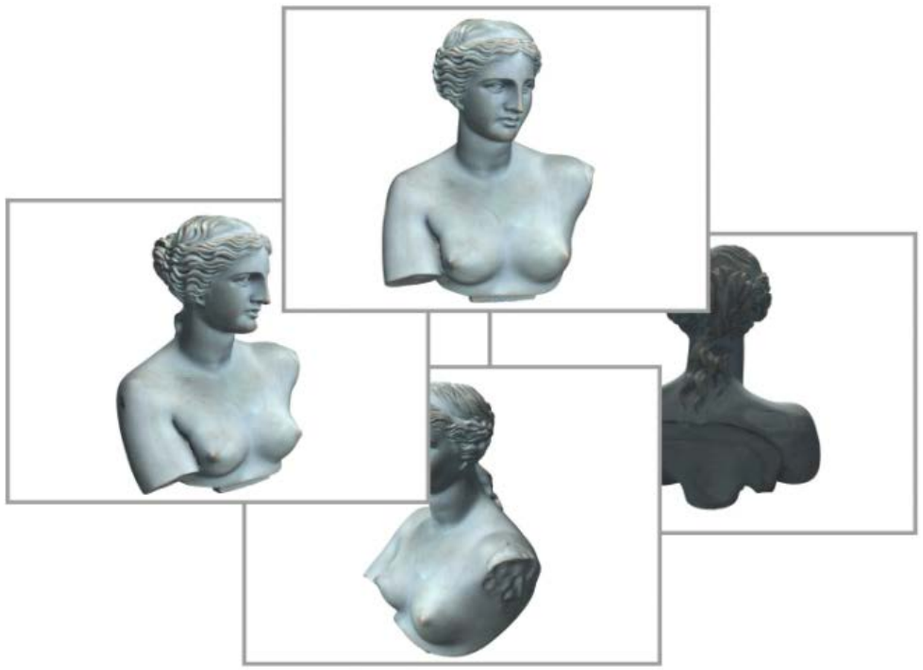
Shape from Silhouette
(visual hull)

- Surface
- do not capture convexities

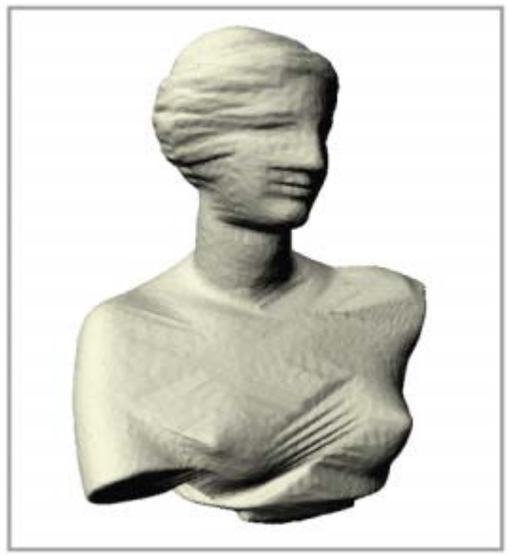
Input images



3D Surface Reconstruction



Input images



Shape from Silhouette
(visual hull)

- Surface
- do not capture convexities

Pairwise or Multi-view Stereo Matching

- Point cloud
- capture convexities

How do we fuse together the results of these two methods?

1st Approach

- Initial solution =  Visual Hull
- Optimize <u>locally</u>

$$\underset{u\in\mathbb{C}^4(\Omega,\mathbb{R}^3)}{\arg\min} \int_\Omega E(u(s,t))\ dsdt + \int_\Omega \left\|\frac{\partial u}{\partial s}\right\|^2 + \left\|\frac{\partial u}{\partial t}\right\|^2 dsdt + \int_\Omega \left\|\frac{\partial^2 u}{\partial s^2}\right\|^2 + \left\|\frac{\partial^2 u}{\partial t^2}\right\|^2 + 2\left\|\frac{\partial^2 u}{\partial s \partial t}\right\|^2 dsdt$$

Penalizes surfaces far from the point cloud

$\{q_i\}_i$

$$E(p) = \min\{\|p - q_i\|, \forall q_i\}$$

Membrane Energy
penalizes non uniformly parameterized surfaces

Thin Plate Energy
penalizes non smooth surfaces

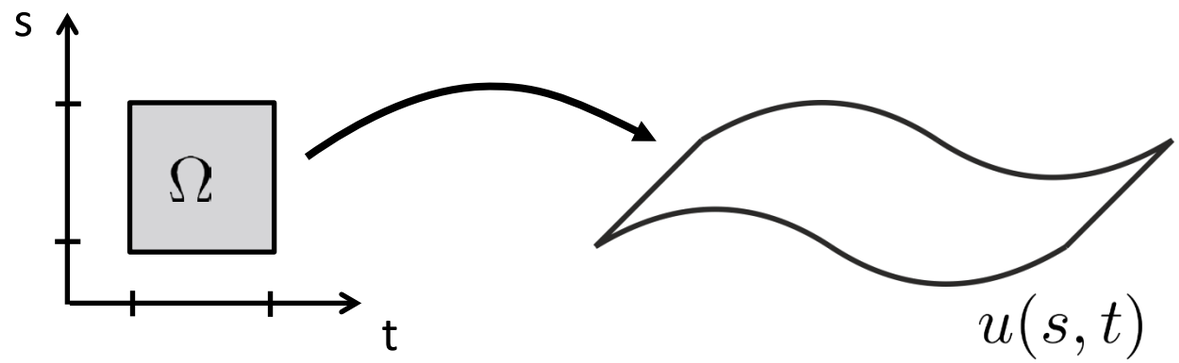
- Equal to the “total curvature”

$$\kappa = \kappa_1^2 + \kappa_2^2$$

iff the parameterization is uniform

Another Functional

- Given $L : C^4(\Omega \subseteq \mathbb{R}^2, \mathbb{R}^m) \to \mathbb{R}$



$$L(u) = \int_\Omega \psi \left(s, t, u, \frac{\partial u}{\partial s}, \frac{\partial u}{\partial t}, \frac{\partial^2 u}{\partial s^2}, \frac{\partial^2 u}{\partial t^2}, \frac{\partial^2 u}{\partial s \partial t}, \frac{\partial^2 u}{\partial t \partial s} \right) \, dsdt$$

- The gradient in this case is

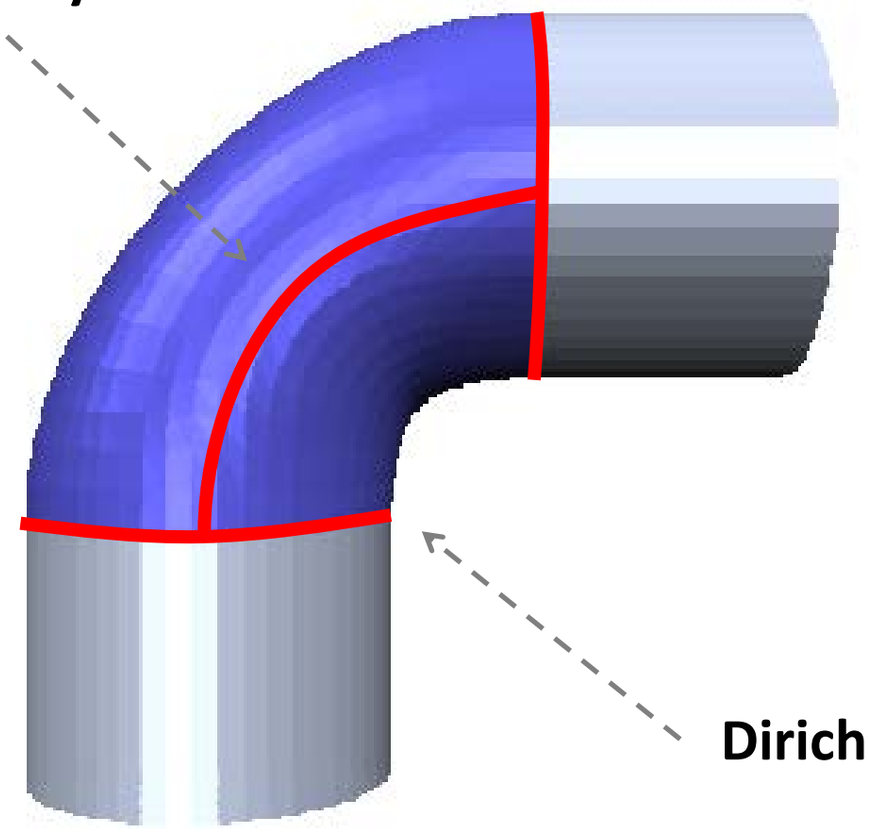
$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \frac{\partial u}{\partial s}} - \frac{\partial}{\partial t}\frac{\partial \psi}{\partial \frac{\partial u}{\partial t}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial s^2}} + \frac{\partial^2}{\partial t^2}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial t^2}} + \frac{\partial^2}{\partial s \partial t}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial s \partial t}} + \frac{\partial^2}{\partial t \partial s}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial t \partial s}} \right)$$

- Boundary conditions?

Mixed Boundary Conditions

Dirichlet boundary condition

(no name) boundary condition



Dirichlet boundary condition

3D Surface Reconstruction

$$\arg\min_{u\in\mathbb{C}^4(\Omega,\mathbb{R}^3)} \underbrace{\int_\Omega E(u(s,t))\ dsdt}_{} + \underbrace{\int_\Omega \left\|\frac{\partial u}{\partial s}\right\|^2 + \left\|\frac{\partial u}{\partial t}\right\|^2 dsdt}_{} + \int_\Omega \left\|\frac{\partial^2 u}{\partial s^2}\right\|^2 + \left\|\frac{\partial^2 u}{\partial t^2}\right\|^2 + 2\left\|\frac{\partial^2 u}{\partial s \partial t}\right\|^2 dsdt$$

$$\nabla L(u) = \quad \nabla E(u(s,t)) \qquad \textbf{?}$$

$$\nabla L(u) = \left(\frac{\partial \psi}{\partial u} - \frac{\partial}{\partial s}\frac{\partial \psi}{\partial \frac{\partial u}{\partial s}} - \frac{\partial}{\partial t}\frac{\partial \psi}{\partial \frac{\partial u}{\partial t}} + \frac{\partial^2}{\partial s^2}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial s^2}} + \frac{\partial^2}{\partial t^2}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial t^2}} + \frac{\partial^2}{\partial s \partial t}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial s \partial t}} + \frac{\partial^2}{\partial t \partial s}\frac{\partial \psi}{\partial \frac{\partial^2 u}{\partial t \partial s}} \right)$$

3D Surface Reconstruction

$$\underset{u\in\mathbb{C}^4(\Omega,\mathbb{R}^3)}{\arg\min} \underbrace{\int_\Omega E(u(s,t))\ dsdt}_{\nabla E(u(s,t))} + \underbrace{\int_\Omega \left\|\frac{\partial u}{\partial s}\right\|^2 + \left\|\frac{\partial u}{\partial t}\right\|^2 dsdt}_{-2\nabla^2 u} + \underbrace{\int_\Omega \left\|\frac{\partial^2 u}{\partial s^2}\right\|^2 + \left\|\frac{\partial^2 u}{\partial t^2}\right\|^2 + 2\left\|\frac{\partial^2 u}{\partial s \partial t}\right\|^2 dsdt}_{2\nabla^4 u}$$

$$\nabla L(u) = \quad \nabla E(u(s,t)) \qquad -2\nabla^2 u \qquad 2\nabla^4 u$$

Discretizations:

$$\nabla = \left(\frac{\partial}{\partial s}, \frac{\partial}{\partial t}\right)$$ **Gradient** operator

$$\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial s^2} + \frac{\partial^2}{\partial t^2}$$ **Laplace** operator $\xrightarrow{\tilde{=}}$ **?**

$$\nabla^4 = \nabla^2\ \nabla^2 = \left(\frac{\partial^2}{\partial s^2} + \frac{\partial^2}{\partial t^2}\right)\left(\frac{\partial^2}{\partial s^2} + \frac{\partial^2}{\partial t^2}\right) = \frac{\partial^4}{\partial s^4} + \frac{\partial^4}{\partial t^4} + 2\frac{\partial^4}{\partial s^2 \partial t^2}$$

Bi-Laplace operator (Biharmonic operator) $\xrightarrow{\tilde{=}}$

3D Surface Reconstruction

$$\underset{u\in\mathbb{C}^4(\Omega,\mathbb{R}^3)}{\arg\min} \underbrace{\int_\Omega E(u(s,t))\ dsdt}_{\nabla E(u(s,t))} + \underbrace{\int_\Omega \left\|\frac{\partial u}{\partial s}\right\|^2 + \left\|\frac{\partial u}{\partial t}\right\|^2 dsdt}_{-2\nabla^2 u} + \underbrace{\int_\Omega \left\|\frac{\partial^2 u}{\partial s^2}\right\|^2 + \left\|\frac{\partial^2 u}{\partial t^2}\right\|^2 + 2\left\|\frac{\partial^2 u}{\partial s \partial t}\right\|^2 dsdt}_{2\nabla^4 u}$$

$$\nabla L(u) = \quad \nabla E(u(s,t)) \qquad -2\nabla^2 u \qquad 2\nabla^4 u$$

Discretizations:

$$\nabla = \left(\frac{\partial}{\partial s}, \frac{\partial}{\partial t}\right)$$ **Gradient** operator

$$\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial s^2} + \frac{\partial^2}{\partial t^2}$$ **Laplace** operator $\xrightarrow{\tilde{=}}$ $\tilde{\nabla}^2(v) = \frac{1}{\#N(v)}\left(\sum_{i\in N(v)} v_i\right) - v$

$$\nabla^4 = \nabla^2\ \nabla^2 = \left(\frac{\partial^2}{\partial s^2} + \frac{\partial^2}{\partial t^2}\right)\left(\frac{\partial^2}{\partial s^2} + \frac{\partial^2}{\partial t^2}\right) = \frac{\partial^4}{\partial s^4} + \frac{\partial^4}{\partial t^4} + 2\frac{\partial^4}{\partial s^2 \partial t^2}$$

Bi-Laplace operator (Biharmonic operator) $\xrightarrow{\tilde{=}}$ $\tilde{\nabla}^2(v) = \tilde{\nabla}\left(\tilde{\nabla}(v)\right)$

Observation

$$\arg\min_{u \in \mathbb{C}^4(\Omega,\mathbb{R}^3)} \int_\Omega E(u(s,t))\, dsdt + \int_\Omega \left\| \frac{\partial u}{\partial s} \right\|^2 + \left\| \frac{\partial u}{\partial t} \right\|^2 dsdt + \int_\Omega \left\| \frac{\partial^2 u}{\partial s^2} \right\|^2 + \left\| \frac{\partial^2 u}{\partial t^2} \right\|^2 + 2 \left\| \frac{\partial^2 u}{\partial s \partial t} \right\|^2 dsdt$$

$$\nabla L(u) = \nabla E(u(s,t)) \qquad -2\nabla^2 u \qquad 2\nabla^4 u$$

Mesh smoothing

(filter the mesh using a gaussian filter)
[Taubin 95]

$$(\lambda + \mu)\nabla^2 u - (\lambda\mu)\nabla^4 u$$

- One step corresponds to a Gaussian filter pass (spectral analysis)
- The global minima is the empty set

Observation

$$\underset{u\in\mathbb{C}^4(\Omega,\mathbb{R}^3)}{\arg\min} \int_\Omega dsdt + \int_\Omega \left\|\frac{\partial u}{\partial s}\right\|^2 + \left\|\frac{\partial u}{\partial t}\right\|^2 dsdt + \int_\Omega \left\|\frac{\partial^2 u}{\partial s^2}\right\|^2 + \left\|\frac{\partial^2 u}{\partial t^2}\right\|^2 + 2\left\|\frac{\partial^2 u}{\partial s \partial t}\right\|^2 dsdt$$

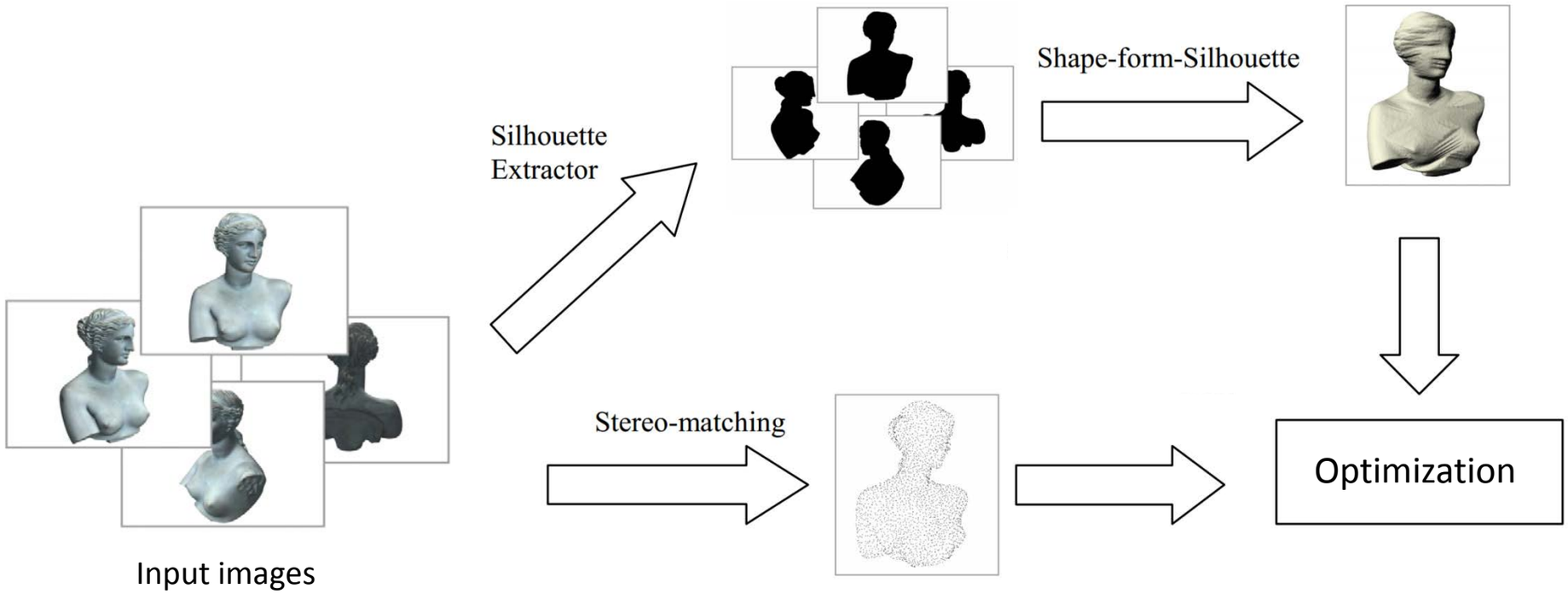
Balloon term

(volume preserving term)

[Taubin 95] -> Inflate term

- The global minima is a sphere with a similar volume

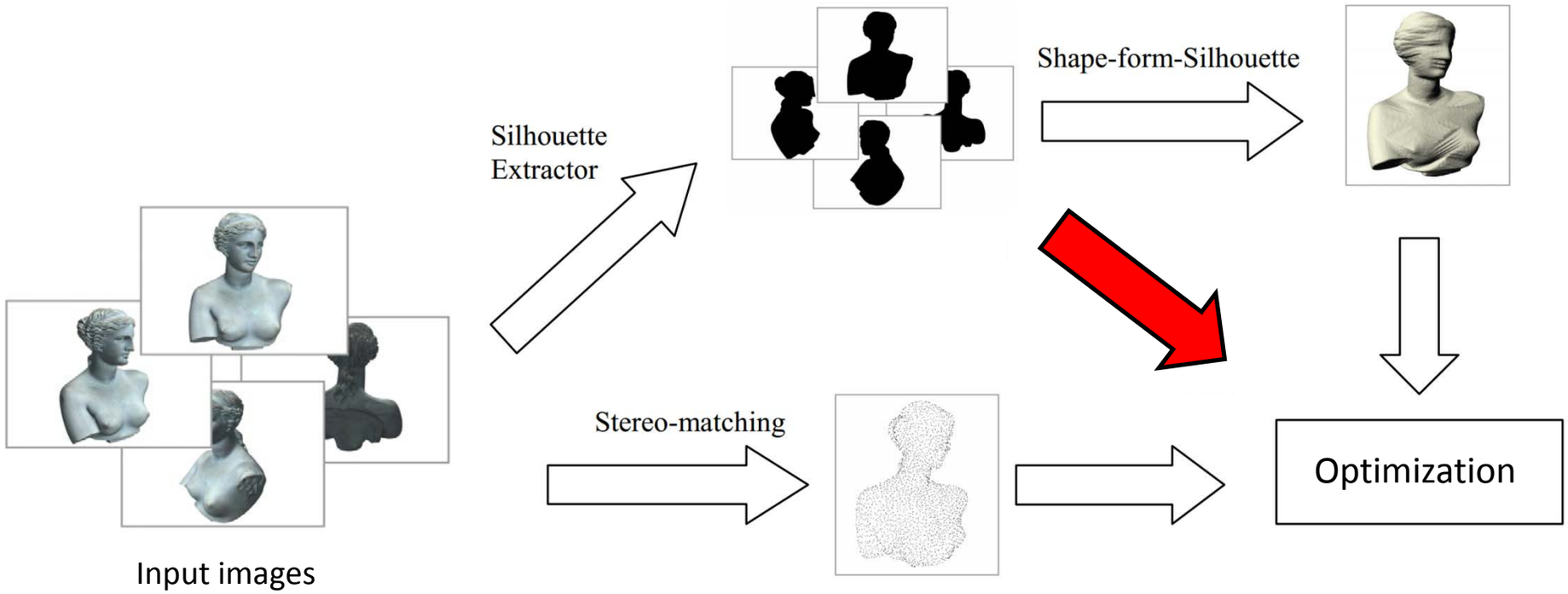
3D Surface Reconstruction



$$\arg\min_{u \in \mathbb{C}^4(\Omega,\mathbb{R}^3)} \int_\Omega E(u(s,t))\ dsdt + \int_\Omega \left\| \frac{\partial u}{\partial s} \right\|^2 + \left\| \frac{\partial u}{\partial t} \right\|^2 dsdt + \int_\Omega \left\| \frac{\partial^2 u}{\partial s^2} \right\|^2 + \left\| \frac{\partial^2 u}{\partial t^2} \right\|^2 + 2 \left\| \frac{\partial^2 u}{\partial s \partial t} \right\|^2 dsdt$$

- The local minima is a **smooth surface close to the point cloud** (computed by the stereo matching algorithm)
- If the visual hull is used to initialize the minimization, the resulting surface is also close to it

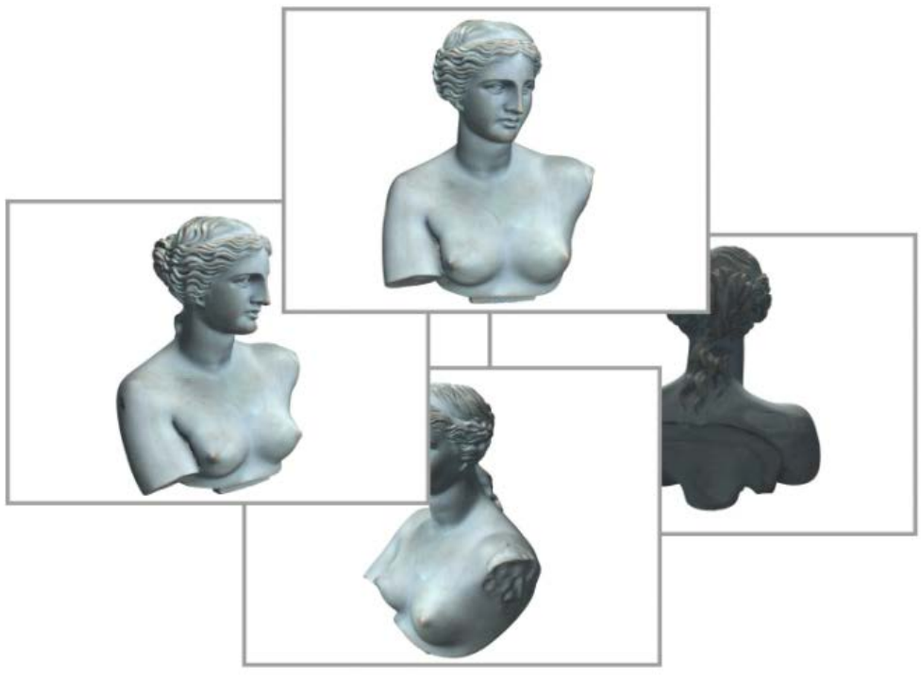
3D Surface Reconstruction



$$\arg\min_{u \in \mathbb{C}^4(\Omega,\mathbb{R}^3)} \int_\Omega E(u(s,t))\ dsdt + \int_\Omega \left\| \frac{\partial u}{\partial s} \right\|^2 + \left\| \frac{\partial u}{\partial t} \right\|^2 dsdt + \int_\Omega \left\| \frac{\partial^2 u}{\partial s^2} \right\|^2 + \left\| \frac{\partial^2 u}{\partial t^2} \right\|^2 + 2 \left\| \frac{\partial^2 u}{\partial s \partial t} \right\|^2 dsdt$$

$$+ \int_\Omega D\left(\Pi\left(u\left(s,t\right)\right)\right) dsdt$$

Generalization



Input images

Information type 3

Information type 1

Information type 2

Information fusion

Images

An image can be viewed as a function

$$u : (\Omega \subseteq \mathbb{R}^n) \rightarrow \mathbb{R}^d$$

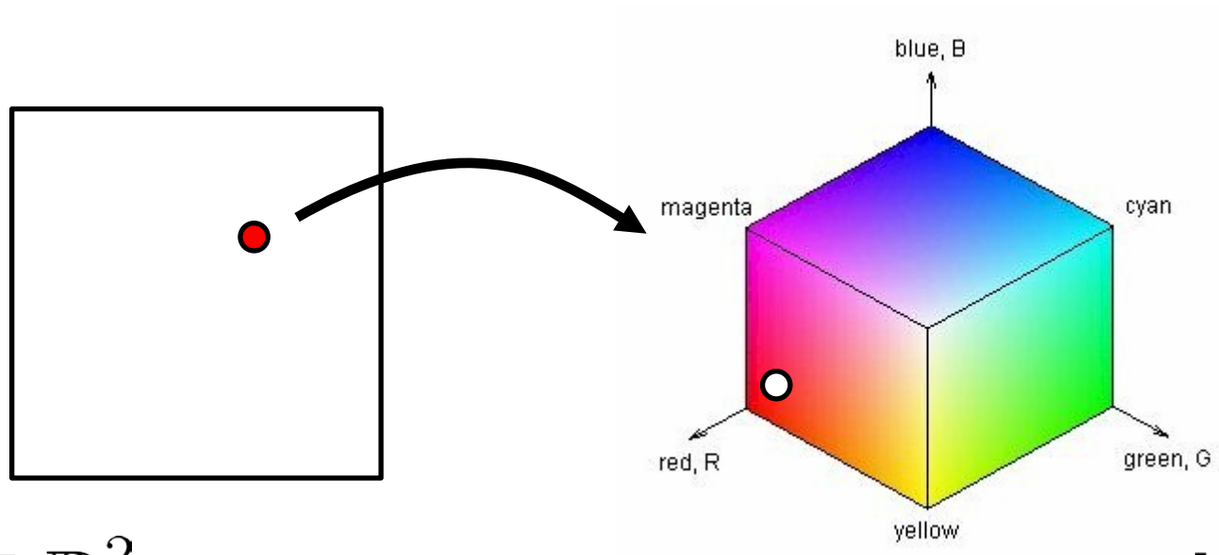
- n=2: Image
- n=3: Video (or a volumetric representation of a scene)
- n=4: Volumetric representation of a scene + time

- d=1: Brightness images/videos (or density volumes)
- d=3: Color images/videos/volumes
- d>1: Multispectral images/videos/volumes

Images

An image can be viewed as a function

$$u : (\Omega \subseteq \mathbb{R}^n) \rightarrow \mathbb{R}^d$$



$\Omega \subseteq \mathbb{R}^2$

RGB cube $[0,1]^3 \subset \mathbb{R}^3$

Image De-convolution (De-blurring)

u

$f = A * u$

- **Image De-convolution:** Given an image f and a kernel A, we aim at recovering the image u such that $f = A * u$

- In a Variational framework, both u and f are modeled as continuous functions to the a color space

$$u : \Omega \to [0,1]^3$$
$$f : \Omega \to [0,1]^3$$

RGB or YCbCr/YUV
(channels are de-correlated)

Image De-convolution (De-blurring)

u

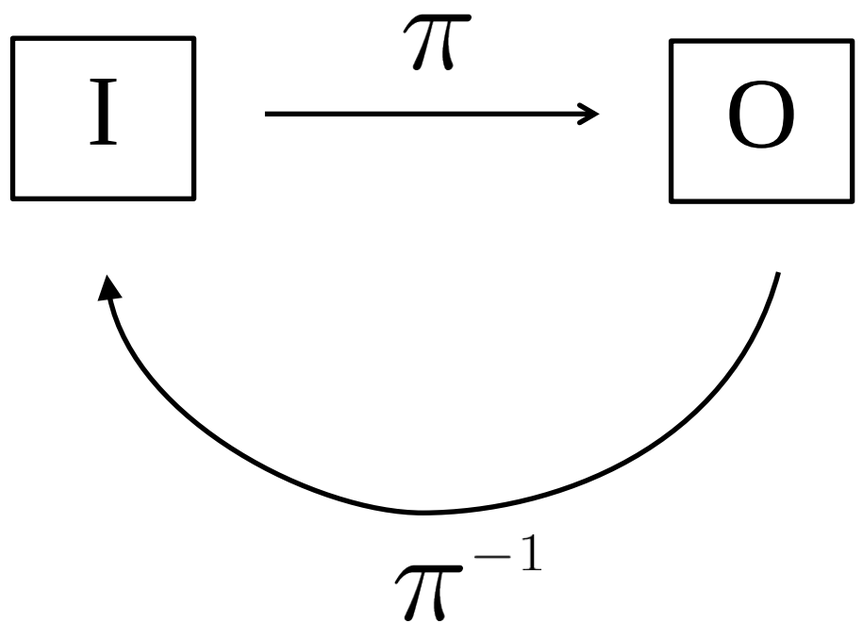
$f = A * u$

- find u such that $f = A * u$

$$\arg\min_{u} \int_{\Omega} \|A * u - f\|^2$$

Generative approach to the problem
(problem solving paradigm)

Generative Approach



Formal definition of a **problem** π

i = problem instance
$\pi(i)$ = solution of the problem instance i

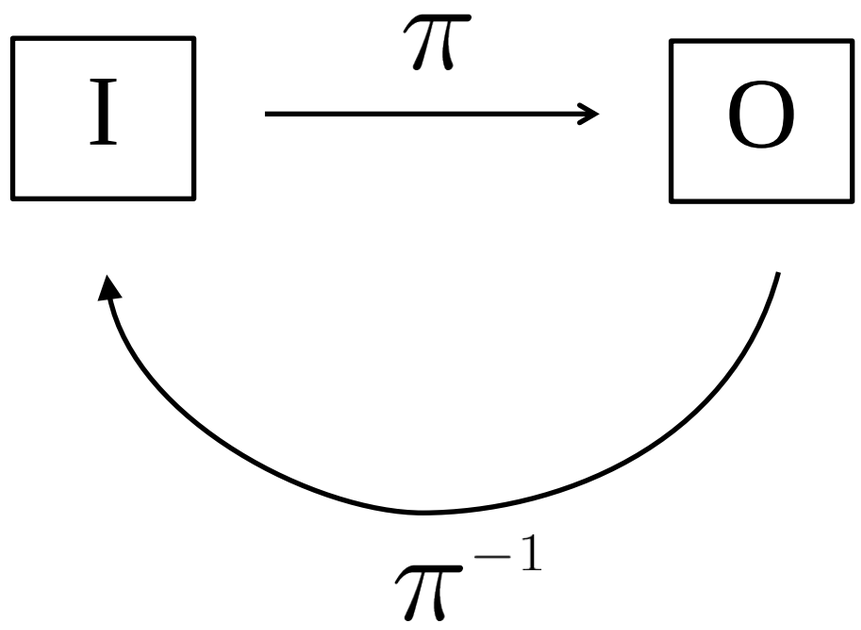
$\pi(i)$ is typically difficult to compute

$$\pi(i) = \arg\min_{o} \mathbf{L}\left(\pi^{-1}(o), i\right)$$

Generative approach to the problem

- **L** is a loss functional evaluating two input elements (e.g. a distance)

Generative Approach

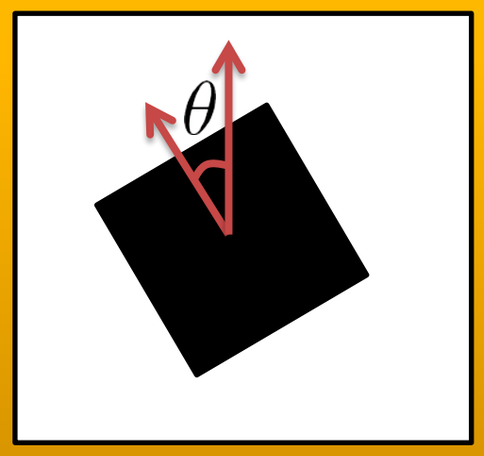


Formal definition of a **problem** π

i = problem instance
$\pi(i)$ = solution of the problem instance i

$\pi(i)$ is typically difficult to compute

- **Example:**

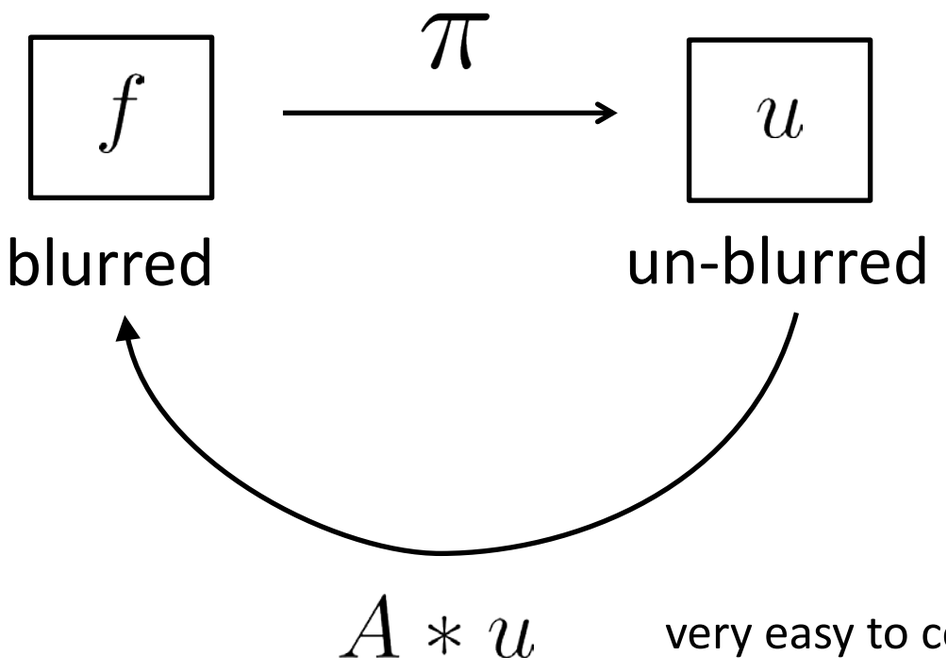
I = 

Image

π

O = orientation angle θ of the square

Generative Approach: Image De-blurring



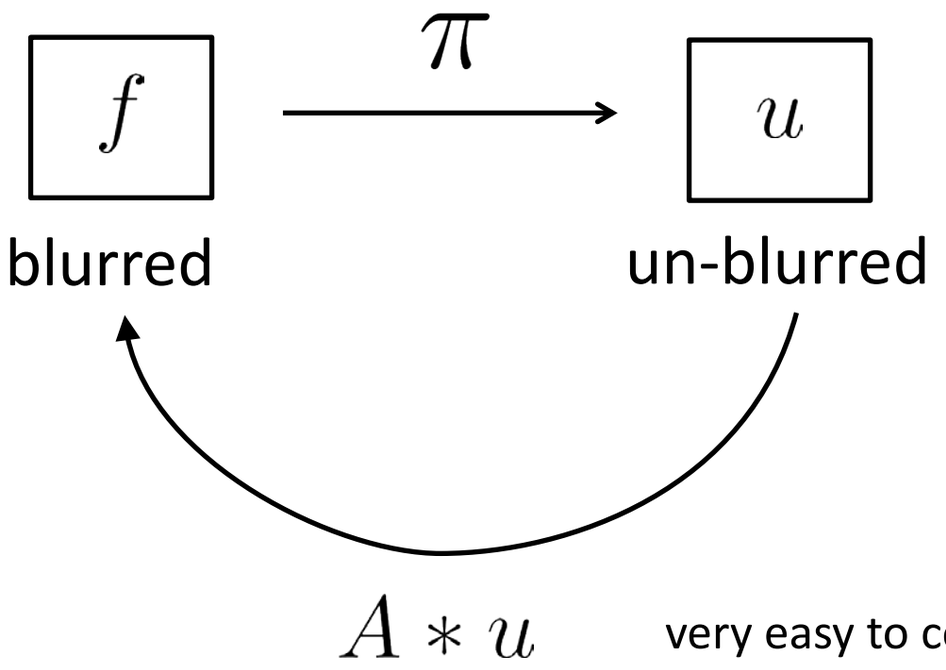
very easy to compute

$$\pi(f) = \arg\min_{u} \mathbf{L}\left(A * u, f\right)$$

Generative approach to the problem

- test (all) the u
- convolve them with A
- evaluate a cost functional with the original f

Generative Approach: Image De-blurring



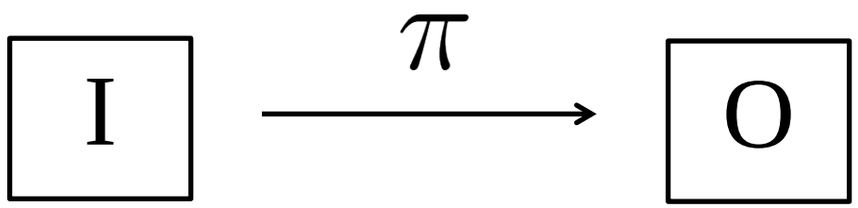
$$\pi(f) = \arg\min_u \mathbf{L}\left(A * u, f\right)$$

$$= \arg\min_u \int_\Omega \|(A * u)(p) - f(p)\|^2 dp$$

Generative approach to the problem

- What does the choice of the loss functional corresponds to?
- Is there any meaning?
- Can we choose any arbitrary functional?

Generative Approach seen as a MAP or ML



$$\pi(i) = \arg\max_o P(O = o | I = i)$$

Maximum a posteriori estimator
(the desired o is the one with maximum probability assuming the input i)

$$= \arg\min_o -\log\left[P\left(I = i | O = o\right)\right] - \log\left[P\left(O = o\right)\right]$$

uniform

Likelihood
(loss functional **L**)

Prior on o
(additional information about our output)

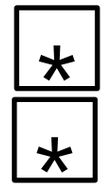
$$= \arg\max_o P\left(I = i | O = o\right)$$

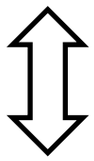
Maximum Likelihood
(the desired o is the one which generates the observed input i with the maximum probability, i.e., which is likely to generate i.)

Image De-convolution (De-blurring)

$$\arg\min_u \int_\Omega \|(A * u)(p) - f(p)\|^2 dp$$

Generative approach to the problem
(the two quantities should be equal up to Gaussian noise)





$$\arg\min_u \|A * u - f\|_2^2$$

norm L2

these two objects has to be close

Image De-convolution (De-blurring)

$$\arg\min_u \int_\Omega \|(A * u)(p) - f(p)\|^2 dp + \int_\Omega \|\nabla u(p)\|^2 dp$$

$$\arg\min_u \|A * u - f\|_2^2 + \|\nabla u\|_2^2$$

Prior on u

Image De-convolution (De-blurring)

$$\arg\min_u \int_\Omega \|(A * u)(p) - f(p)\|^2 dp + \int_\Omega \|\nabla u(p)\| dp$$

$$\arg\min_u \|A * u - f\|_2^2 + \|\nabla u\|_1$$

Prior on u

$*$

L1 Prior + L2 “linear” cost = **Lasso problem**

$$\nabla u(x) = \sum_i \delta(x - p_i) + \eta(x)$$

Sparse gradient

Natural images and Sparsity

$$\nabla u(x) = \sum_i \delta(x - p_i) + \eta(x)$$

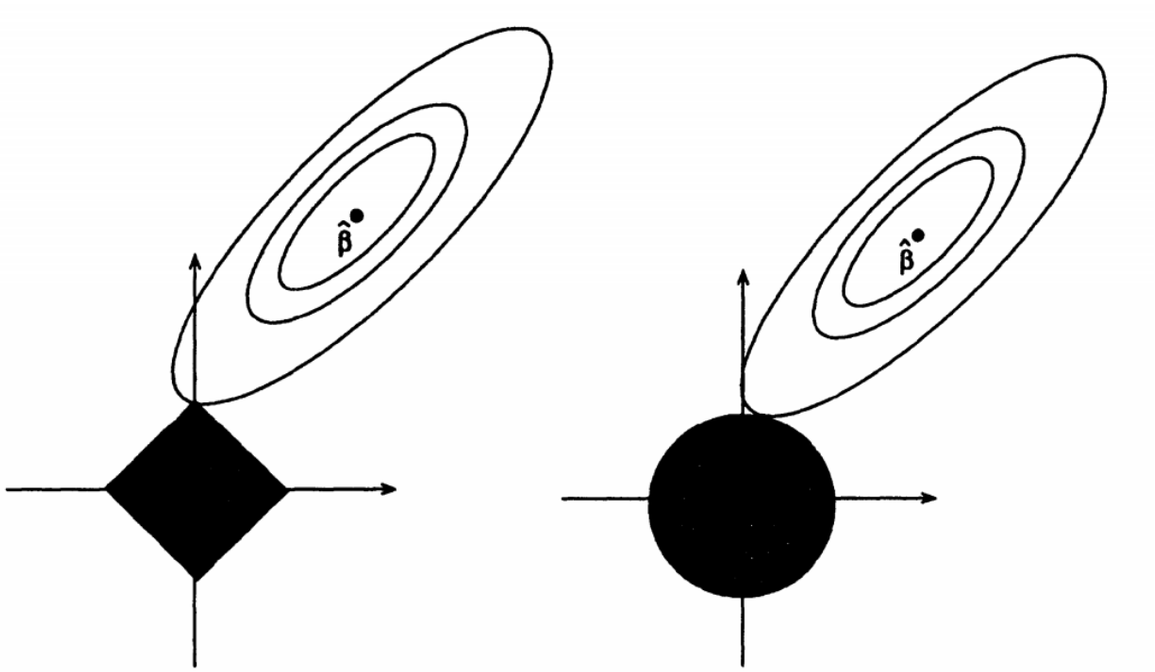
Lasso Problem

$$\arg\min_{x} \quad \|L(x) - y\|_2^2 + \|x\|_1$$

Tibshirani, R., "Regression shrinkage and selection via the lasso". 1996
Journal of the Royal Statistical Society
(useful in Compressed Sensing)

$L(\cdot)$ is linear and orthonormal ⇒ Solution will be sparse

$$\begin{cases} L(x) = Ax \\ AA^T = I \end{cases}$$



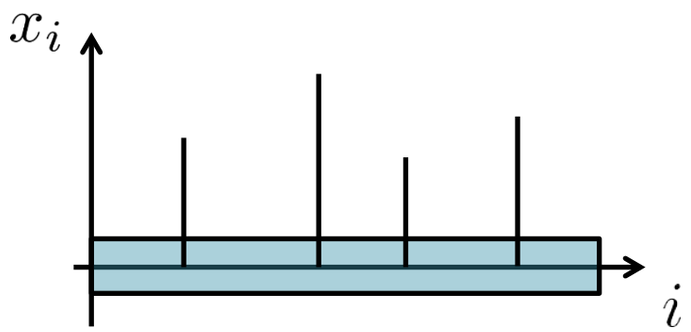
Lasso Problem

$$\arg\min_{x} \quad \|L(x) - y\|_2^2 + \|x\|_1$$

Tibshirani, R., "Regression shrinkage and selection via the lasso". 1996
Journal of the Royal Statistical Society
(useful in Compressed Sensing)

$L(\cdot)$ is linear ⇨ Solution will **typically** be sparse

The further one goes from the linearity and ortho-normality of $L(\cdot)$ ⇨ The more the sparse property will disappear



Infinite Dimensional Case

$$\begin{cases} \arg\min_u \quad \|L(u) - y\|_2^2 + \|\nabla u\|_1 \\ L(\cdot) \quad \text{is linear and orthonormal} \end{cases} \Rightarrow \quad \nabla u \quad \text{will be sparse}$$

Proof:

$$\begin{cases} g = \nabla u \\ I(g)(q) = \int_{\gamma[p,q]} g(r) \cdot dr \end{cases} \Rightarrow \quad I(\nabla u)(q) = u(q) - u(p)$$

$$\int_{\gamma[p,q]} \nabla u(r) \cdot dr = u(q) - u(p) \quad \text{Gradient theorem}$$

Infinite Dimensional Case

$$\begin{cases} \arg\min_u \quad \|L(u) - y\|_2^2 + \|\nabla u\|_1 \\ L(\cdot) \quad \text{is linear and orthonormal} \end{cases} \Rightarrow \quad \nabla u \quad \text{will be sparse}$$

Does it work in infinite dimensional case???

Proof:

$$\begin{cases} g = \nabla u \\ I(g)(q) = \int_{\gamma[p,q]} g(r) \cdot dr \end{cases} \Rightarrow \quad I(\nabla u)(q) = u(q) - u(p) = u(q)$$

Linear operator (orthonormal?????)

select p in such a way $u(p) = 0$ (maybe in a restricted domain of u)

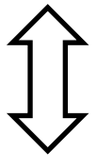
$$= \|L(I(g)) - y\|_2^2 + \|g\|_1$$

Combination of linear operators is linear

C.V.D.

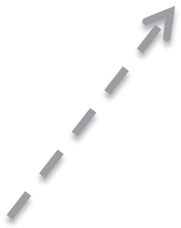
Image De-convolution (De-blurring)

$$\arg\min_u \int_\Omega \|(A * u)(p) - f(p)\|^2 dp + \int_\Omega \|\nabla u(p)\| dp$$



$$\arg\min_u \|A * u - f\|_2^2 + \|\nabla u\|_1$$

Lasso problem



Convolution is a linear operator

$\|\nabla u\|_1$ Total Variation (TV)

$\arg\min_u \|A * u - f\|_2^2 + \|\nabla u\|_1$ TV-L2 = L2 cost + Total Variation

Image De-convolution (De-blurring)

$$\arg\min_u \quad \int_\Omega \|(A * u)(p) - f(p)\|^2 dp + \int_\Omega \|\nabla u(p)\| dp$$

Not derivable in 0 ⇨ it is not C^2

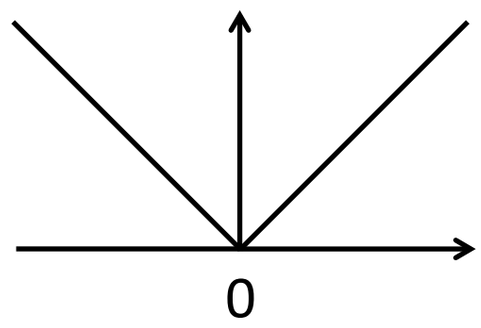
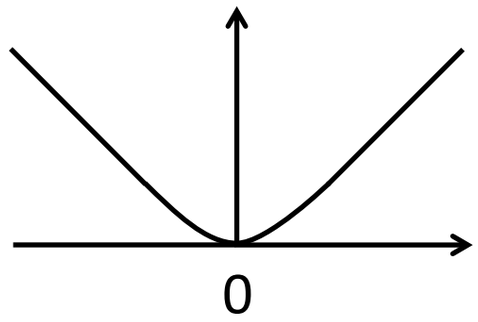


Image De-convolution (De-blurring)

$$\arg\min_u \quad \int_\Omega \|(A * u)(p) - f(p)\|^2 dp + \int_\Omega \|\nabla u(p)\|_\epsilon dp$$

Norm-1 (ϵ): C^2 approximation $\quad \|x\|_\epsilon = \sqrt{\sum x_i^2 + \epsilon}$



$$\frac{\partial}{\partial x}\|x\|_\epsilon = \frac{x}{\|x\|_\epsilon}$$

$$\nabla L\,(u) = 2A * (A * u - f) - div\left(\frac{\nabla u}{\|\nabla u\|_\epsilon}\right)$$

Gradient of our functional

(only if the kernel A is symmetric)

Image De-convolution (De-blurring)

```
Filter = fspecial('gaussian', 30,2);
f=imfilter(I,Filter);

u=f;
a=0.0001;
b=1;
epsilon=0.1;
it=400;

for o=1:it
    fu=imfilter(u,Filter) - f;
    fu=-imfilter(fu,Filter);

    [ux,uy]=gradient(u);
    norm=sqrt(ux.*ux+uy.*uy+epsilon);
    ux=ux./norm;
    uy=uy./norm;

    [uxx]=gradient(ux);
    [uxy,uyy]=gradient(uy);
    DIV=uxx+uyy;

    u = u + b*(fu+a*DIV);
end
```

Image De-noising

u

$f = u + \mu$

$$\arg\min_u \ \|u - f\|_2^2 + \|\nabla u\|_1$$

Generative approach to the problem

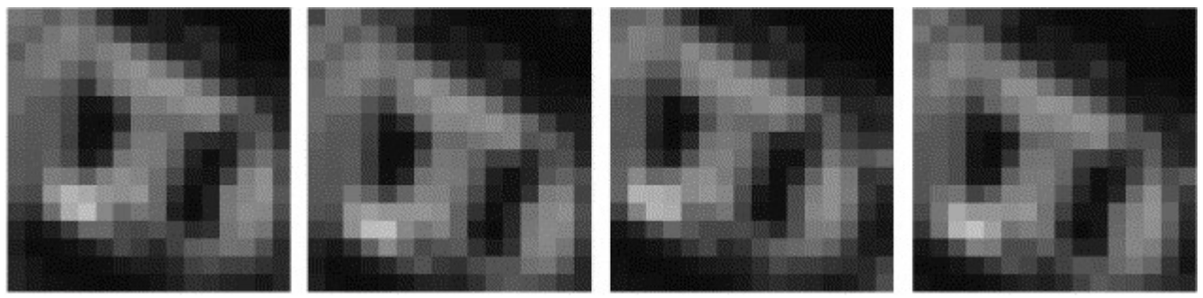
L2 + L1 = Lasso problem

L2 + TV = TV-L2 problem

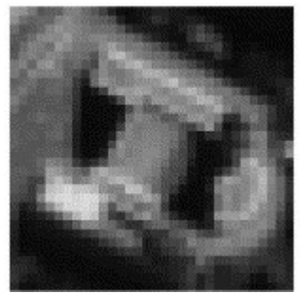
$$\arg\min_u \ \|u - f\|_1 + \|\nabla u\|_1$$

L1 + TV = TV-L1 problem (No lasso)

Video Super-Resolution



sequence of images
(of the same scene with the camera translating a bit)



super resolution result
(aerial image of a building)

- **Why it works?** a sequence of n images provides n observations of a point in the scene: sometimes in the center, sometimes ¼ of a pixel on the right, sometimes on the left, top, down etc…

- We just need to fuse all these information together.

- **Secondary objectives**: - Eliminate sensor noise from the video (thermal noise or spikes (video restoration))
  - Eliminate not wanted occlusions

Video Super-Resolution

- **Video Super-Resolution:** given a sequence of images f_i representing the same image u translated by w_i and down-sampled, find the original image u

$u(x, y)$

Original image (our unknown)

$u_{w_i}(x, y) = u(x + w_i^x, y + w_i^y)$

u translated by $w_i = (w_i^x, w_i^y)$
(sub-pixel accuracy needed)

$A * u_{w_i}$

Down-Sampling
(modeled as convolution with a sinc kernel,
i.e., a low pass filter)

$$\Downarrow$$

$$\sum_{i=1}^{n} \int_{\Omega} \|(A * u_{w_i})(p) - f_i(p)\|^2 dp$$

Generative approach

Video Super-Resolution

- **Video Super-Resolution:** given a sequence of images f_i representing the same image u translated by w_i and down-sampled, find the original image u

$u(x, y)$

Original image (our unknown)

$u_{w_i}(x, y) = u(x + w_i^x, y + w_i^y)$

u translated by $w_i = (w_i^x, w_i^y)$
(sub-pixel accuracy needed)

$A * u_{w_i}$

Down-Sampling
(modeled as convolution with a sinc kernel,
i.e., a low pass filter)

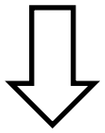
$$\sum_{i=1}^{n} \|A * u_{w_i} - f_i\|_2^2$$

Generative approach

Video Super-Resolution

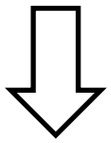
$$\sum_{i=1}^{n} \|A * u_{w_i} - f_i\|_2^2 + \|\nabla u\|_1$$

L1 Prior + L2 cost = Lasso



Solution is piecewise smooth

Sum of L2-Norm is still a 2-type-Norm

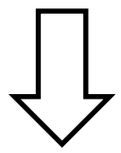


The two quantities should be equal
up to a Gaussian noise

Video Super-Resolution

$$\sum_{i=1}^{n} \|A * u_{w_i} - f_i\|_2^2 + \|\nabla u\|_1$$

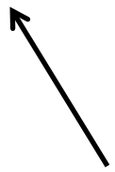
What about an L1 cost, instead?



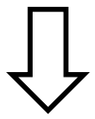
$$\sum_{i=1}^{n} \|A * u_{w_i} - f_i\| + \|\nabla u\|_1$$

L1 Prior + L1 cost = Lasso

(not guarrantee to have piecewise smooth solutions but it behaves similarly due to the median theorem (see later)



Sum of L1-Norm is a 1-type-Norm



The two quantities should be equal up to a Gaussian noise with spikes

More robust to burst noise/spikes/outliers

Why?

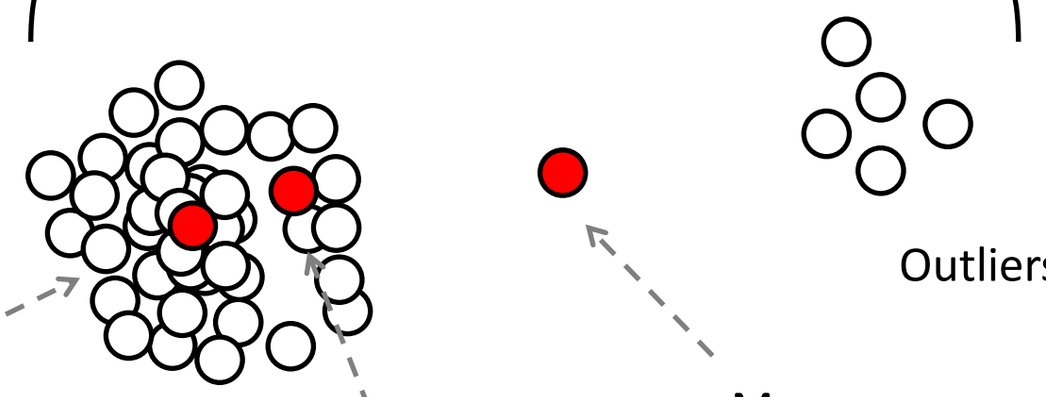
$$\sum_{i=1}^{n} \|A * u_{w_i} - f_i\|_2^2$$

When you have multiple information regarding a single unknown

The result of the minimization depends on the norm

The p-type-Norm Minimizations

Distribution of the observations $\{y_i\}_i$



Outliers

Mean
(influenced by outliers)

Mode
(insensitive to outliers)

Median
(robust by outliers)

$$\arg\min_x \sum_i \|x - y_i\|_2$$ Mean

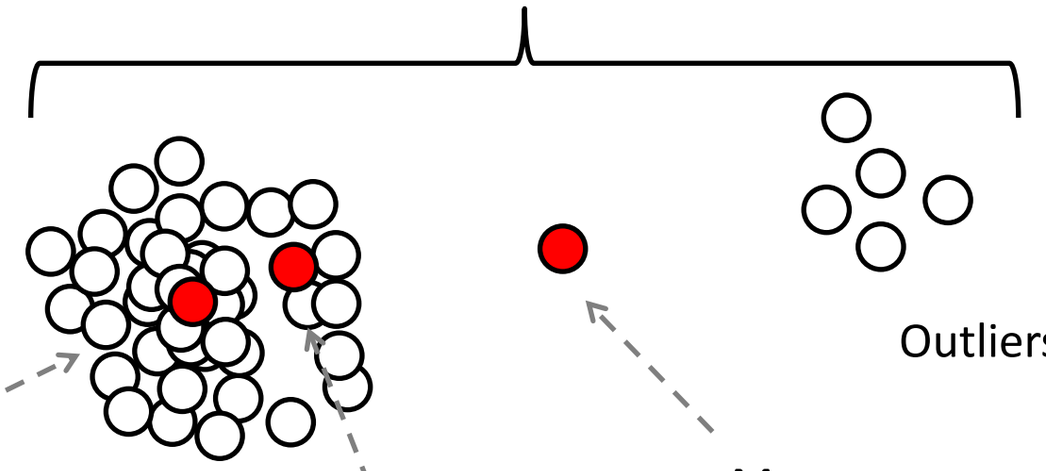
$$\|x - y\|_\epsilon$$

$$\arg\min_x \sum_i \|x - y_i\|_1$$ Median ← Not differentiable, difficult to optimize

$$\arg\min_x \sum_i \|x - y_i\|_0$$ Mode

The p-type-Norm Minimizations

Distribution of the observations $\{y_i\}_i$



Outliers

Mean
(influenced by outliers)

Mode
(insensitive to outliers)

Median
(robust by outliers)

$$\arg\min_x \sum_i \|x - y_i\|_2 \quad \text{Mean}$$

$$\arg\min_x \sum_i \|x - y_i\|_1 \quad \text{Median}$$

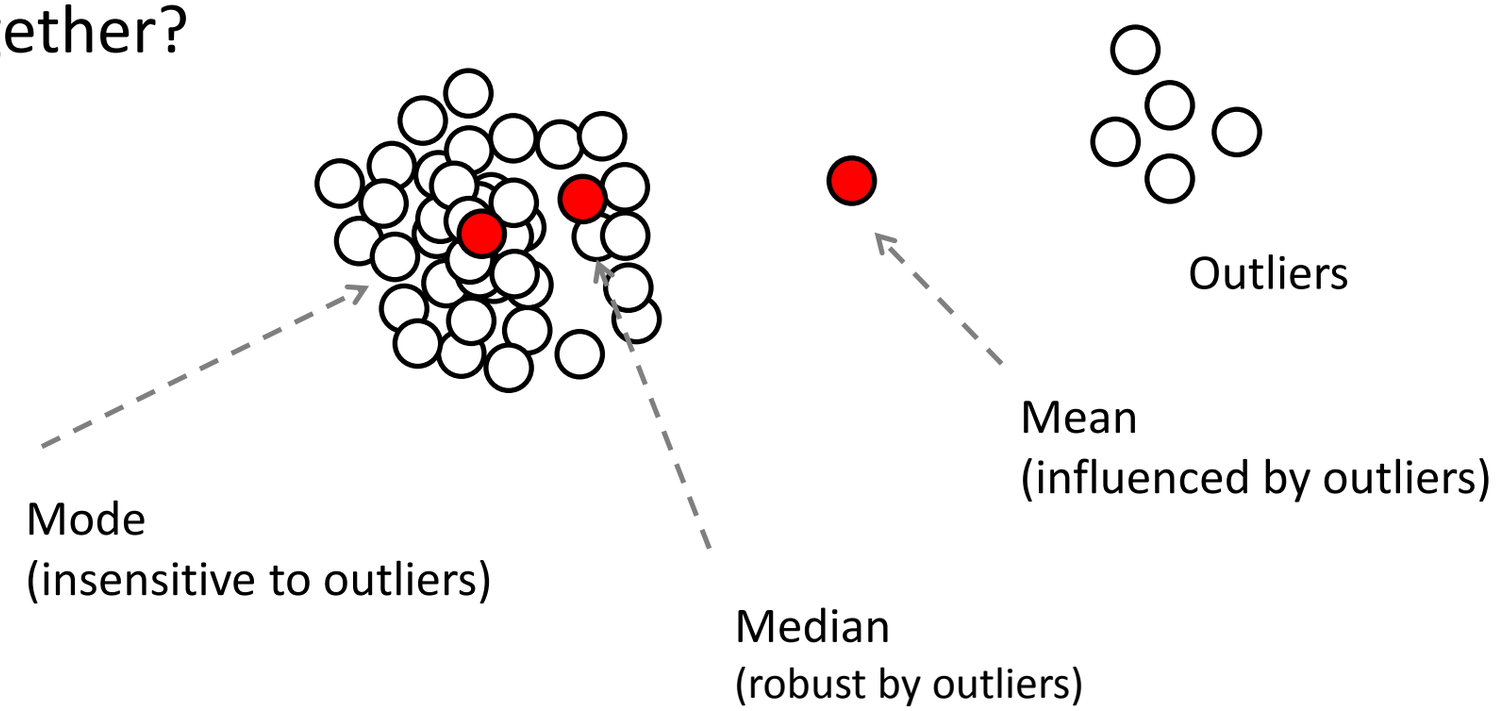
$$\arg\min_x \sum_i \|x - y_i\|_0 \quad \text{Mode}$$

$$\|x - y\|_{0.001,\epsilon}$$

Not a norm, difficult to optimize

Conclusions: in general…

- if one has multiple information regarding an unknown, how does he fuse them together?



- so.. the best way is to sum them together and minimize a functional

$$\arg\min_{x} \sum_{i} \|x - y_i\|_p$$

Does the used norm matter?

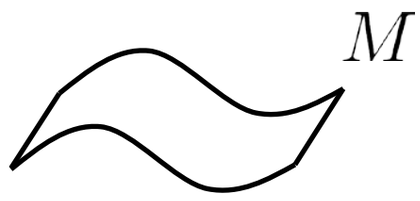
Mathematical Foundations of
Computer Graphics and Vision

Metrics on SO(3) and Inverse Kinematics

Luca Ballan

Optimization on Manifolds

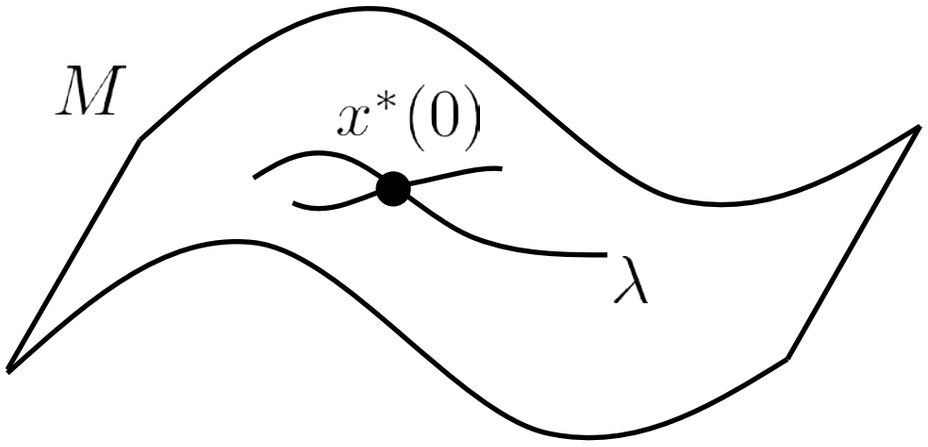
$$\begin{cases} \min f(x) \\ x \in M \end{cases}$$



$$\begin{cases} x^*(0) = x_0 \\ \dfrac{\partial x^*}{\partial t}(t) = -\beta d \end{cases}$$

- Descent approach
- d is a ascent direction

Optimization on Manifolds

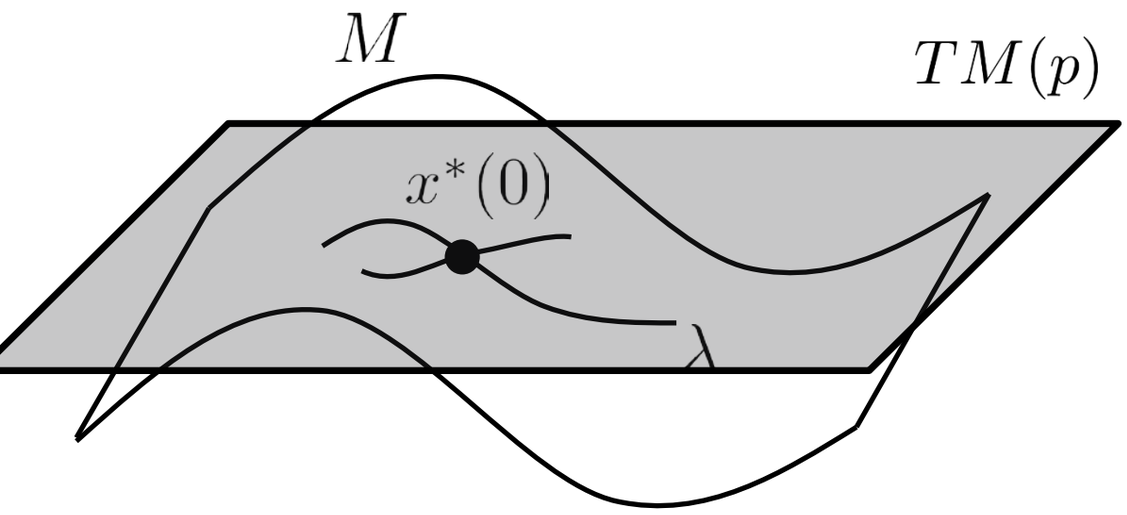


- Given the current point $x^*(0)$
- Compute the directional derivative for each direction λ, i.e. for each curve

$$\frac{\partial L}{\partial \lambda}(x^*(0)) \in \mathbb{R}$$

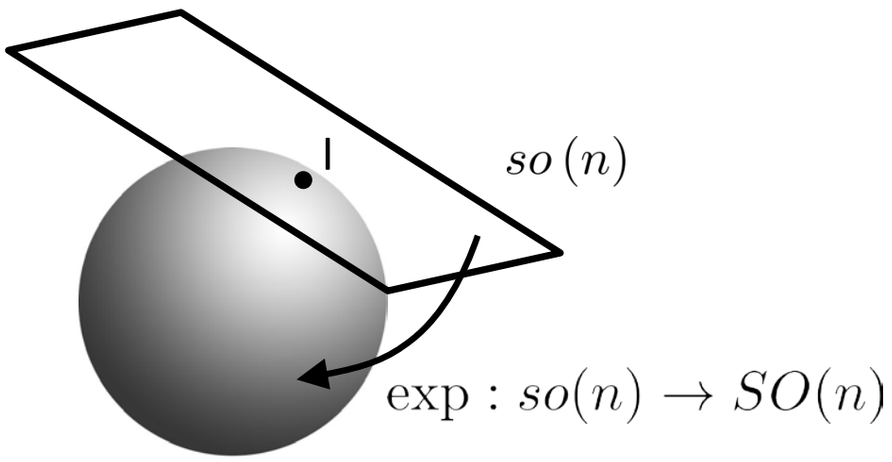
- Determine the λ for which $\frac{\partial L}{\partial \lambda}(x^*(0))$ is maximum
- Move along this λ

Optimization on Manifolds



- Given the current point $x^*(0)$
- Compute the directional derivative for each direction λ, i.e. for each curve

$$\frac{\partial L}{\partial \lambda}(x^*(0)) \in \mathbb{R}$$

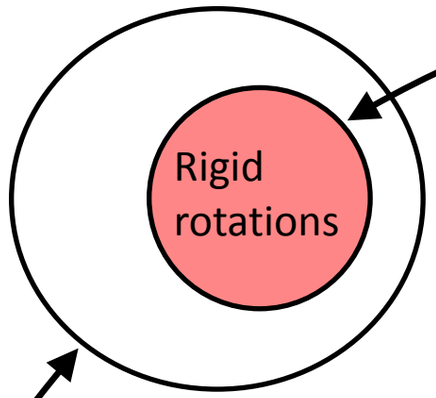


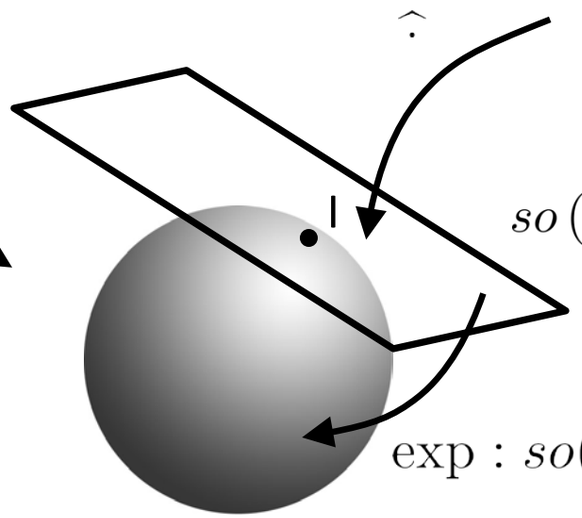
- Determine the λ for which $\frac{\partial L}{\partial \lambda}(x^*(0))$ is maximum
- Move along this λ

Last Lecture

Rigid transformations
(maps which preserve distances and space orientation)

isomorphic





$\mathbb{R}^n$ ▪ Angle/axis representation

$so\,(n)$ = ▪ Tangent space at the identity
▪ Lie algebra of the nxn skew-symmetric matrices

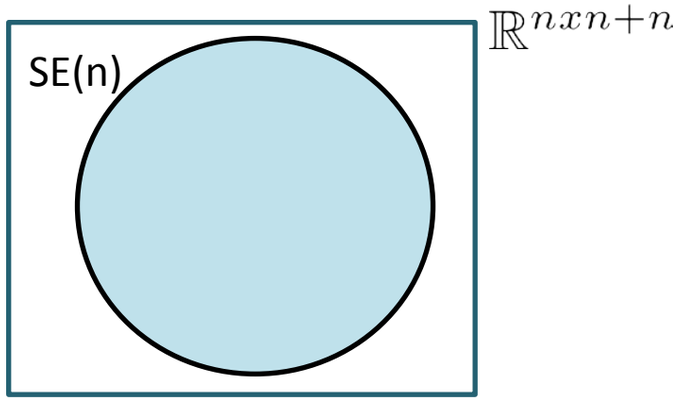
$\exp : so(n) \rightarrow SO(n)$ ▪ smooth map
▪ coincides with the matrix exponential

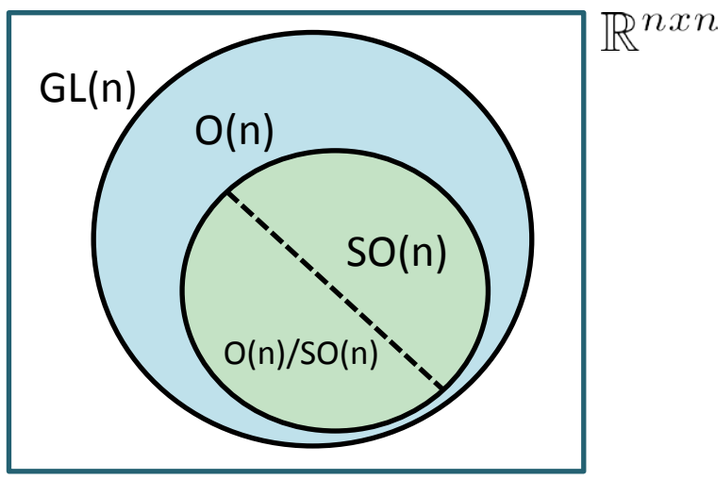
$$\exp(M) = \sum_{k=0}^{\infty} \tfrac{1}{k!} M^k$$

$SO\,(n)$ = ▪ k – manifold of the rotation matrices immerse in $\mathbb{R}^{n \times n}$
▪ Lie group

isomorphic

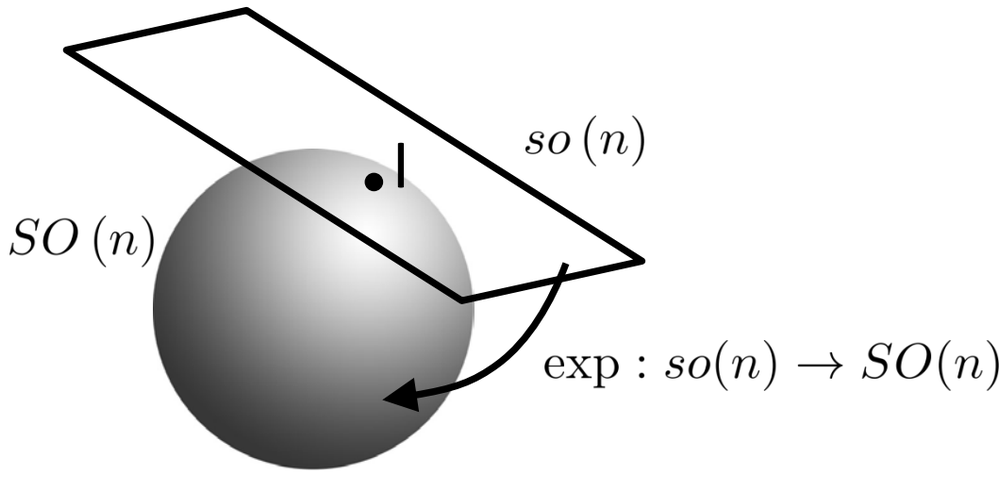
$SE\,(n)$ = ▪ k+n – Manifold defined as $SO\,(n) \times \mathbb{R}^n$
▪ Lie group





Exponential Map

- The exponential map is a function proper of a Lie Group



- For matrix groups

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

- For SO(3), Rodrigues' rotation formula:

$$\exp(\widehat{a}) = I + \frac{\sin(\|a\|)}{\|a\|}\widehat{a} + \frac{(1-\cos(\|a\|))}{\|a\|^2}\widehat{a}^2$$

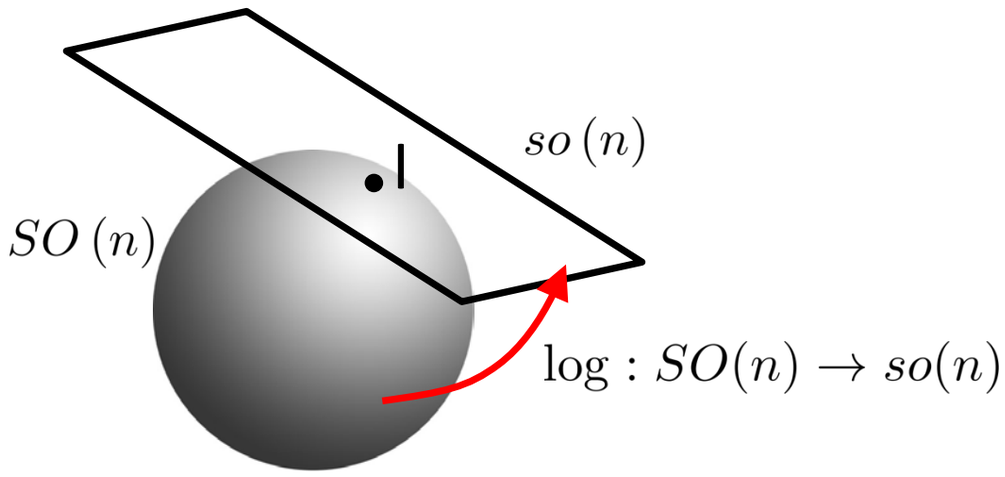
- Smooth
- Surjective
- not Injective

- not Linear $e^{X+Y} \neq e^X e^Y$ (not an isomorphism)

- $\partial e^X = \partial X e^X = e^X \partial X$

Logarithm Map

- Since $\exp(\cdot)$ is surjective… it exists at least an inverse



- The inverse of $\exp(\cdot)$ is

$$\log(X) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}(X - I)^k$$

- For SO(3), Rodrigues' rotation formula:

$$\log(X) = \frac{1}{2\sin(\theta)}\left(X - X^T\right) \qquad R \neq I$$

$$\theta = arccos\left(\frac{trace(X)-1}{2}\right)$$

Properties

$$\log(I) = 0 = \widehat{0}$$

Identity

$$\log(X^{-1}) = -\log(X)$$

Inverse

$$\log(XY) \neq \log(X) + \log(Y)$$

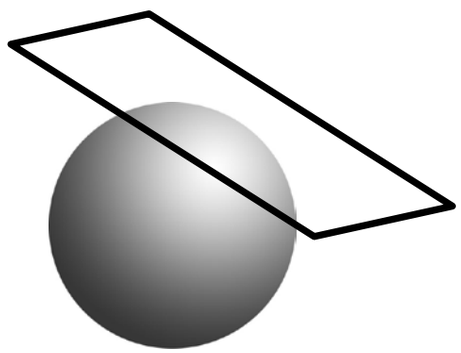
in general not “Linear” **(different from the standard log in $\mathbb{R}$)**

$$e^X e^Y \neq e^Y e^X$$

$$e^{\log(X)} = X$$

$$\log(e^A) = \ ?$$

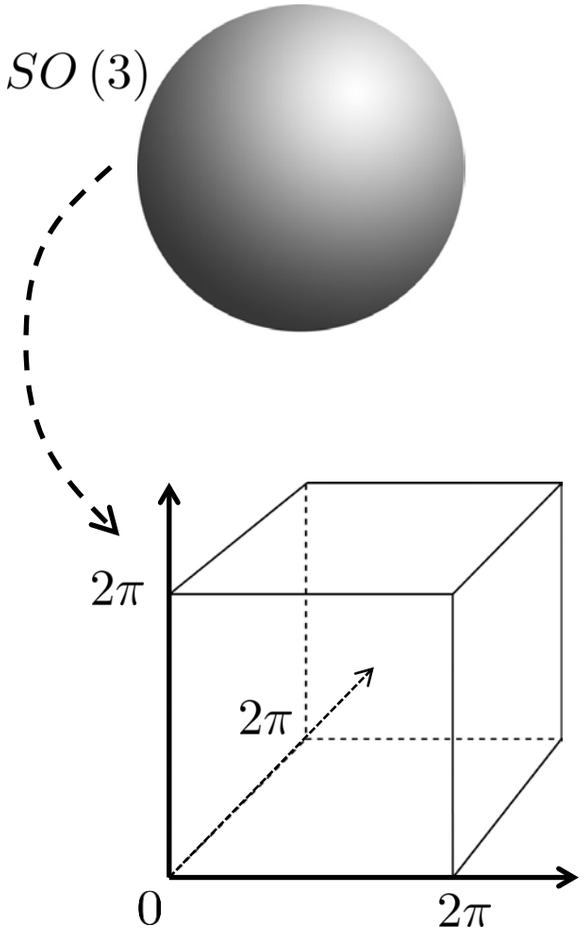
$$\partial \log(X) = X^{-1} \partial X$$



Derivative

Last Lecture

- There exists a famous “local chart” for $SO(3)$



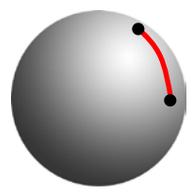
- Euler-Angle representation (cube)

- Intuitive (easy to visualize)
- Easy to set constraints

- any rotation matrix in SO(3) can be describe as a non-unique combination of 3 rotations
(e.g. one along the x-axis, one on the y-axis, and one on the z-axis)

  - Although it is widely used, this representation has some problems

- Topology is not conserved $(0, 2\pi)$
- Metric is distorted



- Derivative is complex
(although people use it)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

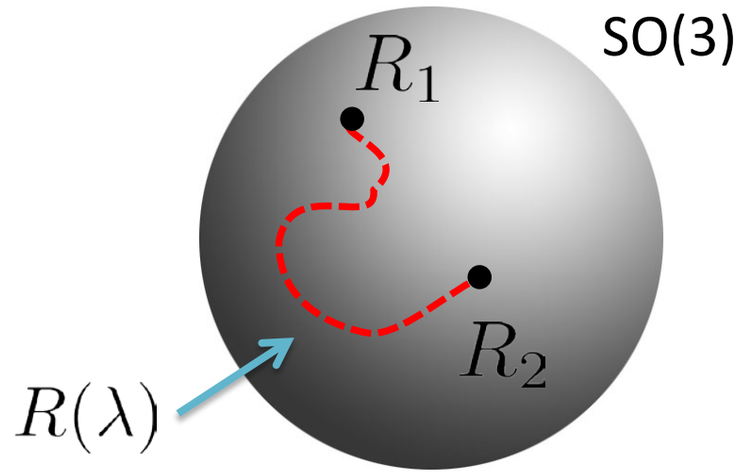
- Gimbal Lock

Content

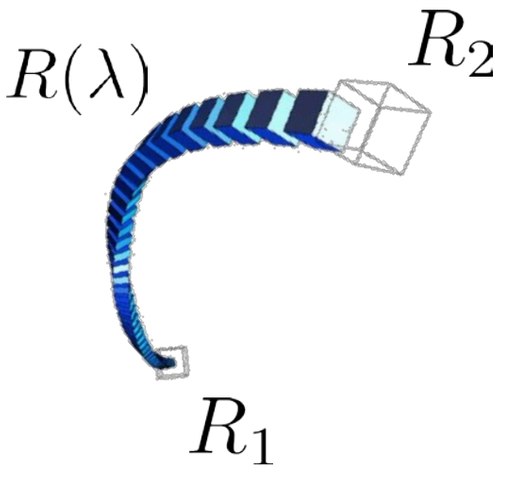
- **Interpolation in SO(3)**
- Metric in SO(3)
- Kinematic chains

Interpolation in SO(3)

- Given two rotation matrices $R_1, R_2 \in SO(3)$, one would like to find a **smooth path** in SO(3) connecting these two matrices.



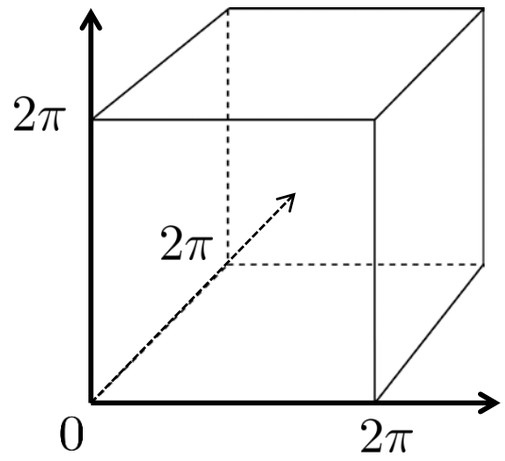
$$
\begin{cases}
R(\lambda) \in SO(3) \quad \lambda \in [0,1] \\
R(\lambda) \quad \text{smooth} \\
R(0) = R_1 \\
R(1) = R_2
\end{cases}
$$



Interpolation in SO(3)

Approach 1: Linearly interpolate R1 and R2 in one of their representation

- **Euler angles**:



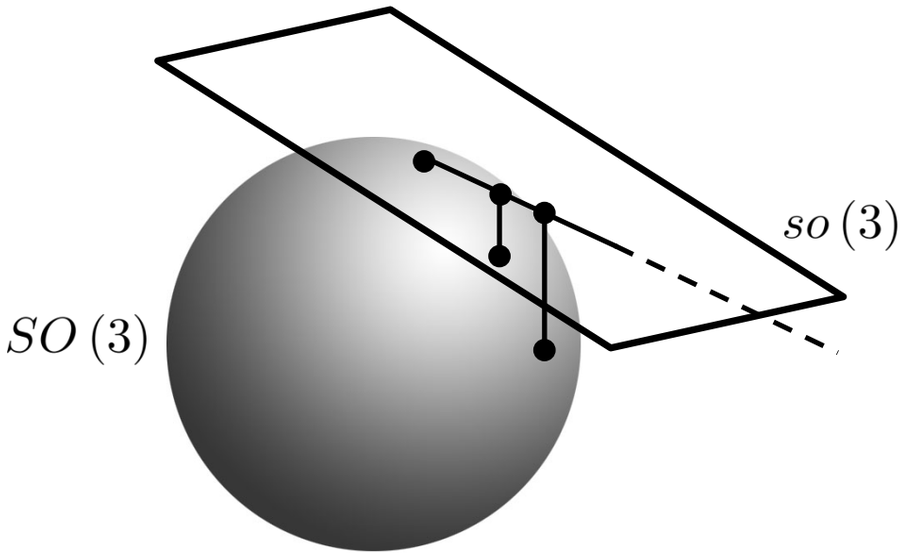
- R1, R2 too far -> not intuitive motion
- Topology is not conserved

Interpolation in SO(3)

Approach 1: Linearly interpolate R1 and R2 in one of their representation

- **Angle-Axis**:

$$\omega(\lambda) = (\lambda\omega_1 + (1-\lambda)\omega_2)$$



- Interpolate on a plane and then project on a sphere

- The movement is not linear with a constant speed. It gets faster the more away it is from the Identity

Interpolation in SO(3)

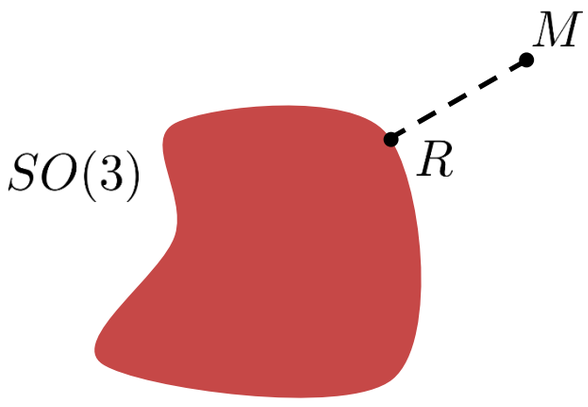
Approach 2: Linearly interpolate R1 and R2 as matrices

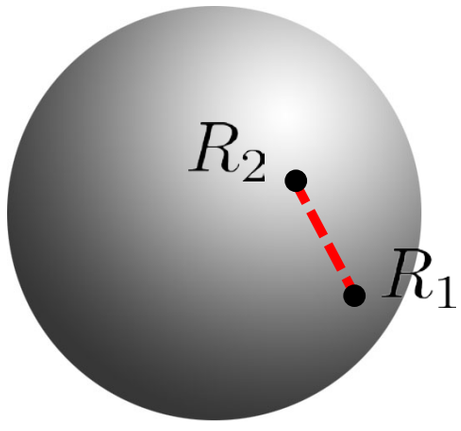
$$R(\lambda) = \underbrace{(\lambda R_1 + (1-\lambda)R_2)}_{\notin SO(3)}$$

- it needs to be projected back on the sphere

$$\pi_{SO(3)}(M) = \underset{R \in SO(3)}{\arg\min} \|M - R\|_F^2$$

Not an element of SO(3) because it is a multiplicative group not an additive one



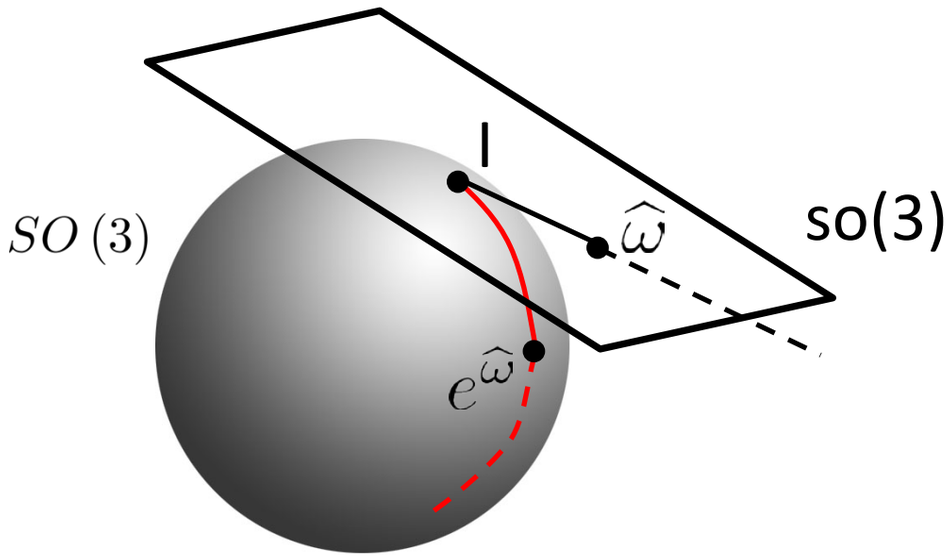


- if R1, R2 are far away from each other, the speed is not linear at all

Interpolation in SO(3)

Approach 3: use the geodesics of SO(3)

- Lie Groups: a line passing through 0 in the Lie algebra **maps** to a geodesic of the Lie group through the identity



⇨ consequently the curve

$$R(\lambda) = e^{\lambda\widehat{\omega}}$$

is a geodesic of SO(3) passing through I

This holds only for any line passing through 0 and consequently for any geodesic passing through the identity

Interpolation in SO(3)

- To find the geodesic passing through R_1 and R_2 we need to rotate the ball SO(3) by R_1^{-1}

$$R_1^{-1}R_1 \longrightarrow R_1^{-1}R_2$$

$$I \longrightarrow e^{\log(R_1^{-1}R_2)}$$

$$I \longrightarrow e^{\lambda \log(R_1^{-1}R_2)} \quad \text{geodesic between I and } R_1^{-1}R_2$$

SLERP
(spherical linear interpolation)

$$R_1 \longrightarrow R_1 e^{\lambda \log(R_1^{-1}R_2)} \quad \text{geodesic between } R_1 \text{ and } R_2$$

- The resulting motion is very intuitive and it is performed at uniform angular speed in $SO(3)$

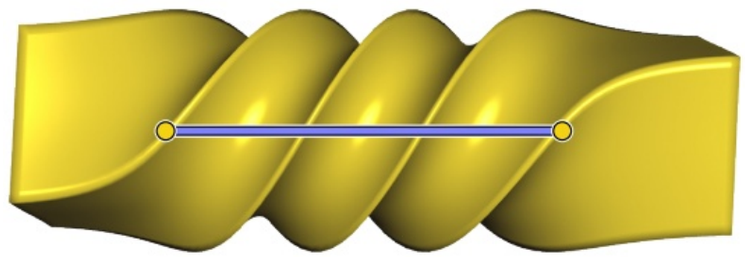
Interpolation in SO(3)

- On a vector space with Euclidean metric, the geodesic connecting R_1 and R_2 would have corresponded to the straight line

$$R(\lambda) = R_1 + \lambda(R_2 - R_1)$$

Questions?

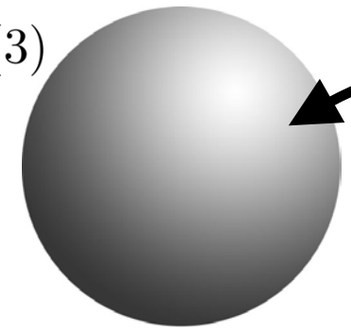
- Given two rotations R1 and R2, interpolate along the geodesic starting from R1 passing n times through R2 and R1 and ending in R2.



something like this but not limited to a single axis

from [jacobson 2011]

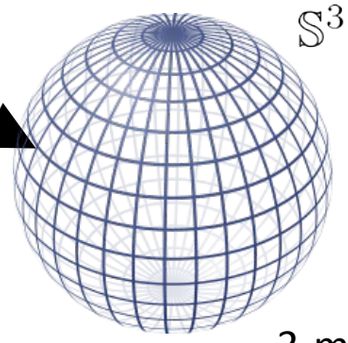
A word about quaternions…



$SO(3)$

isomorphic

isometry



$\mathbb{S}^3/\{-I, I\}$ = the hypersphere in $\mathbb{R}^4$ quotient the antipodal points.
(the hemisphere)

3-manifold

3-manifold

- quaternion multiplication

$(\mathbb{S}^3/\{-I, I\}, \cdot) =$ **Group of the unit quaternions**

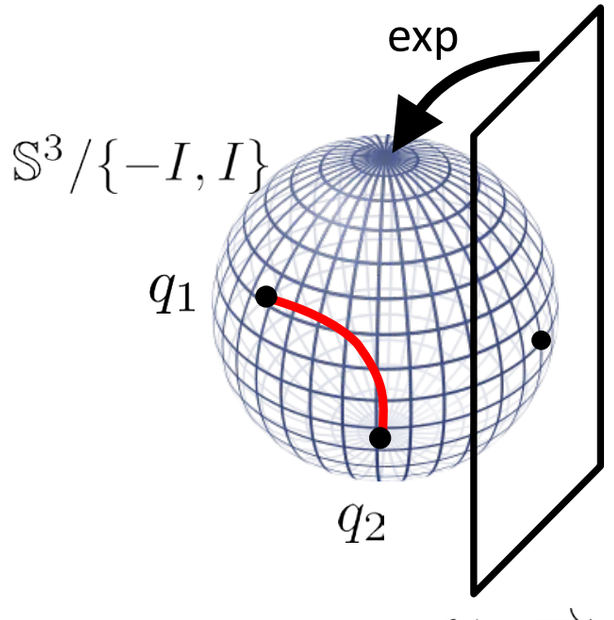
- 3-manifold
- Lie group

Tangent space at the identity

- Lie Algebra
- Imaginary numbers

Exponential map

- quaternion exponential



$\mathrm{Im} = \{(0, \vec{v}) \mid \forall \vec{v} \in \mathbb{R}^3\}$

Geodesics

$$q_1 \longrightarrow q_1 e^{\lambda \log(q_1^{-1} q_2)}$$

$$q_1 \longrightarrow q_1 (q_1^{-1} q_2)^{\lambda}$$

- they are the same as the geodesics in $SO(3)$

- PRO: easy to compute SLERP
- CON: difficult to perform derivatives in this space $q \cdot s \cdot q^{-1}$

Content

- Interpolation in SO(3)
- **Metric in SO(3)**
- Kinematic chains

Metric in SO(3)

- We talk about geodesics, but what was the used metric?
  - a metric tells how close two rotations are
  - it is necessary to evaluate an estimator w.r.t. a ground truth

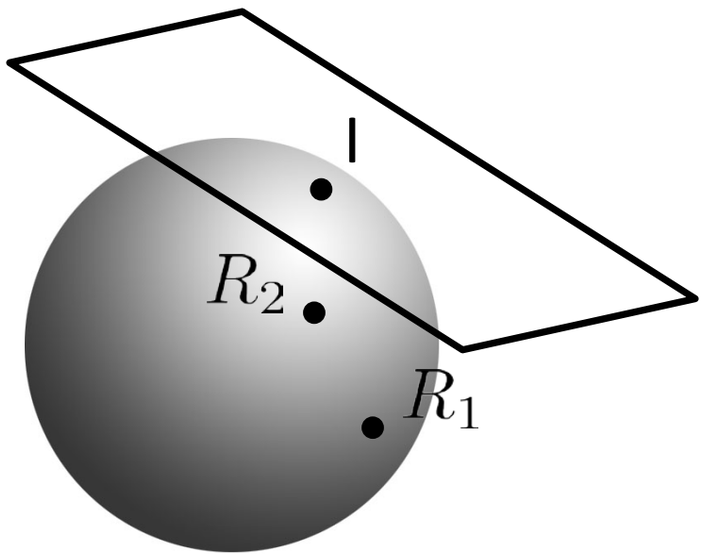
Metric in SO(3)

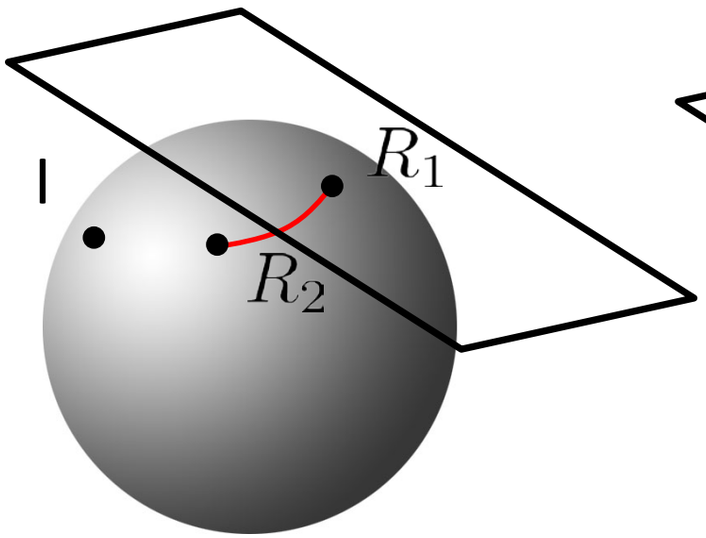
- We talk about geodesics, but what was the used metric?

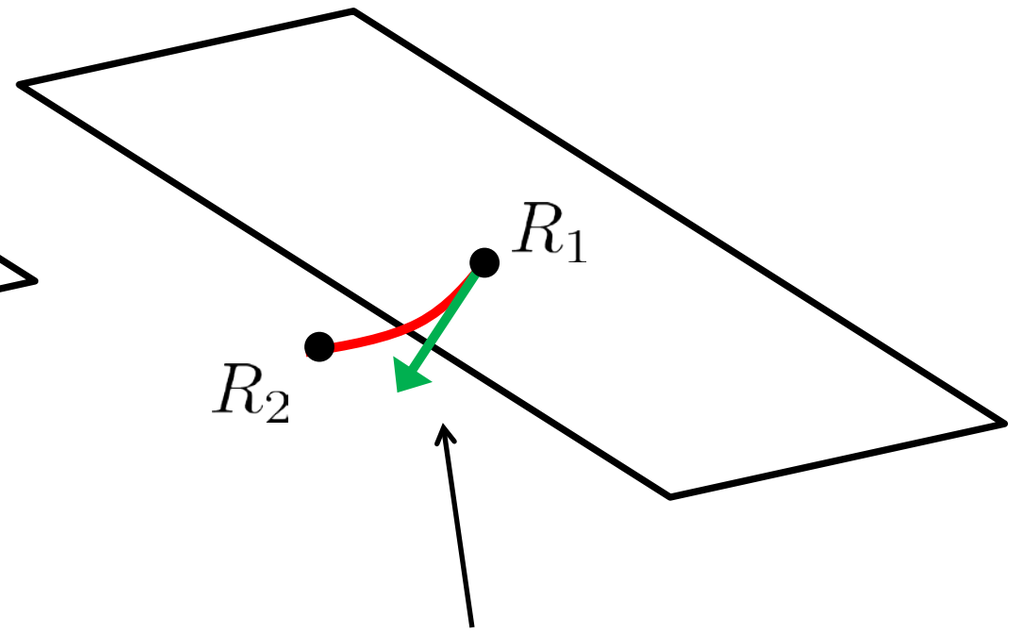
$$d_R(R_1, R_2) = \frac{1}{\sqrt{2}} \| \log(R_1^{-1} R_2) \|_F$$

Riemannian/Geodesic/Angle metric
(= to the length of the geodesic
connecting R1 and R2)

*







*

Metric in SO(3)

- We talk about geodesics, but what was the used metric?

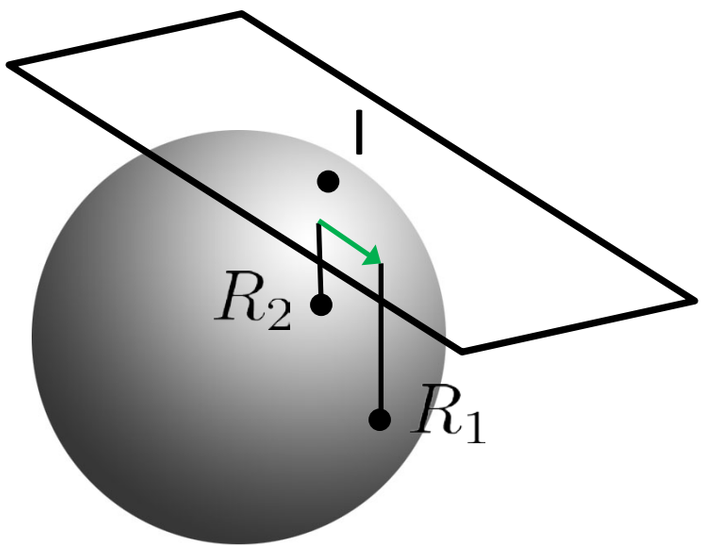
$$d_R(R_1, R_2) = \frac{1}{\sqrt{2}} \| \log(R_1^{-1} R_2) \|_F$$

Riemannian/Geodesic/Angle metric
(= to the length of the geodesic connecting R1 and R2)

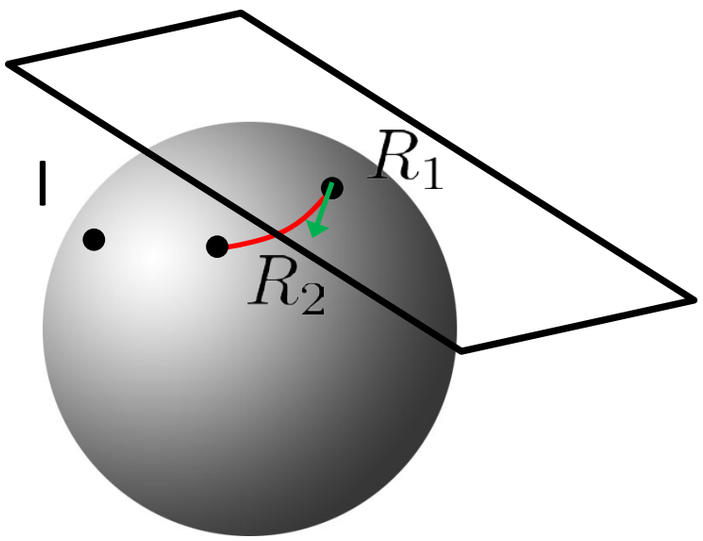
$$d_H(R_1, R_2) = \| \log(R2) - \log(R1) \|_F$$

Hyperbolic metric

- similar to the Riemannian if R1=I



Hyperbolic metric



Riemannian metric

Metric in SO(3)

- We talk about geodesics, but what was the used metric?

$$d_R(R_1, R_2) = \frac{1}{\sqrt{2}} \| \log(R_1^{-1} R_2) \|_F$$

Riemannian/Geodesic/Angle metric
(= to the length of the geodesic connecting R1 and R2)

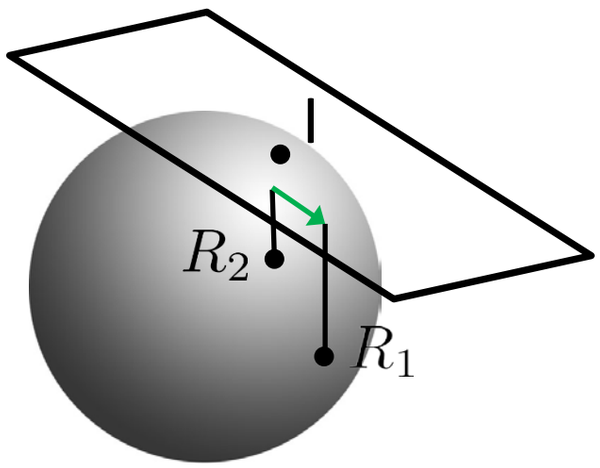
$$d_H(R_1, R_2) = \| \log(R2) - \log(R1) \|_F$$

Hyperbolic metric

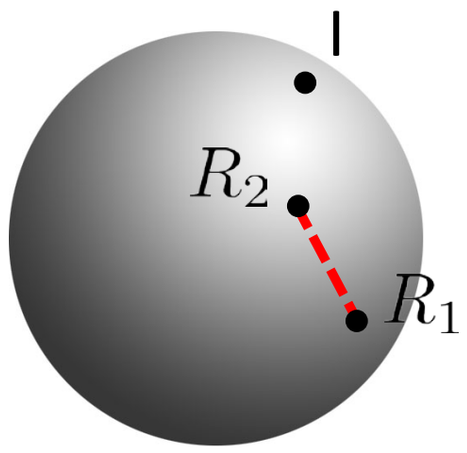
$$d_F(R_1, R_2) = \| R1 - R2 \|_F$$

Frobenius/Chordal metric

- not similar to Hyperbolic
- similar to the Riemannian if R1 and R2 are close to each other



Hyperbolic metric



Frobenius metric

Metric in SO(3)

- We talk about geodesics, but what was the used metric?

$$d_R(R_1, R_2) = \frac{1}{\sqrt{2}} \| \log(R_1^{-1} R_2) \|_F$$

Riemannian/Geodesic/Angle metric
(= to the length of the geodesic connecting R1 and R2)

$$d_H(R_1, R_2) = \| \log(R2) - \log(R1) \|_F$$

Hyperbolic metric

$$d_F(R_1, R_2) = \| R1 - R2 \|_F$$

Frobenius/Chordal metric

$$d_{\mathbb{S}^3}(q_1, q_2) = \| q_1 - q_2 \|_2$$

Quaternion metric
(related to the space of quaternions, not specifically to the sphere of unit quaternions)

- Similar to the Hyperbolic one

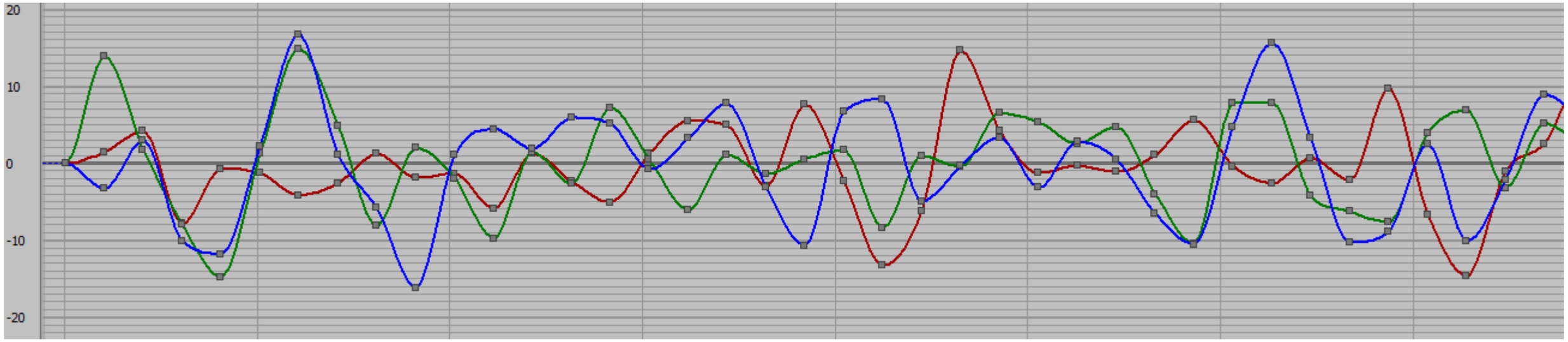
Filtering in SO(3)

- Given n different estimation for the rotation of an object

$$R_1, \ldots, R_n$$

- how can I get a better estimate of R ?

Object at unknown rotation R



Filtering in SO(3)

- Given n different estimation for the rotation of an object

$$R_1, \ldots, R_n$$

- how can I get a better estimate of R ?

Object at unknown rotation R

- **Solution:** which of these is the best?
  - Average the rotation matrices R_i ?
  - Average the Euler angles of each R_i?
  - Average the angle-axes of each R_i ?
  - Average the quaternions related to each R_i?

$$\frac{1}{n}\sum_{i=1}^{n} R_i$$ (not rotation)

$$\left(\frac{1}{n}\sum_{i=1}^{n}\alpha_i, \frac{1}{n}\sum_{i=1}^{n}\beta_i, \frac{1}{n}\sum_{i=1}^{n}\gamma_i\right)$$

$$\frac{1}{n}\sum_{i=1}^{n}\omega_i$$

$$\frac{1}{n}\sum_{i=1}^{n} q_i$$

(rotation matrices)

- Why average?

Filtering in SO(3)

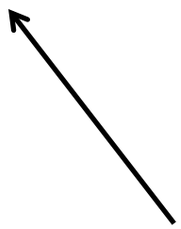
- Why average?
  - By saying average, I'm implicitly assuming that the error in the measurements is Gaussian with zero mean

$$\underset{x}{\operatorname{argmin}} \sum_{i=1}^{n} \|x - x_i\|^2 \qquad \text{Average} \qquad = \frac{1}{n} \sum_{i=1}^{n} x_i$$

ℓ_2

$$\underset{x}{\operatorname{argmin}} \sum_{i=1}^{n} \|x - x_i\| \qquad \text{Median} \qquad = \operatorname{sort}(\{x_i\})[n/2]$$

ℓ_1



This can be generalized using metrics instead of norms

Filtering in SO(3)

- Why average?
  - By saying average, I'm implicitly assuming that the error in the measurements is Gaussian with zero mean

$$\operatorname*{argmin}_x \sum_{i=1}^{n} d(x, x_i)^2$$

$$\operatorname*{argmin}_x \sum_{i=1}^{n} d(x, x_i)$$

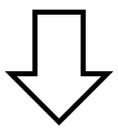
in case of SO(3),
which metric do we use here?

Average ℓ_2	$= \frac{1}{n} \sum_{i=1}^{n} x_i$
Median ℓ_1	$= \operatorname{sort}(\{x_i\})[n/2]$

This formulas can be applicable only to $\mathbb{R}$ neither to $\mathbb{R}^n$

Filtering in SO(3)

$$d_H(R_1, R_2) = \| \log(R2) - \log(R1) \|_F$$



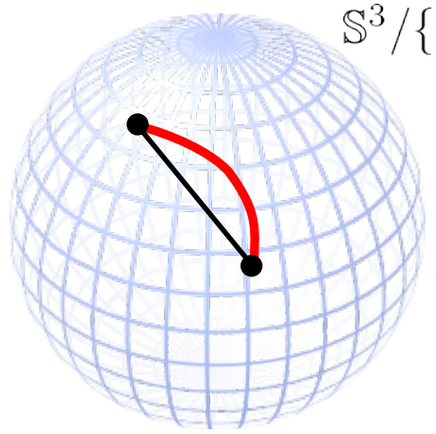
$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_H(R, R_i)^2$$

Geometric mean

- $= \frac{1}{n} \sum_{i=1}^{n} \log(R_i) = \frac{1}{n} \sum_{i=1}^{n} \omega_i$ Average of the angle-axes of each R_i

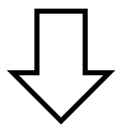
*

- Similar to the **projection** of $\frac{1}{n} \sum_{i=1}^{n} q_i$



Filtering in SO(3)

$$d_F(R_1, R_2) = \|R1 - R2\|_F$$

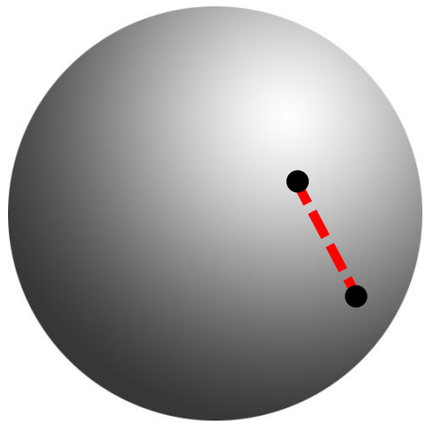


$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_F(R, R_i)^2$$

Matrix mean

- Similar to the **projection** of $\frac{1}{n}\sum_{i=1}^{n} R_i$

Average of the each matrix element



Filtering in SO(3)

$$d_R(R_1, R_2) = \frac{1}{\sqrt{2}} \| \log(R_1^{-1} R_2) \|_F$$

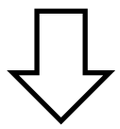
$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_R(R, R_i)^2$$

Fréchet/Karcker mean

- No close form solution
- Solve a minimization problem
- when the solution R is close to I ⇨ = Geometric mean
- when the R_i are all close together ⇨ = Matrix mean

Filtering in SO(3)

$$d_R(R_1, R_2) = \frac{1}{\sqrt{2}} \| \log(R_1^{-1} R_2) \|_F$$



$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_R(R, R_i)^2$$

Fréchet/Karcker mean

- Why is so different?
  - we need to find the rotation R such that the squared sum of the lengths of all the geodesics connecting R to each R_i is minimized
  - The geodesics should start from R and not from the identity (like in the geometric mean)
  - we need to find the tangent space such that the squared sum of the lengths of all the geodesics of each R_i is minimized

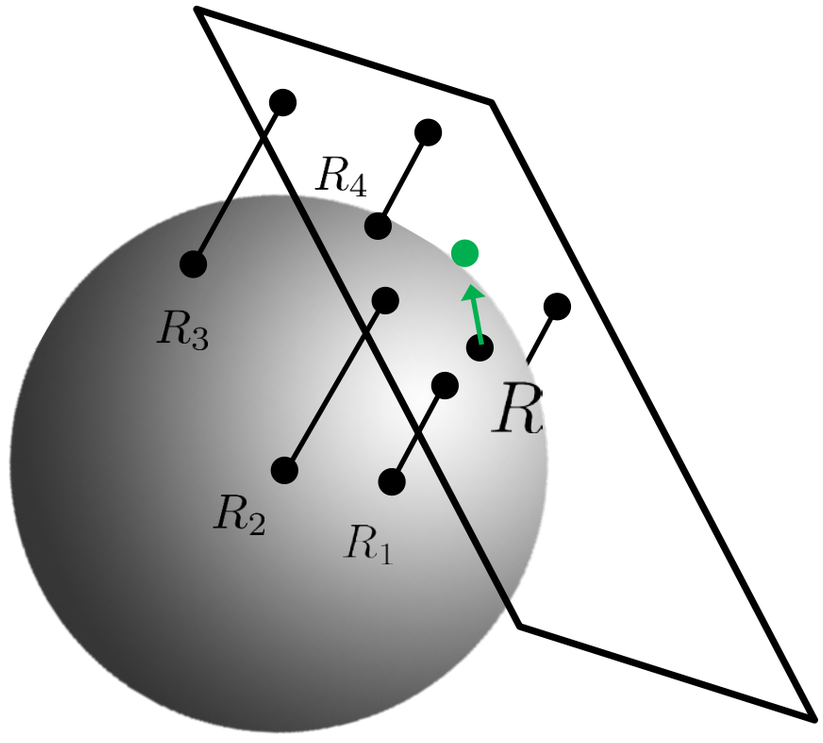
*

Fréchet mean

$$\underset{R \in SO(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_R(R, R_i)^2$$

- Gradient descent on the manifold
- J. H. Manton, A globally convergent numerical algorithm for computing the centrer of mass on compact Lie groups, ICARCV 2004

- Set $\bar{R} = R$ Matrix or Geometric mean
- Compute the average on the tangent space of R $r = \sum_{i=1}^{n} \log(R^{-1} R_i)$
- Move towards r $R = Re^r$



Content

- Interpolation in SO(3)
- Metric in SO(3)
- **Kinematic chains**

Special Euclidean group SE(3)

$$SE(3) = (SO(3) \times \mathbb{R}^3, \times)$$

Special Euclidean group of order 3

- for simplicity of notation, from now on, we will use homogenous coordinates

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in SE(3)$$

- A way of parameterize SE(3) is the following

$$\xi = (\omega, t) \rightarrow \begin{bmatrix} e^{\widehat{\omega}} & t \\ 0 & 1 \end{bmatrix} = e^{\widehat{\xi}}$$

This is not the real exponential map in SE(3)
(but it is more intuitive)

Translation $t \in \mathbb{R}^3$

Angle/axis representation of the rotation $\omega \in so(3)$

- (ω, t) is called **twist**, and usually indicated with the symbol ξ

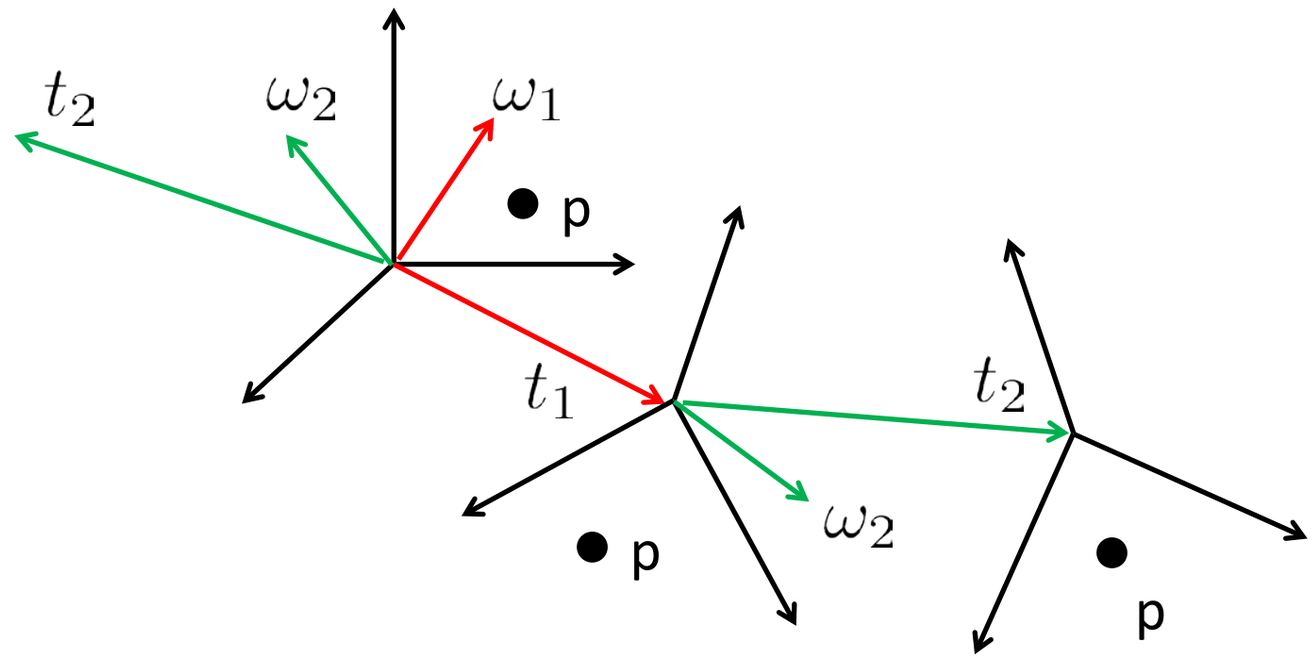
Composition of Rigid Motions

$$e^{\widehat{\xi_1}}\ e^{\widehat{\xi_2}}\ p$$

Transform p

Transform the transformation of p

$$\xi_1 = (\omega_1, t_1)$$
$$\xi_2 = (\omega_2, t_2)$$



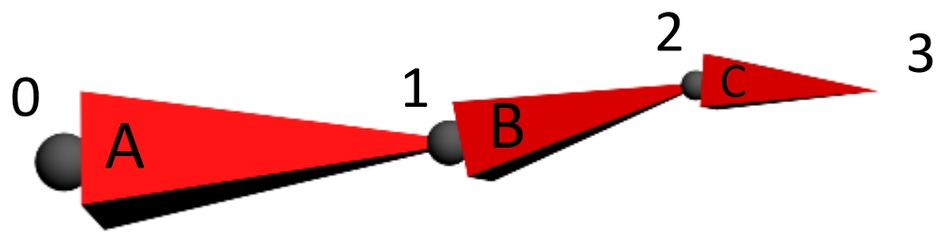
ξ_2 is expressed in local coordinates relative to the framework induced by ξ_1

The second transformation is actually performed on the twist

$$e^{\widehat{\xi_1}}\,\xi_2$$

Kinematic Chain

- A kinematic chain is an ordered set of rigid transformations



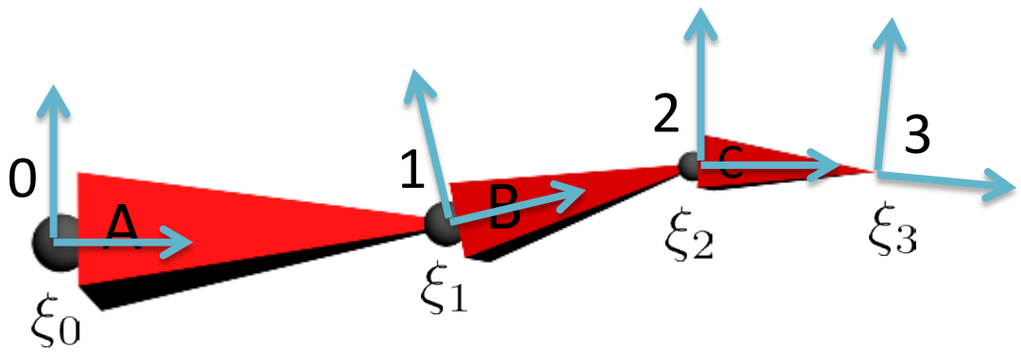
- Each  is called **bone** (A,B,C)

- Each  is called **joint** (0,1,2,3)

- joint 0 is called **base/root** (and assumed to be fixed)
- joint 3 is called **end effector**

Kinematic Chain

- A kinematic chain is an ordered set of rigid transformations



- Each bone has its own coordinate system determining its position in the space and the orientation of its local axes

- **the bones A, B, C are oriented accordingly to the x-axis of the reference system**

- The base of each bone corresponds to a joint

- Each reference system is an element of SE(3) determined by a **twists** $(\xi_0, \xi_1, \xi_2, \xi_3)$

- the twists ξ_0, ξ_1, ξ_2, and ξ_3 all together determine completely the configuration of the kinematic chain

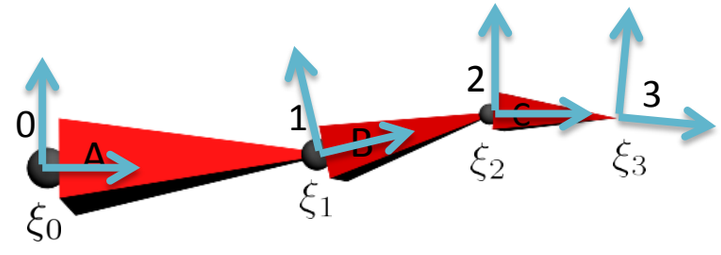
Kinematic Chain

- The **base twist** ξ_0 has the form

$$\xi_0 = (\omega_0, T_0)$$

represents the coordinates of the joint 0

determine the orientation of the reference system of bone A



- All the **internal twists** (ξ_1 and ξ_2) are defined as

$$\xi_1 = (\omega_1, (l_1, 0, 0))$$

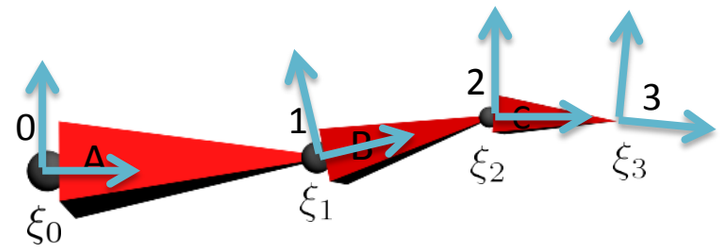
the translation is applied only along the x-axis with amount l_1

$$\xi_2 = (\omega_2, (l_2, 0, 0))$$

l_1 and l_2 denote the length of the bone A and B, respectively

Kinematic Chain

- The **end effector twist** ξ_3 has the form

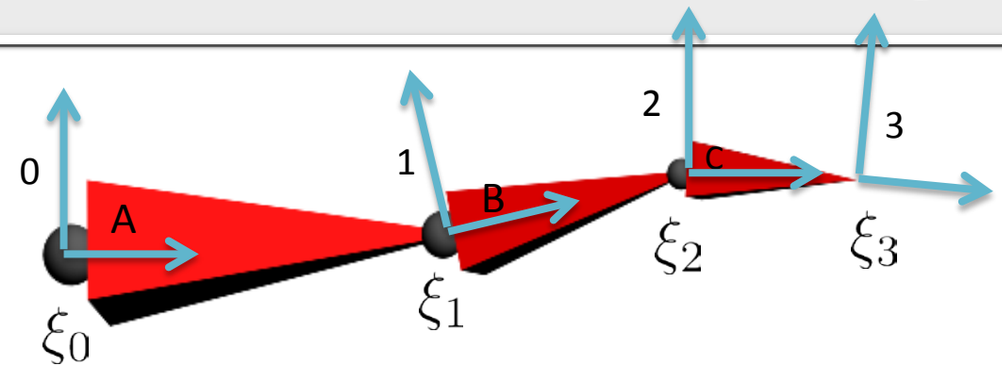


$$\xi_3 = (0, (l_3, 0, 0))$$

l_3 denote the length of the bone C

The orientation of the end effector is the same as the bone C

Kinematic Chain: Summary



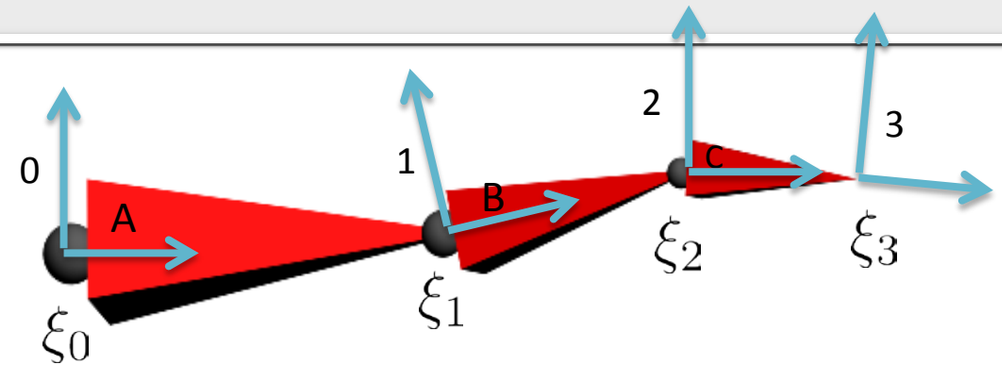
- ξ_0 determines the position of joint 0 and the orientation of bone A $\xi_0 = (\omega_0, T_0)$

- ξ_1 determine the position of joint 1, the length of bone A, and the orientation of bone B w.r.t. the reference system of joint 0 $\xi_1 = (\omega_1, (l_1, 0, 0))$

- ξ_2 determine the position of joint 2, the length of bone B, and the orientation of bone C w.r.t. the reference system of joint 1 $\xi_2 = (\omega_2, (l_2, 0, 0))$

- ξ_3 determine the position of joint 3 and the length of bone C $\xi_3 = (0, (l_3, 0, 0))$

Kinematic Chain: DOF



- Given the constraints

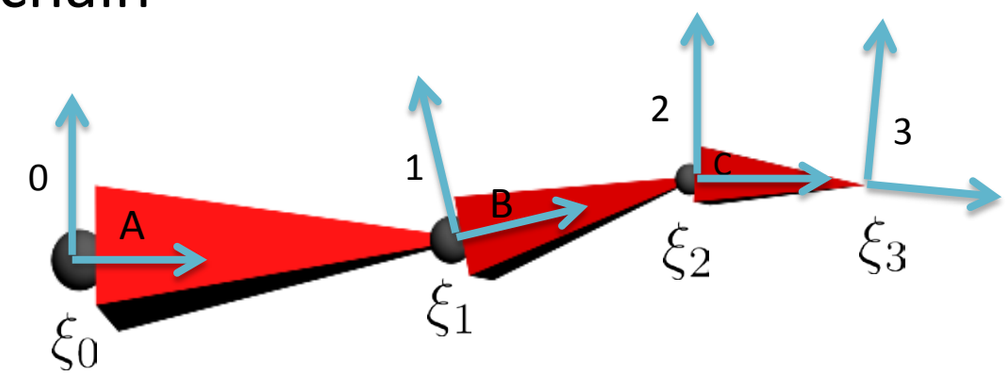
$$\begin{aligned}\xi_0 &= (\omega_0, T_0)\\ \xi_1 &= (\omega_1, (l_1, 0, 0))\\ \xi_2 &= (\omega_2, (l_2, 0, 0))\\ \xi_3 &= (0, (l_3, 0, 0))\end{aligned}$$

- the actual DOFs of this particular kinematic chain are

$\omega_0, \omega_1, \omega_2$	3x3 DOF	(**ball joints**)
T_0	+3 DOF if the base can move	
l_1, l_2, l_3	+3x1 DOF if the bone is extendible	(**prismatic joints**)

Kinematic Chain Problems

Given a kinematic chain



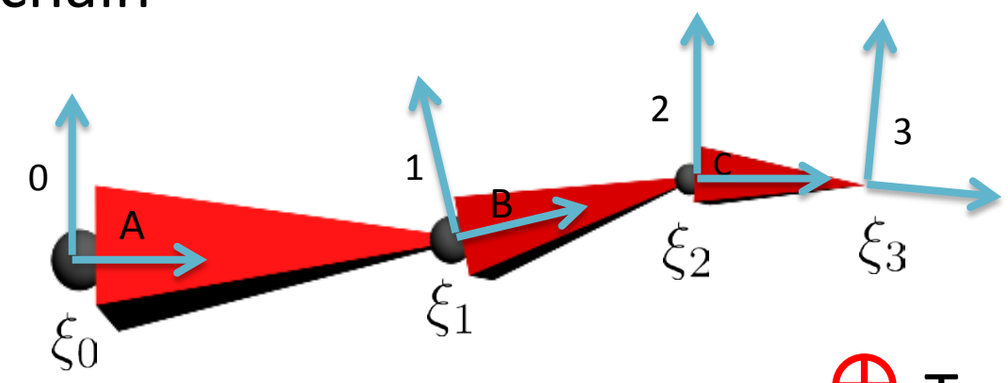
- A **Forward Kinematics Problem** consists in finding the coordinates of the end effector given a specific kinematic chain configuration $(\xi_0, \xi_1, \xi_2, \xi_3)$

$$p(\xi_0, \xi_1, \xi_2, \xi_3) = e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} e^{\widehat{\xi_2}} e^{\widehat{\xi_3}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Forward Kinematics of the end effector

Kinematic Chain Problems

Given a kinematic chain



- An **Inverse Kinematics Problem** consists in finding the configuration of the kinematic chain for which the distance between the end effector and a pre-defined target point **q** is minimized

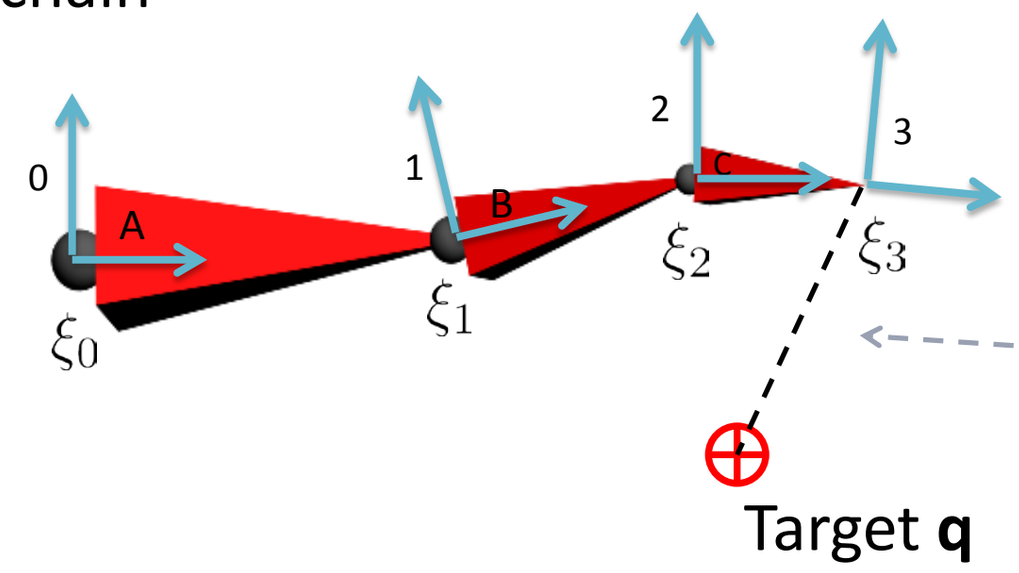
$$\begin{cases} \arg\min \|p(\xi_0, \xi_1, \xi_2, \xi_3) - q\| \\ \\ \text{subject to} \quad \begin{array}{l} \xi_0 = (\omega_0, T_0) \\ \xi_1 = (\omega_1, (l_1, 0, 0)) \\ \xi_2 = (\omega_2, (l_2, 0, 0)) \\ \xi_3 = (0, (l_3, 0, 0)) \end{array} \end{cases}$$

l_1, l_2, l_3 fixed/or not

T_0 fixed/or not

Inverse Kinematics Problem

Given a kinematic chain



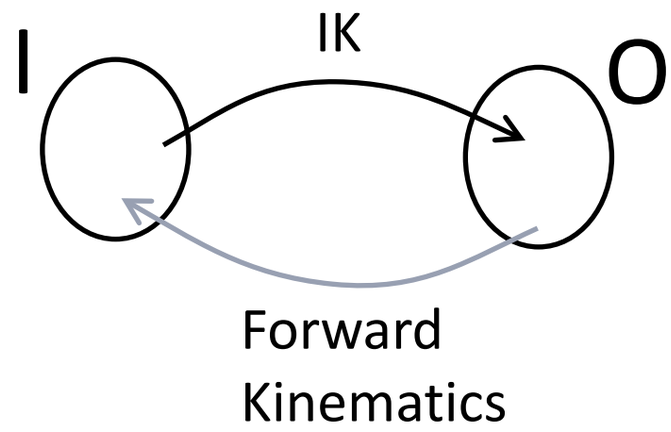
Minimize the distance between where the end effector is and where it should be

$$\arg\min \|\underbrace{p(\xi_0, \xi_1, \xi_2, \xi_3)}_{} - q\|$$

Generative model for p

= Forward Kinematics

Generative approach to IK



Inverse Kinematics Problem

$$\begin{cases} \arg\min \|p(\xi_0, \xi_1, \xi_2, \xi_3) - q\| \\ \\ \text{subject to} \quad \begin{array}{l} \xi_0 = (\omega_0, T_0) \\ \xi_1 = (\omega_1, (l_1, 0, 0)) \\ \xi_2 = (\omega_2, (l_2, 0, 0)) \\ \xi_3 = (0, (l_3, 0, 0)) \end{array} \end{cases}$$

l_1, l_2, l_3 fixed/or not

T_0 fixed/or not

- it is equivalent to a **non-linear least square optimization problem**

  (it is equivalent to the squared norm and this is $\|\cdot\|^2 = x^2 + y^2 + z^2$)

  (note: here it does not matter if the norm is squared or not, later it will)

*

- The problem is **under-constrained**, 3 equations and (at least) 9 unknowns
  - If q is reachable by the kinematic chain, there are **infinite solutions** to the problem
  - If q is not reachable, the solution is unique up to rotations along the bones axes

A Possible Solution

Newton's method

- let denote with x our unknowns $x = (\xi_0, \xi_1, \xi_2, \xi_3)$

$$\arg\min \|p(x) - q\|$$

- let $\overline{x}$ be the current estimate for the solution
- compute the Taylor expansion of $p(x)$ around $\overline{x}$

$$p(x + \Delta x) = p(\overline{x}) + Jp(\overline{x})\Delta x + \dots$$

$$\arg\min \|\overbrace{p(\overline{x}) + Jp(\overline{x})\Delta x} - q\|$$

$$\Updownarrow$$

$$p(\overline{x}) + Jp(\overline{x})\Delta x - q = 0$$

$$\Updownarrow$$

$$\Delta x = Jp(\overline{x})^{\dagger}(q - p(\overline{x}))$$

$Jp(\overline{x})^{\dagger}$ can be computed using SVD, or approximated as $\cong Jp(\overline{x})^{T}$ if speed is critical

The Jacobian of the Forward Kinematics

- Given the forward kinematic $p(\xi_0, \xi_1, \xi_2, \xi_3) = e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} e^{\widehat{\xi_2}} e^{\widehat{\xi_3}} p$

- assuming
$$\begin{aligned} \xi_0 &= (\omega_0, T_0) \\ \xi_1 &= (\omega_1, (l_1, 0, 0)) \\ \xi_2 &= (\omega_2, (l_2, 0, 0)) \\ \xi_3 &= (0, (l_3, 0, 0)) \end{aligned}$$
l_1, l_2, l_3 fixed
T_0 fixed

- and $\omega_i = (\theta_i^x, \theta_i^y, \theta_i^z)$

- the Jacobian of the forward kinematic is

1x3 column vector

$$Jp = \left[\frac{\partial p}{\partial \theta_0^x} \quad \frac{\partial p}{\partial \theta_0^y} \quad \frac{\partial p}{\partial \theta_0^z} \quad \frac{\partial p}{\partial \theta_1^x} \quad \frac{\partial p}{\partial \theta_1^y} \quad \frac{\partial p}{\partial \theta_1^z} \quad \frac{\partial p}{\partial \theta_2^x} \quad \frac{\partial p}{\partial \theta_2^y} \quad \frac{\partial p}{\partial \theta_2^z} \right]$$

only one term depends on θ_2^y

$$\frac{\partial p}{\partial \theta_2^y}(\xi_0, \xi_1, \xi_2, \xi_3) = e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} \frac{\partial e^{\widehat{\xi_2}}}{\partial \theta_2^y} e^{\widehat{\xi_3}} p$$

The Jacobian of the Forward Kinematics

$$\frac{\partial p}{\partial \theta_2^y}(\xi_0,\xi_1,\xi_2,\xi_3) = e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} \frac{\partial e^{\widehat{\xi_2}}}{\partial \theta_2^y} e^{\widehat{\xi_3}} p$$

$$= e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} \begin{bmatrix} \frac{\partial e^{\widehat{\omega_2}}}{\partial \theta_2^y} & 0 \\ 0 & 0 \end{bmatrix} e^{\widehat{\xi_3}} p$$

$$= e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} \begin{bmatrix} \frac{\partial \widehat{\omega_2}}{\partial \theta_2^y} e^{\widehat{\omega_2}} & 0 \\ 0 & 0 \end{bmatrix} e^{\widehat{\xi_3}} p$$

$$= e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} \begin{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} e^{\widehat{\omega_2}} & 0 \\ 0 & 0 \end{bmatrix} e^{\widehat{\xi_3}} p$$

$$\omega_2 = (\theta_2^x, \theta_2^y, \theta_2^z)$$

$$\widehat{\omega_2} = \begin{bmatrix} 0 & -\theta_2^z & \theta_2^y \\ \theta_2^z & 0 & -\theta_2^x \\ -\theta_2^y & \theta_2^x & 0 \end{bmatrix}$$

$$\frac{\partial \widehat{\omega_2}}{\partial \theta_2^y} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

The Jacobian of the Forward Kinematics

- and so on… (all the other derivatives are computed in a similar way)

- The Jacobian of forward kinematic is very easy to compute if the angle/axis representation is used. On the contrary, if quaternions are used instead, the Jacobian is not as trivial

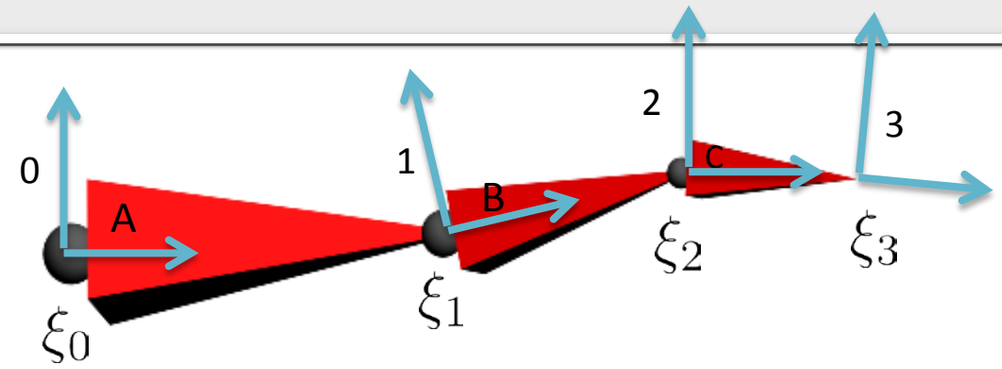
$$q_1 \cdot q_2 \cdot s \cdot q_2^{-1} \cdot q_1^{-1}$$

Mathematical Foundations of
Computer Graphics and Vision

Inverse Kinematics II and Motion Capture

Luca Ballan

Comparison



Fake exponential map

$$(\omega, t) \rightarrow \begin{bmatrix} e^{\widehat{\omega}} & t \\ 0 & 1 \end{bmatrix}$$

- t is equal to the length of the bone
- computing the derivative on the angle is easy

Real exponential map

$$(\omega, v) \rightarrow \begin{bmatrix} e^{\widehat{\omega}} & \frac{1}{\|\omega\|}(I - e^{\widehat{\omega}})(\omega \times v) + \frac{\omega\omega^T}{\|\omega\|}v \\ 0 & 1 \end{bmatrix}$$

- the meaning of v is not intuitive

However

- this incorporates the real concept of geodesic
- interpolation/averaging has to be done in this space

Special Euclidean group SE(3)

$$\xi = (\omega, v) \xrightarrow{\exp} \begin{bmatrix} e^{\widehat{\omega}} & \frac{1}{\|\omega\|}(I - e^{\widehat{\omega}})(\omega \times v) + \frac{\omega\omega^T}{\|\omega\|}v \\ 0 & 1 \end{bmatrix} = e^{\widehat{\xi}} \in SE(3)$$

Twist

Angle/axis representation of the rotation $\widehat{\omega} \in so(3)$

- **Screw motion**: rotation along an axis + a translation along the same axis.

- **Proposition:** Any rigid transformation in SE(3) can be expressed as a rotation about an axis combined with a translation parallel to that axis.

Special Euclidean group SE(3)

$$\xi = (\omega, v)$$

hat

Twist

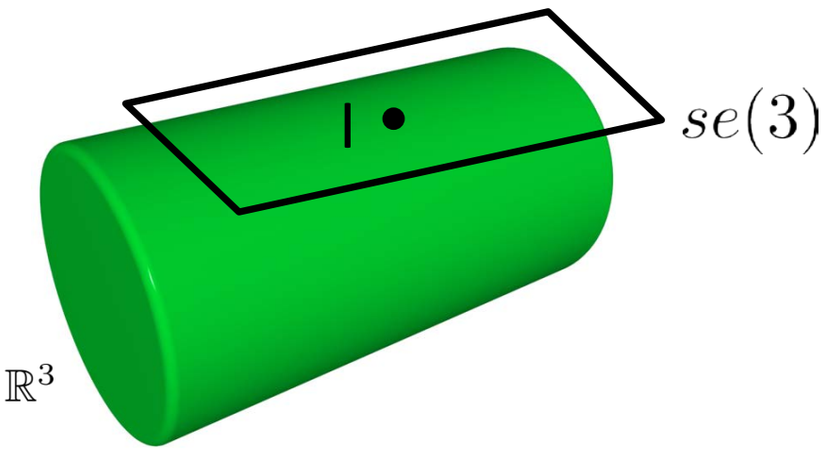
$$\widehat{\xi} = \begin{bmatrix} \widehat{\omega} & v \\ 0 & 0 \end{bmatrix} \in se(3)$$

exp

$$\begin{bmatrix} e^{\widehat{\omega}} & \frac{1}{\|\omega\|}(I - e^{\widehat{\omega}})(\omega \times v) + \frac{\omega\omega^T}{\|\omega\|}v \\ 0 & 1 \end{bmatrix} = e^{\widehat{\xi}} \in SE(3)$$

$$\exp(M) = \sum_{k=0}^{\infty} \frac{1}{k!} M^k$$

Tangent space of SE(3)
at the identity



Properties

$$e^{\widehat{0}} = I$$ **Identity**

$$e^{-X} = (e^X)^{-1}$$ **Inverse** (basic property of exp map)

$$e^{X+Y} \neq e^X e^Y$$ **in general not “Linear” (like in so(3))**

$$\partial e^X = \partial X e^X = e^X \partial X$$ **Derivative**

Metric on SE(3)

$$d_R(M_1, M_2) = \frac{1}{\sqrt{2}} \| \log(M1^{-1} M2) \|_F$$

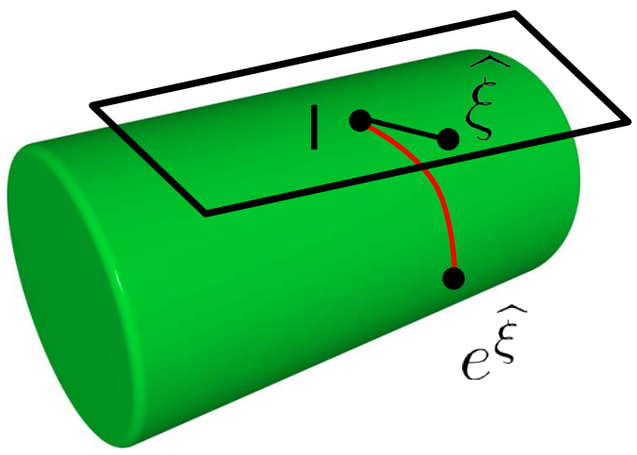
Riemannian/Geodesic/Angle metric
(= the length of the geodesic
connecting M_1 and M_2)

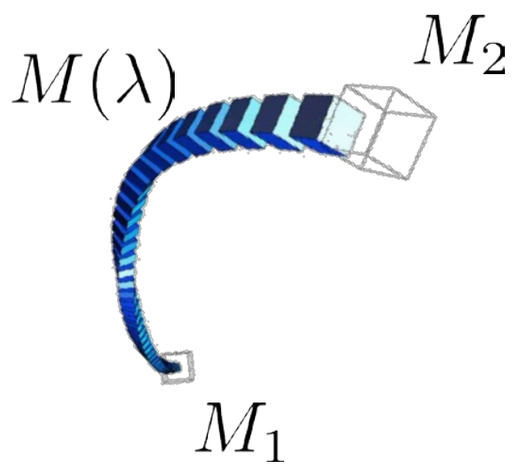
- Interpolation and averaging can be performed in the space of rigid transformations also

$$M(\lambda) = M_1 e^{\lambda \log(M_1^{-1} M_2)}$$

SLERP
(spherical linear
interpolation)

$SE(3)$





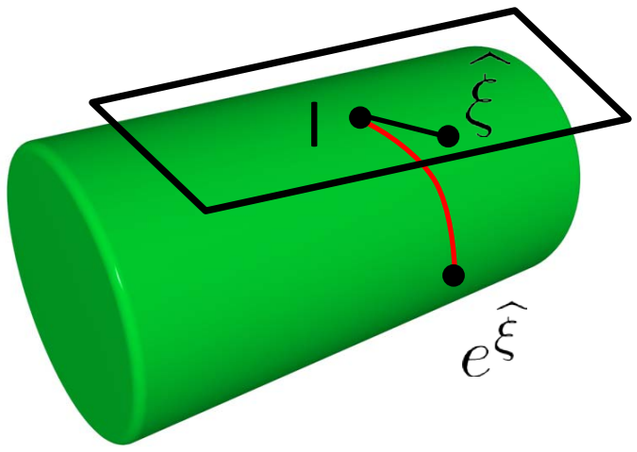
Metric on SE(3)

$$d_R(M_1, M_2) = \frac{1}{\sqrt{2}} \| \log(M1^{-1} M2) \|_F$$

Riemannian/Geodesic/Angle metric
(= the length of the geodesic
connecting M_1 and M_2)

- Interpolation and averaging can be performed in the space of rigid transformations also

$SE(3)$



$$M(\lambda) = M_1 e^{\lambda \log(M_1^{-1} M_2)}$$

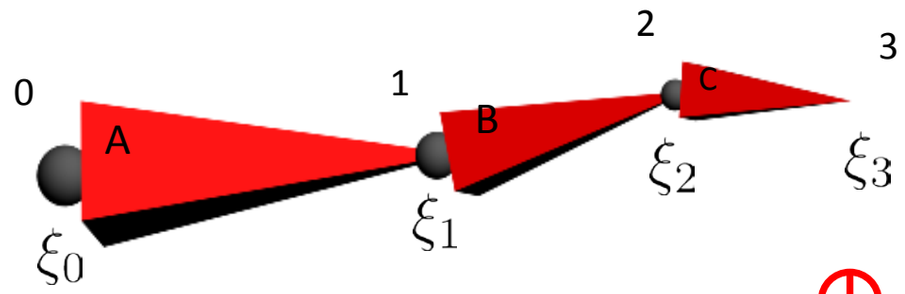
SLERP
(spherical linear
interpolation)

$$\underset{M \in SE(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d_R(M, M_i)^2$$

Fréchet mean

A note on Interpolation

Inverse Kinematics



⊕ Target **q**

Newton's method

- let $\overline{x}$ be the current estimate for the solution
- compute the Taylor expansion of $p(x)$ around $\overline{x}$

$$p(x + \Delta x) = p(\overline{x}) + Jp(\overline{x})\Delta x + \dots$$

$$\arg\min \|\overbrace{p(\overline{x}) + Jp(\overline{x})\Delta x} - q\|$$

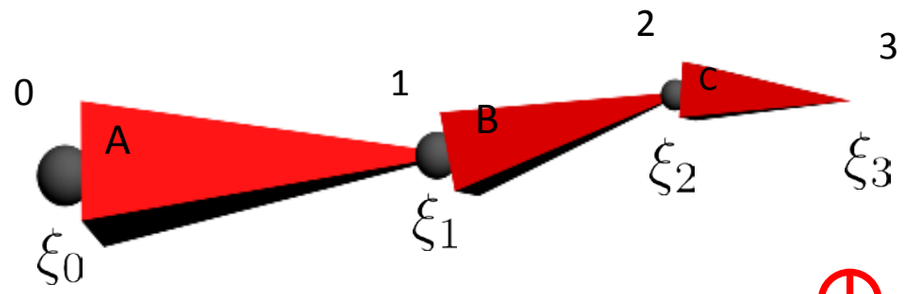
$$\Updownarrow$$

$$p(\overline{x}) + Jp(\overline{x})\Delta x - q = 0$$

$$\Updownarrow$$

$$\Delta x = Jp(\overline{x})^{\dagger}(q - p(\overline{x}))$$

Inverse Kinematics



⊕ Target **q**

Newton's method

- let $\overline{x}$ be the current estimate for the solution
- compute the Taylor expansion of $p(x)$ around $\overline{x}$

$$p(x + \Delta x) = p(\overline{x}) + Jp(\overline{x})\Delta x + \dots$$

$$\arg\min \|\overbrace{p(\overline{x}) + Jp(\overline{x})\Delta x} - q\|$$

$$\Updownarrow$$

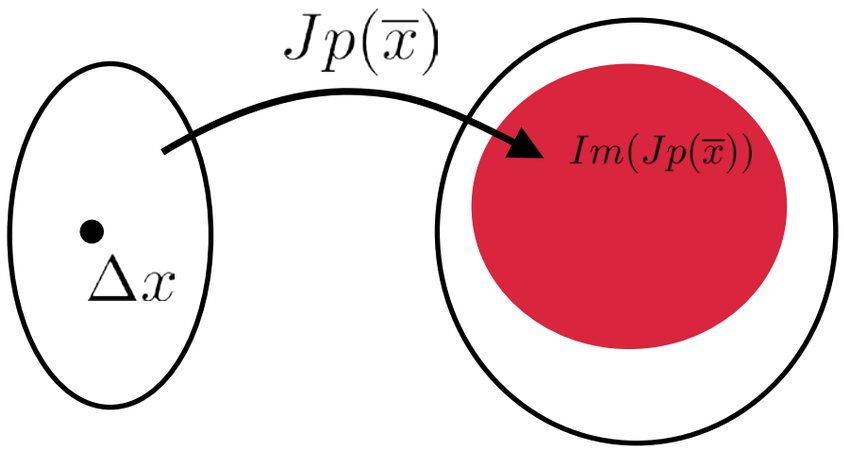
$$p(\overline{x}) + Jp(\overline{x})\Delta x - q = 0$$

$$\Updownarrow$$

Find Δx such that $Jp(\overline{x})\Delta x = (q - p(\overline{x}))$

Inverses

Find Δx such that $Jp(\overline{x})\Delta x = (q - p(\overline{x}))$



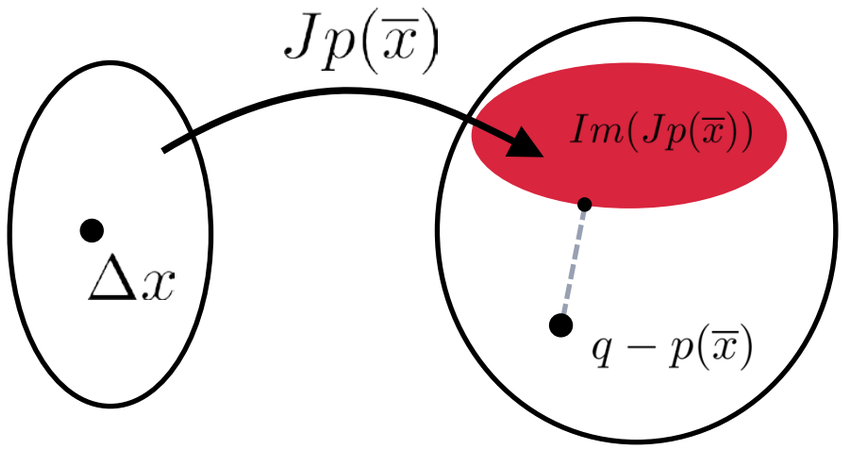
$Jp(\overline{x})$ is not injective (in general)
(multiple inverse exists for each point of the image)

If it is injective, then the left-Inverse exists

$$\Delta x = Jp(\overline{x})^{-1}_{left}(q - p(\overline{x}))$$

Inverses

Find Δx such that $Jp(\overline{x})\Delta x = (q - p(\overline{x}))$



$Im(Jp(\overline{x}))$ might not contain the solution

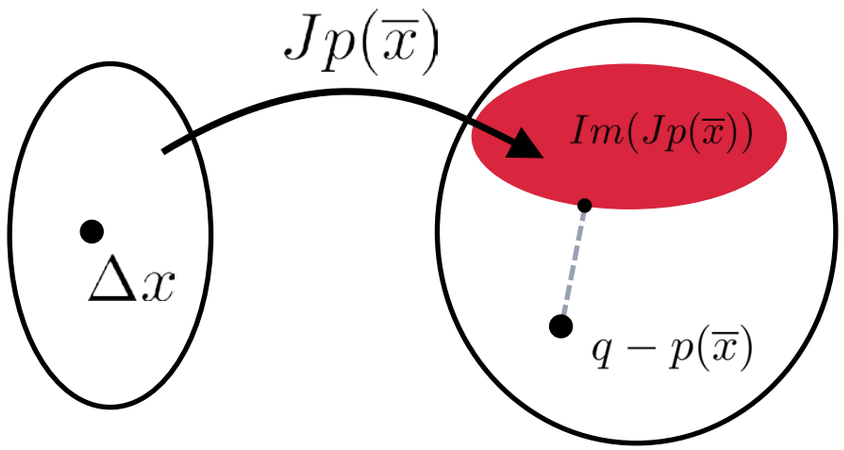
Better to find the Δx such that

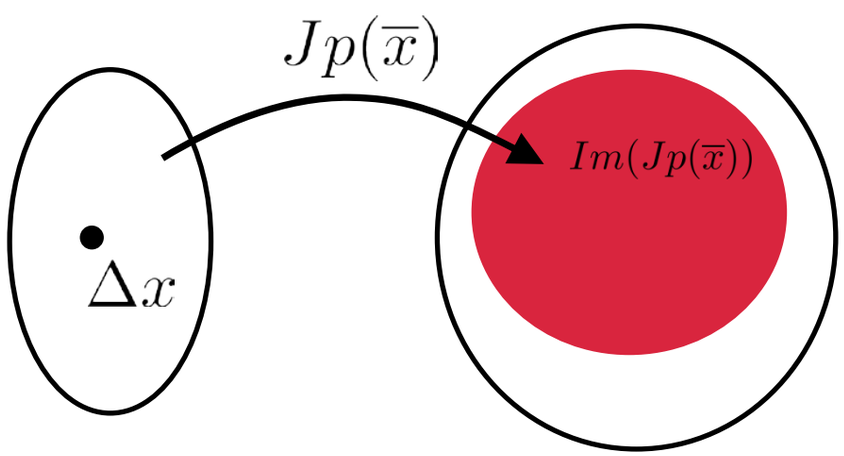
$$\|Jp(\overline{x})\Delta x - (q - p(\overline{x}))\|$$

is minimized

Inverses

Find Δx such that $\|Jp(\overline{x})\Delta x - (q - p(\overline{x}))\|$ is minimized,





To force the unicity of the solution

we force Δx to be the one with minimum norm

$$\Delta x = Jp(\overline{x})^{\dagger}(q - p(\overline{x}))$$

Singularities

- Pseudo inverse works well in many cases but not near a singularity (singular values close to 0)
- Make the constraint on the norm of Δx soft, not hard as before

$$\arg\min \|Jp(\overline{x})\Delta x - q + p(\overline{x})\|^2 + \lambda^2 \|\Delta x\|^2$$

- the solution of this is the same as solving for

$$\underbrace{(Jp(\overline{x})^T Jp(\overline{x}) + \lambda I)}\Delta x = Jp(\overline{x})^T (q - p(\overline{x}))$$

  - this is always non singular if **the damping factor** λ is correctly chosen

- The Newton's method with this damping step is known as the **Damped Least Square method**, or as the **Levenberg-Marquardt algorithm**.

Heuristic approaches

- **Cyclic Coordinate Descent** is an **alternating optimization approach** where only one coordinate at a time is optimized.

$$\arg\min \|p(\xi_0, \xi_1, \xi_2, \xi_3) - q\|$$

- Let $(\xi_0^t, \xi_1^t, \xi_2^t, \xi_3^t)$ be the estimate of the solution at iteration t

$$\xi_0^{t+1} = \arg\min_x \|p(\mathbf{x}, \xi_1^t, \xi_2^t, \xi_3^t) - q\|$$
$$\xi_1^{t+1} = \arg\min_x \|p(\xi_0^{t+1}, \mathbf{x}, \xi_2^t, \xi_3^t) - q\|$$
$$\xi_2^{t+1} = \arg\min_x \|p(\xi_0^{t+1}, \xi_1^{t+1}, \mathbf{x}, \xi_3^t) - q\|$$
$$\xi_3^{t+1} = \arg\min_x \|p(\xi_0^{t+1}, \xi_2^{t+1}, \xi_1^{t+1}, \mathbf{x}) - q\|$$

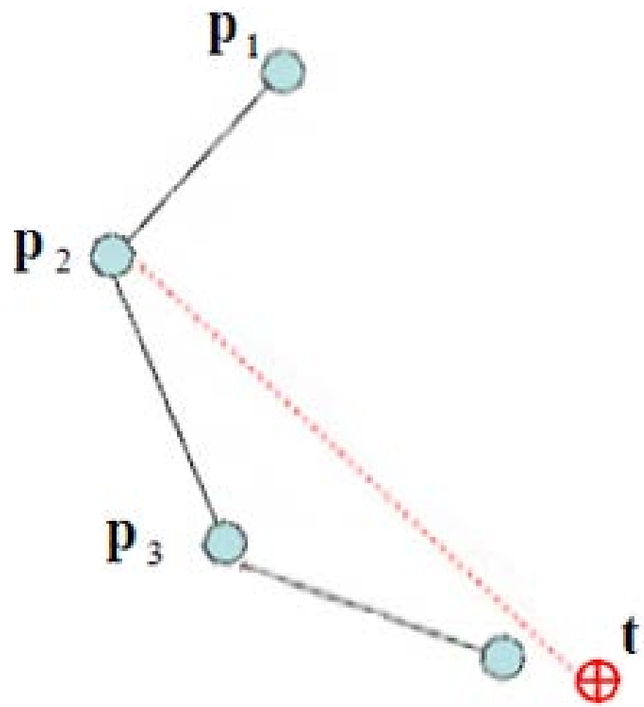
Heuristic approaches

- **Cyclic Coordinate Descent** is an **alternating optimization approach** where only one coordinate at a time is optimized.

$$\arg\min \|p(\xi_0, \xi_1, \xi_2, \xi_3) - q\|$$

- Let $(\xi_0^t, \xi_1^t, \xi_2^t, \xi_3^t)$ be the estimate of the solution at iteration t

$$\xi_0^{t+1} = \arg\min_x \|p(\mathbf{x}, \xi_1^t, \xi_2^t, \xi_3^t) - q\|$$
$$\xi_1^{t+1} = \arg\min_x \|p(\xi_0^{t+1}, \mathbf{x}, \xi_2^t, \xi_3^t) - q\|$$
$$\xi_2^{t+1} = \arg\min_x \|p(\xi_0^{t+1}, \xi_1^{t+1}, \mathbf{x}, \xi_3^t) - q\|$$
$$\xi_3^{t+1} = \arg\min_x \|p(\xi_0^{t+1}, \xi_2^{t+1}, \xi_1^{t+1}, \mathbf{x}) - q\|$$



Heuristic approaches

- **Cyclic Coordinate Descent** is an **alternating optimization approach** where only one coordinate at a time is optimized.

$$\arg\min \|p(\xi_0, \xi_1, \xi_2, \xi_3) - q\|$$

- Let $(\xi_0^t, \xi_1^t, \xi_2^t, \xi_3^t)$ be the estimate of the solution at iteration t

$$\xi_0^{t+1} = \arg\min_x \|p(\mathbf{x}, \xi_1^t, \xi_2^t, \xi_3^t) - q\|$$

solving for a single twist is a very easy problem

$$e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} e^{\widehat{\xi_2}} e^{\widehat{\xi_3}} p = q$$

Heuristic approaches

$$e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} e^{\widehat{\xi_2}} e^{\widehat{\xi_3}} p = q$$

$$\left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right) e^{\widehat{\xi_2}} \left(e^{\widehat{\xi_3}} p\right) = q$$

$$e^{\widehat{\xi_2}} \tilde{p} = \left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right)^{-1} q$$

$$e^{\widehat{\omega_2}} \tilde{p} + T_2 = \left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right)^{-1} q$$

$$e^{\widehat{\omega_2}} \tilde{p} = \left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right)^{-1} q - T_2$$

$$e^{\widehat{\omega_2}} \tilde{p} = \tilde{q}$$

*

Heuristic approaches

$$e^{\widehat{\xi_0}} e^{\widehat{\xi_1}} e^{\widehat{\xi_2}} e^{\widehat{\xi_3}} p = q$$

$$\left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right) e^{\widehat{\xi_2}} \left(e^{\widehat{\xi_3}} p\right) = q$$

$$e^{\widehat{\xi_2}} \tilde{p} = \left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right)^{-1} q$$

$$e^{\widehat{\omega_2}} \tilde{p} + T_2 = \left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right)^{-1} q$$

$$e^{\widehat{\omega_2}} \tilde{p} = \left(e^{\widehat{\xi_0}} e^{\widehat{\xi_1}}\right)^{-1} q - T_2$$

$$e^{\widehat{\omega_2}} \tilde{p} = \tilde{q}$$

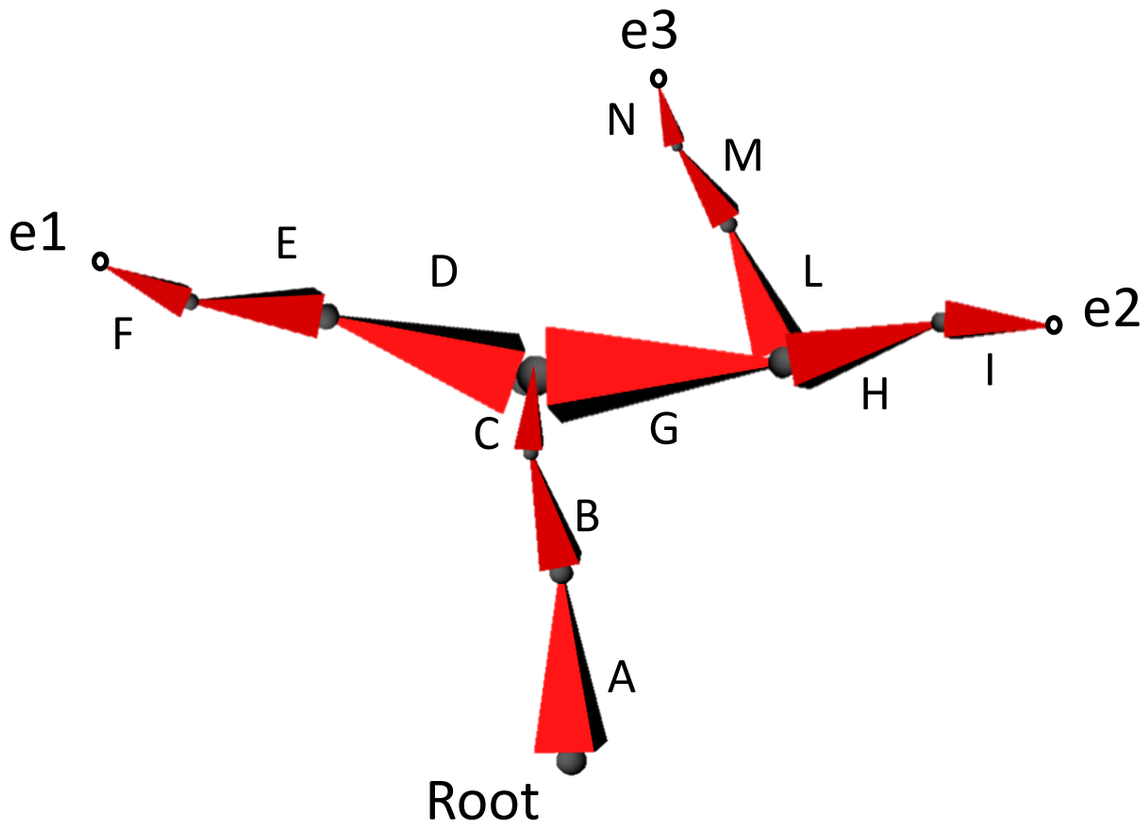
$$\begin{cases} \tilde{p}\tilde{q}^T = U\Sigma V^T \\ R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{bmatrix} U^T \end{cases}$$

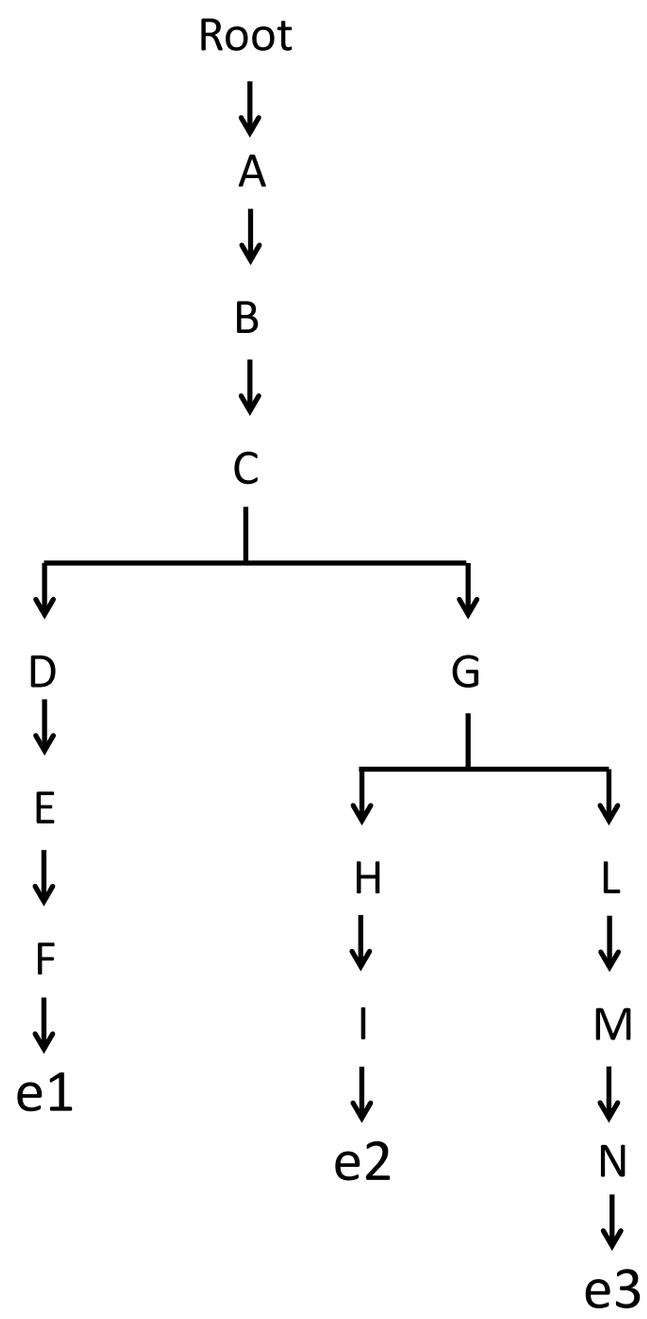
Content

- Inverse Kinematics
- **Kinematic Trees/Graphs**
- Pose Estimation/Motion Capture

Kinematic Tree

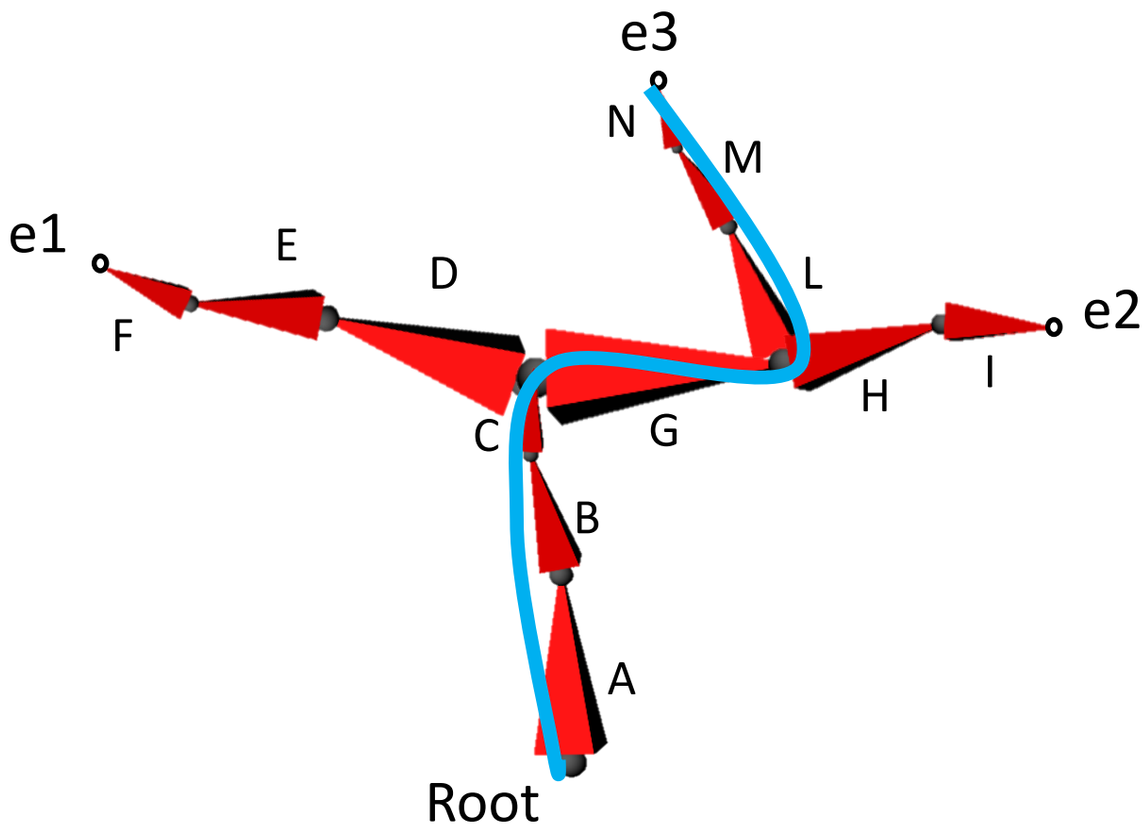
- a **kinematic tree** is a tree of rigid transformations

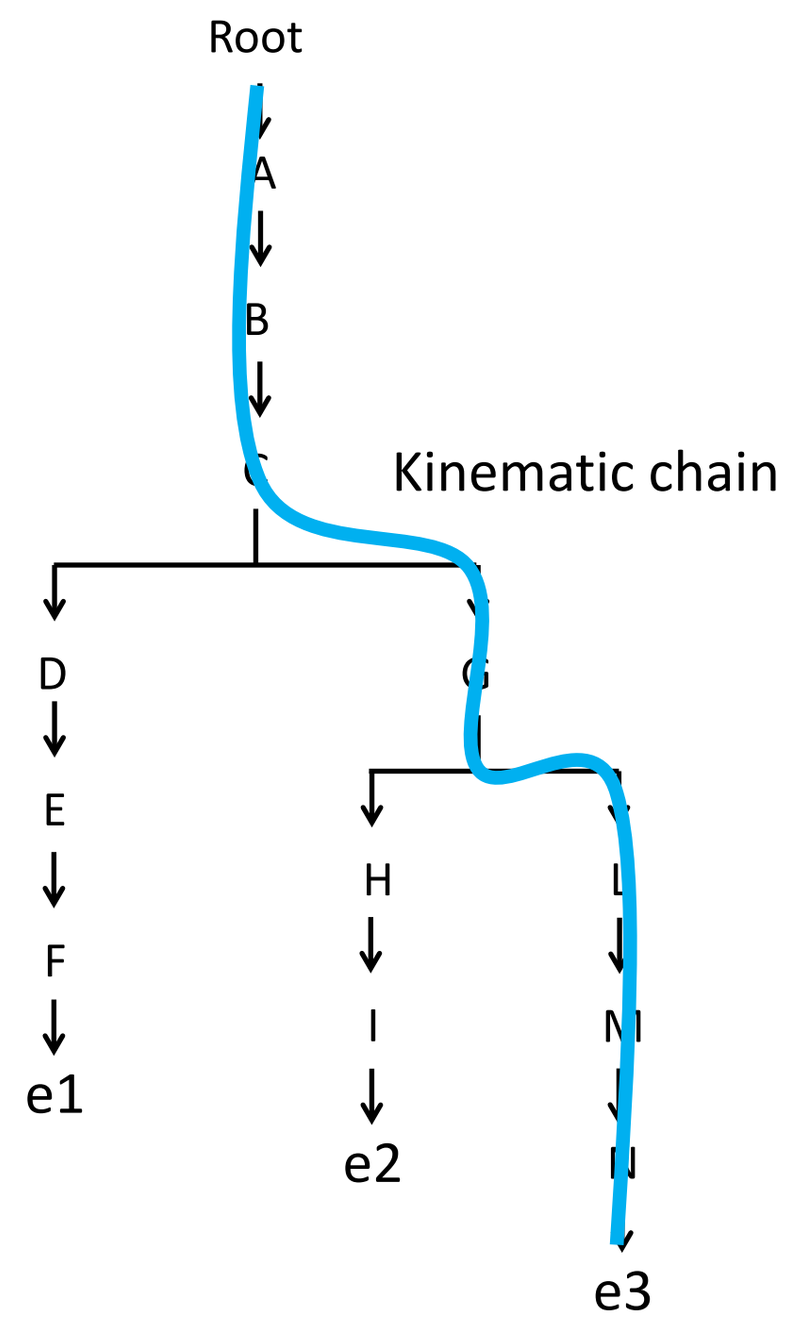




Kinematic Tree

- a **kinematic tree** is a tree of rigid transformations

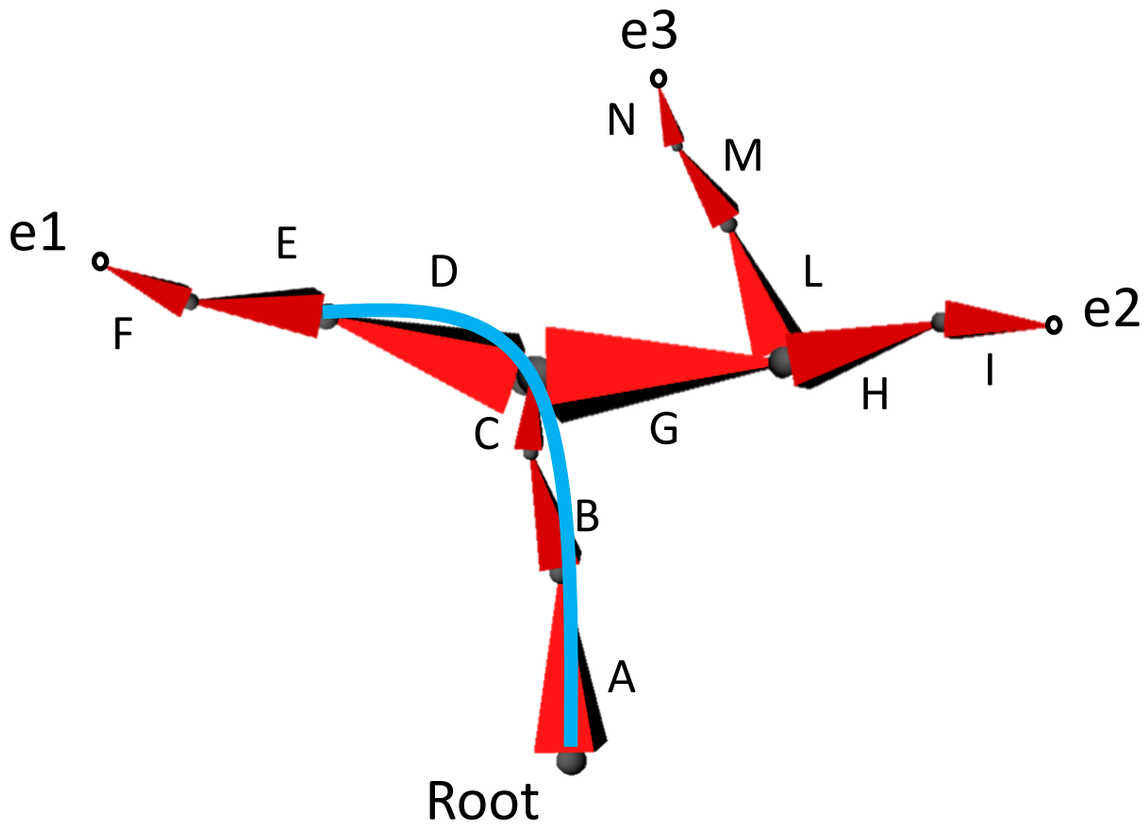


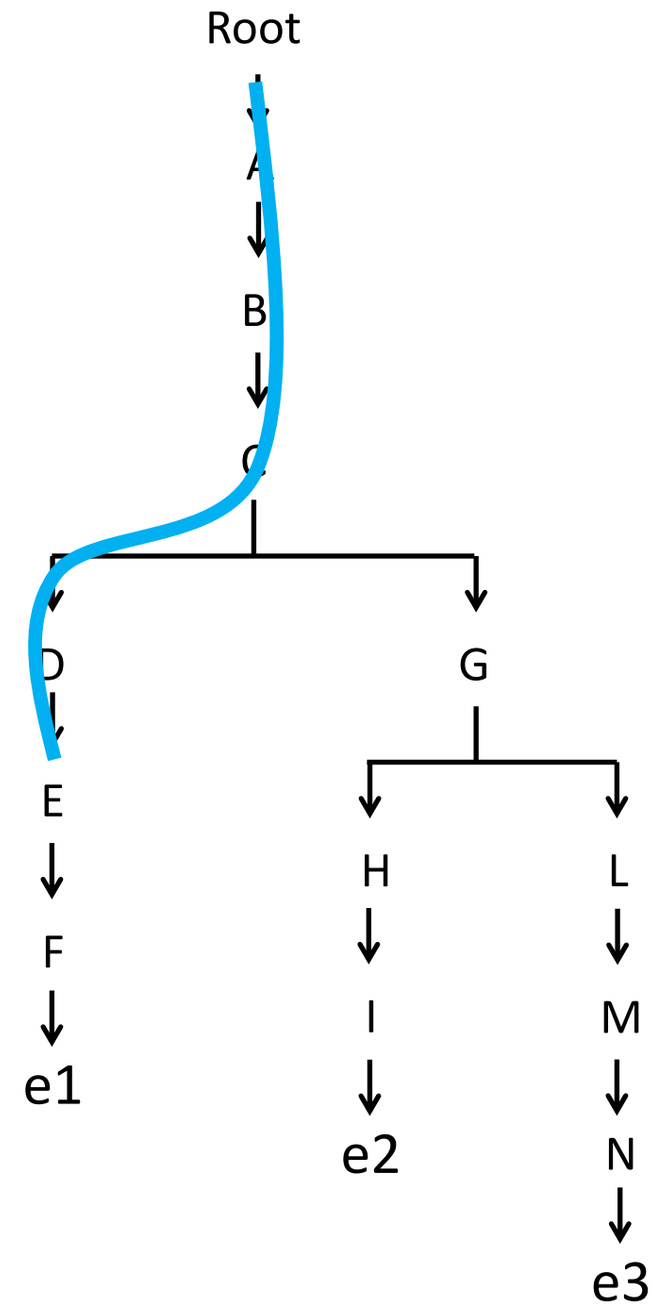


- each path in a **kinematic tree** from the root to <u>any</u> other node is a **kinematic chain**

Kinematic Tree

- a **kinematic tree** is a tree of rigid transformations

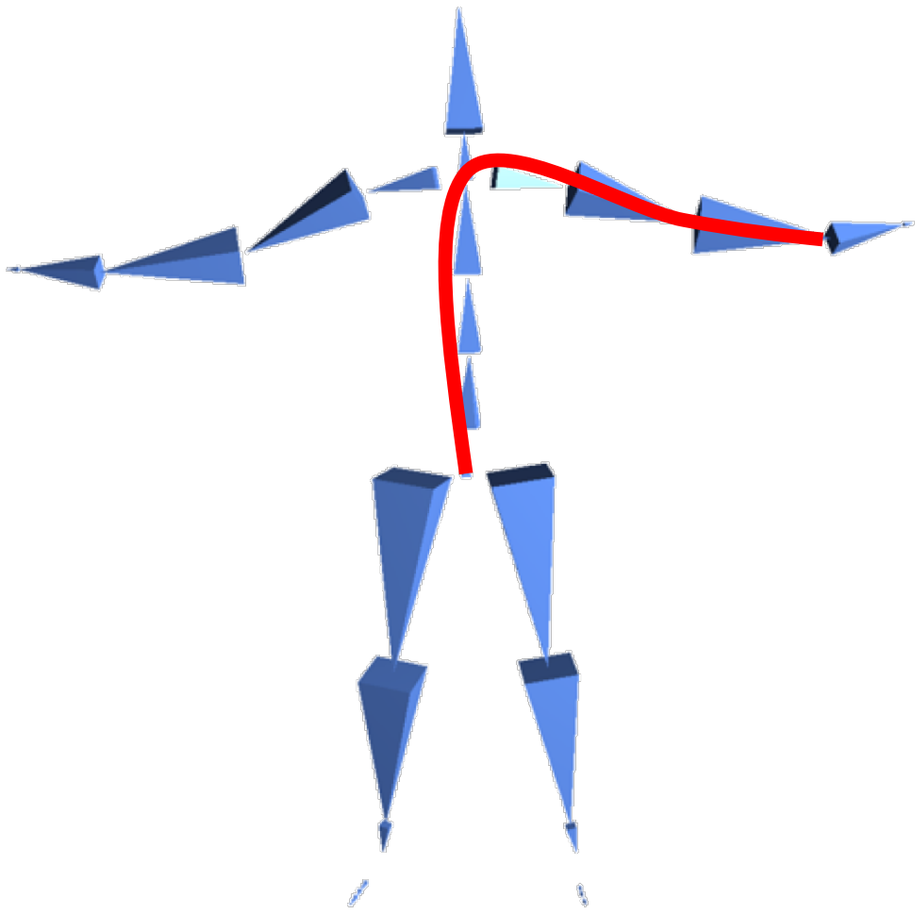


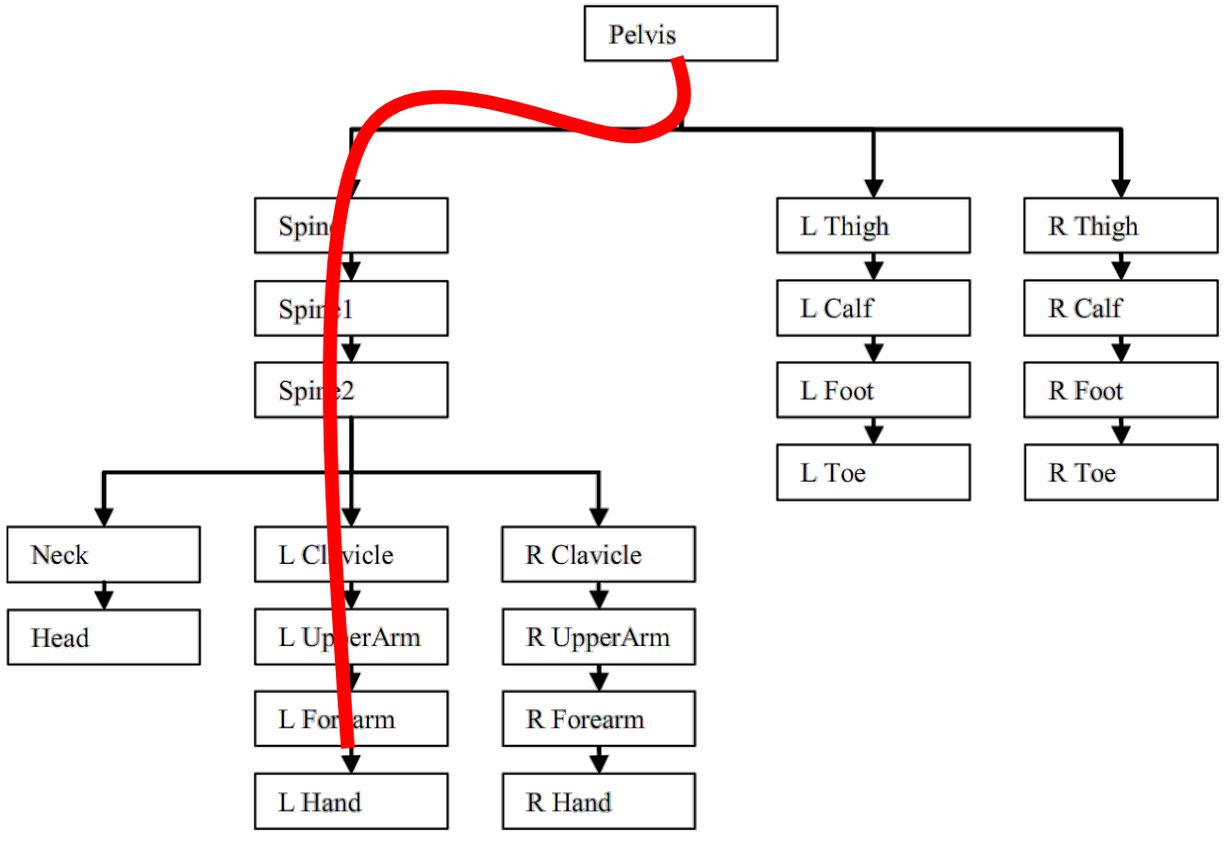


- each path in a **kinematic tree** from the root to <u>any</u> other node is a **kinematic chain**

Kinematic Tree: Human Skeleton

- a **kinematic tree** is a tree of rigid transformations

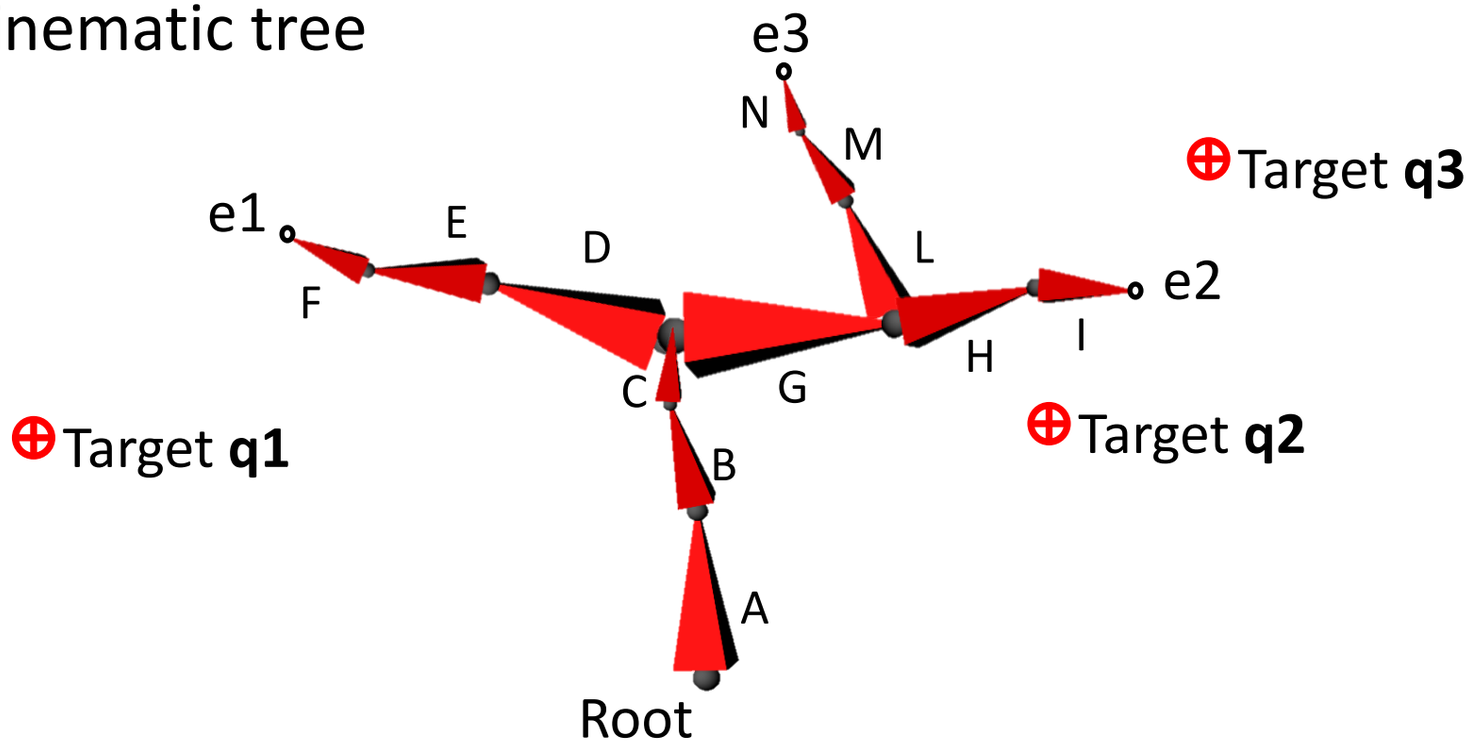




- each path in a **kinematic tree** from the root to <u>any</u> other node is a **kinematic chain**
- The pelvis is typically denoted as the root of the tree (but this is only a convention)

Inverse Kinematics for Trees

- Given a kinematic tree

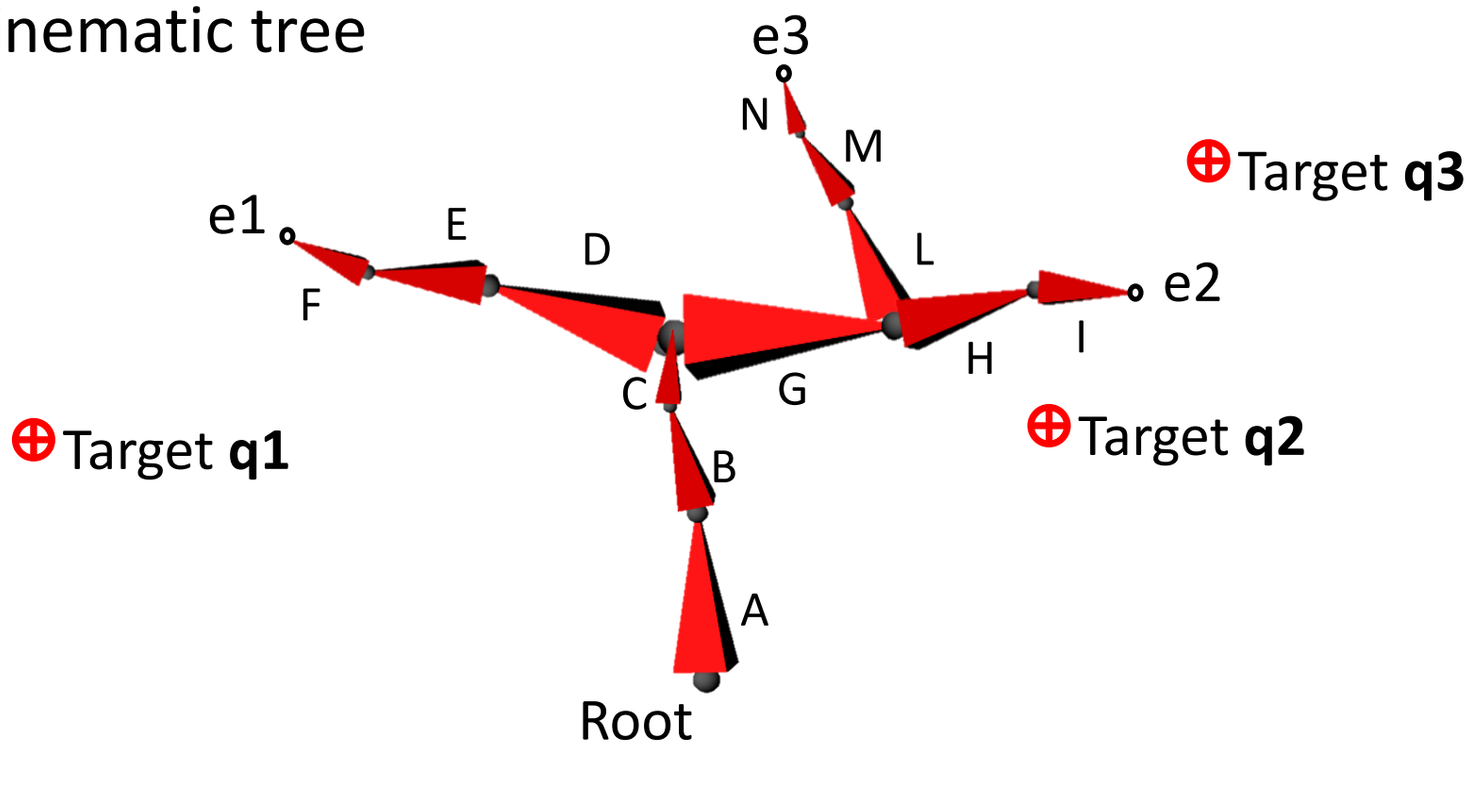


- An **Inverse Kinematics Problem** on a tree consists in finding the configuration of the kinematic chain for which the distance between the end effectors and some pre-defined targets points are minimized

- it is an **Inverse Kinematics Problem with Multiple Targets**

Inverse Kinematics for Trees

- Given a kinematic tree



$$\arg\min_{x} \begin{cases} \|p_1(x) - q_1\| \\ \|p_2(x) - q_2\| \\ \|p_3(x) - q_3\| \end{cases}$$

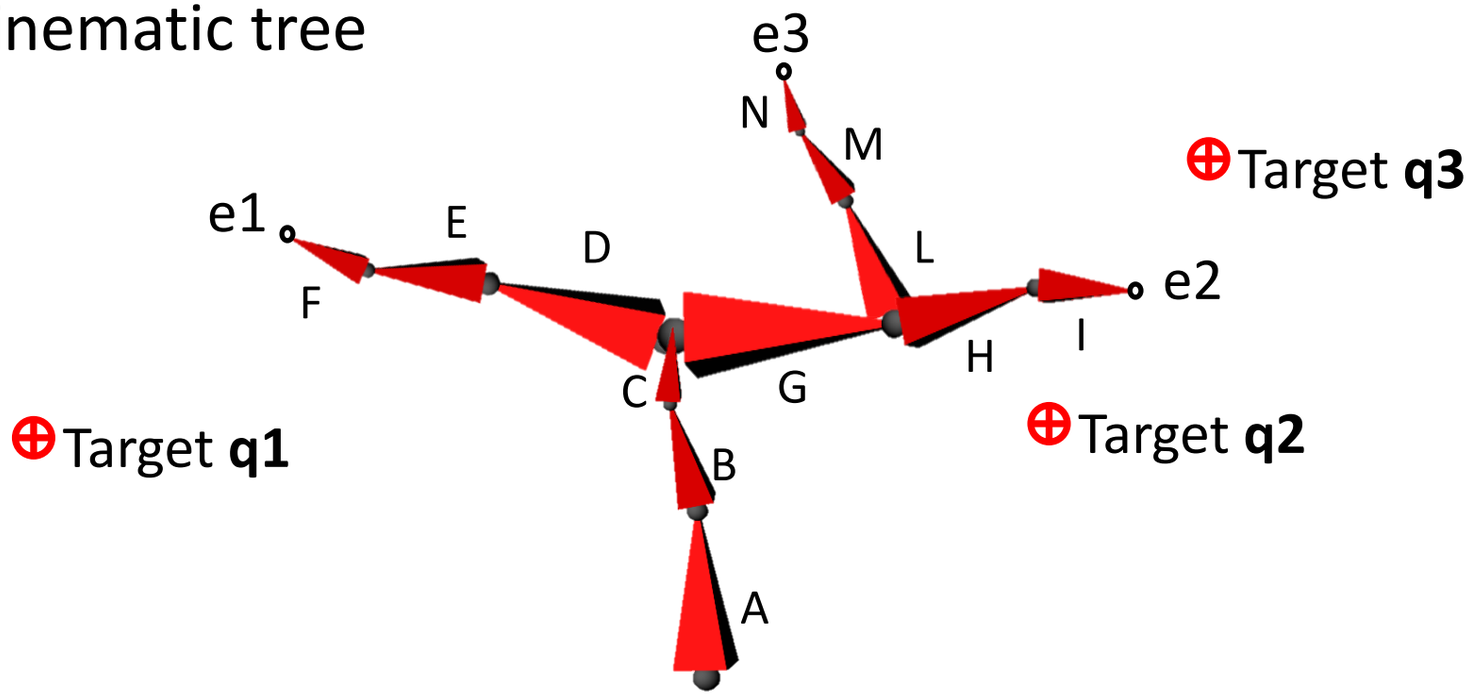
Find **x** such that it simultaneously minimizes all the distances with the targets

Multi-objective optimization

It is difficult to find x satisfying all the minima

Inverse Kinematics for Trees

- Given a kinematic tree



$$\arg\min_x \begin{cases} \|p_1(x) - q_1\| \\ \|p_2(x) - q_2\| \\ \|p_3(x) - q_3\| \end{cases} \Rightarrow \arg\min_x \sum_i \|p_i(x) - q_i\|^2$$

equivalent to

$\|p(x) - q\|_2^2$

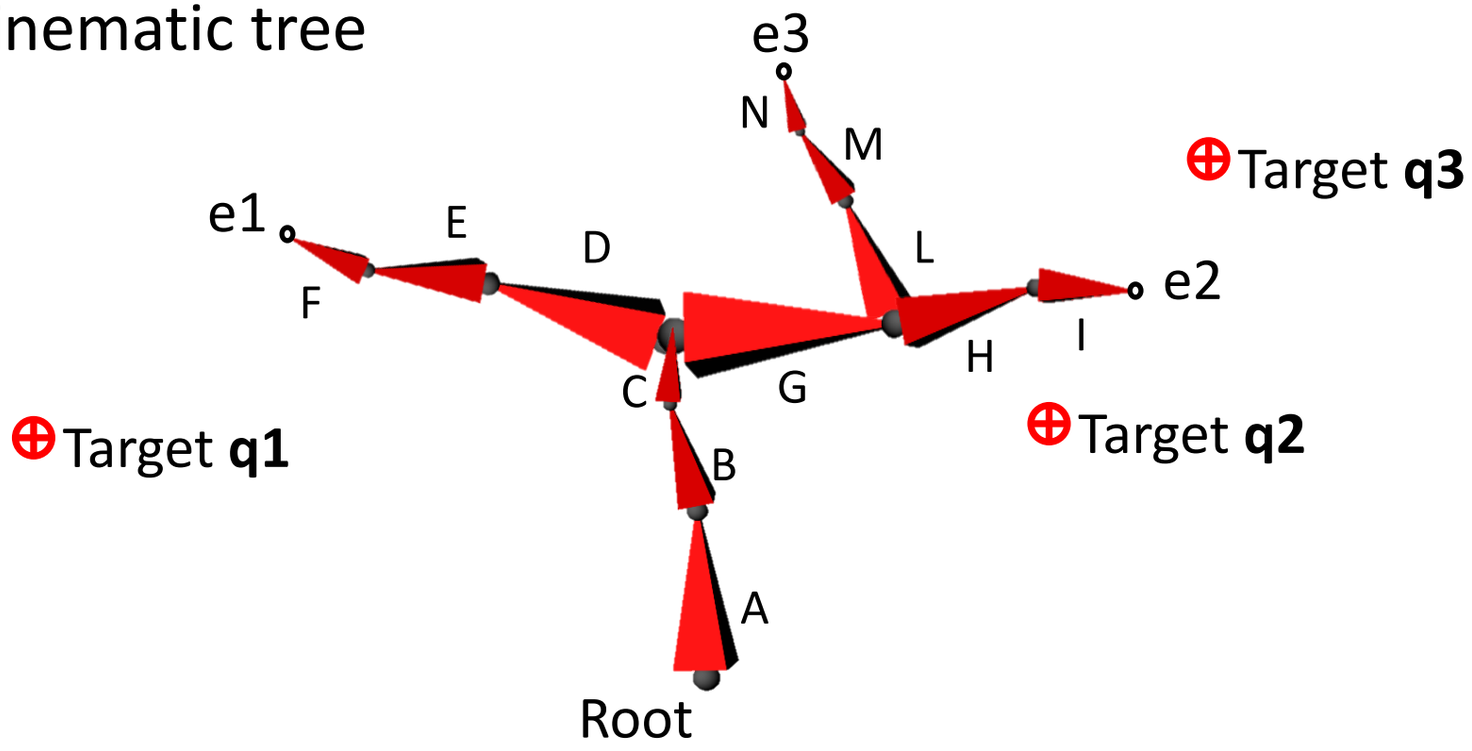
ℓ_2 norm

Find the x which minimizes the sum of the squared errors

i.e. the mean solution

Inverse Kinematics for Trees

- Given a kinematic tree



$$\arg\min_x \begin{cases} \|p_1(x) - q_1\| \\ \|p_2(x) - q_2\| \\ \|p_3(x) - q_3\| \end{cases} \Rightarrow \arg\min_x \sum_i \|p_i(x) - q_i\|$$

equivalent to

$$\|p(x) - q\|_1$$

ℓ_1 norm

Find the x which minimizes the sum of the errors

i.e. the median solution

Inverse Kinematics for Trees: Solution

- The ℓ_2 norm case

$$\arg\min_x \sum_i \|p_i(x) - q_i\|^2$$

- is still a **non-linear least square optimization problem**

  - Newton's method or Levenberg-Marquardt

$$p(x + \Delta x) = p(\overline{x}) + Jp(\overline{x})\Delta x + \dots$$

$$\arg\min \|\overbrace{p(\overline{x}) + Jp(\overline{x})\Delta x} - q\|$$

$$\Updownarrow$$

$$p(\overline{x}) + Jp(\overline{x})\Delta x - q = 0$$

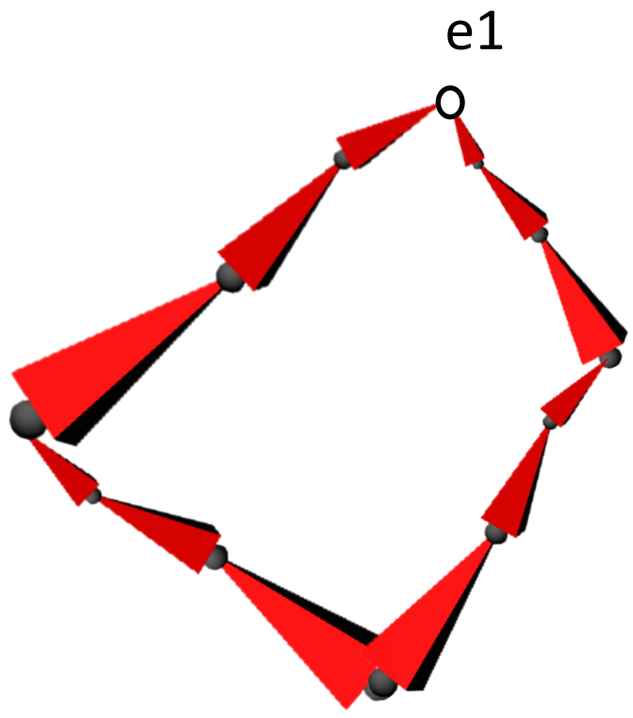
$$\Updownarrow$$

$$\Delta x = Jp(\overline{x})^\dagger (q - p(\overline{x}))$$

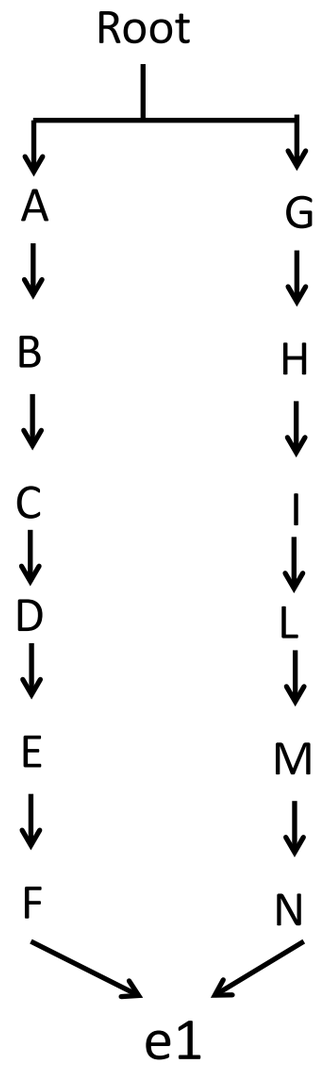
$$p(x) = \begin{bmatrix} p_1(x) \\ \vdots \\ p_n(x) \end{bmatrix}$$

Kinematic Graph

- a **kinematic graph** is a graph of rigid transformations (it contains cycles)

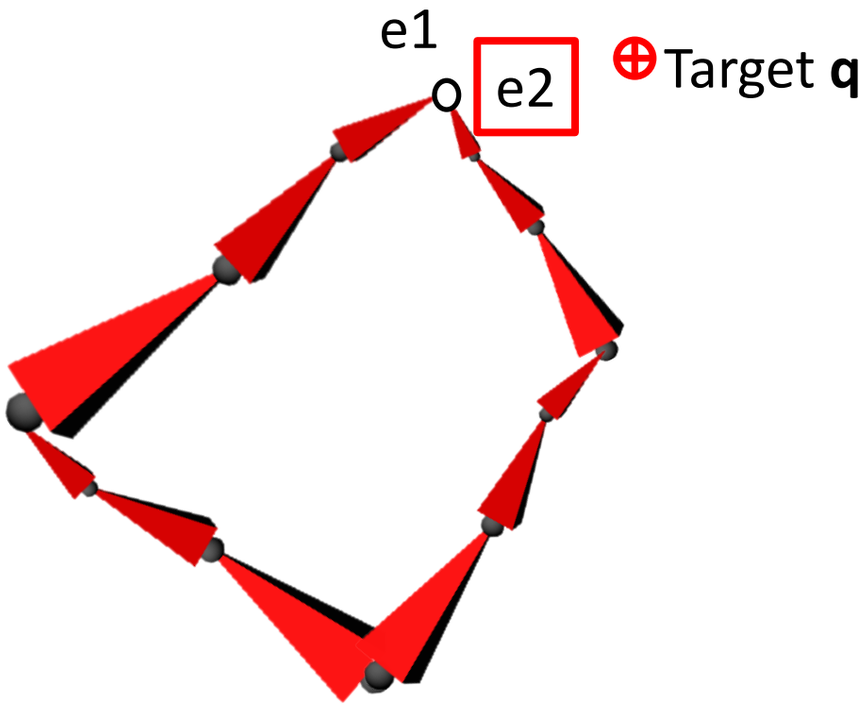


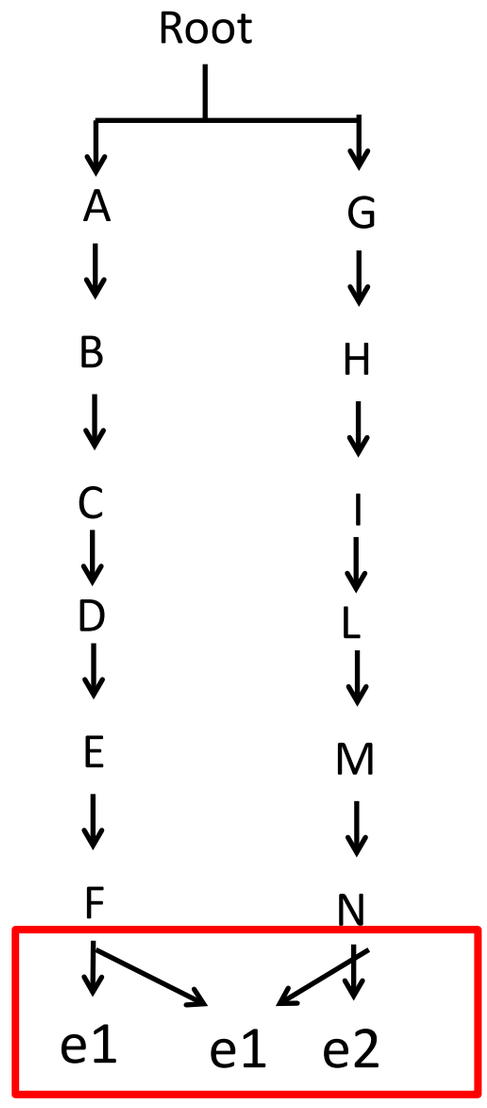
Root



(parallel manipulators)

Inverse Kinematics for Graphs





$$\arg\min \|p_1(x) - q\|$$

subject $p_1(x) = p_2(x)$

Constrained optimization

- Lagrange Multipliers
- or simply

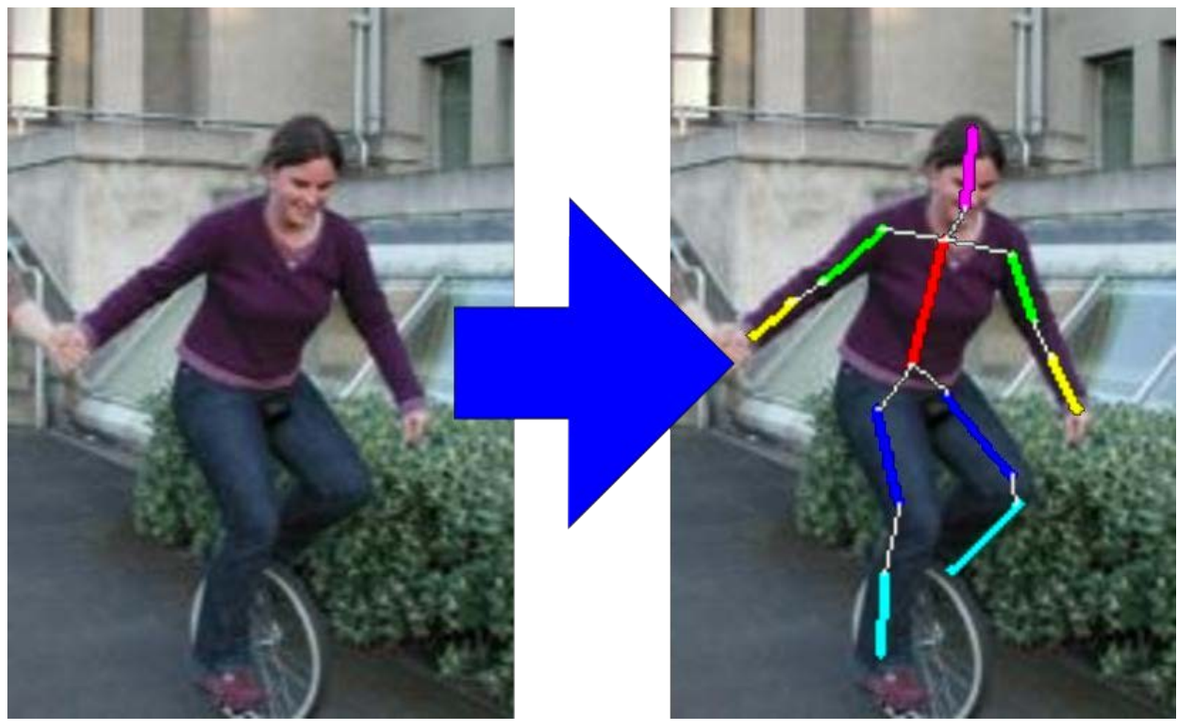
$$\arg\min \|p_1(x) - q\| + \|p_1(x) - p_2(x)\|$$

Content

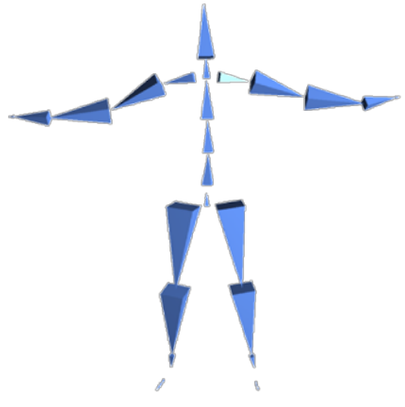
- Inverse Kinematics
- Kinematic Trees/Graphs
- **Pose Estimation/Motion Capture**

Pose Estimation

- Given an image depicting an articulated object, estimate the pose of that object



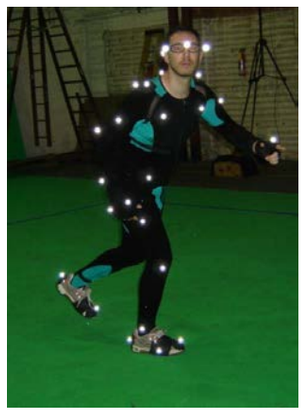
[Eichner 10]



determine all the DOF of that pose

Pose Estimation

- **Marker-based** pose estimation



- **Marker-less** pose estimation

[Faceshift]

- Pose estimation for multiple frames is called **motion capture**

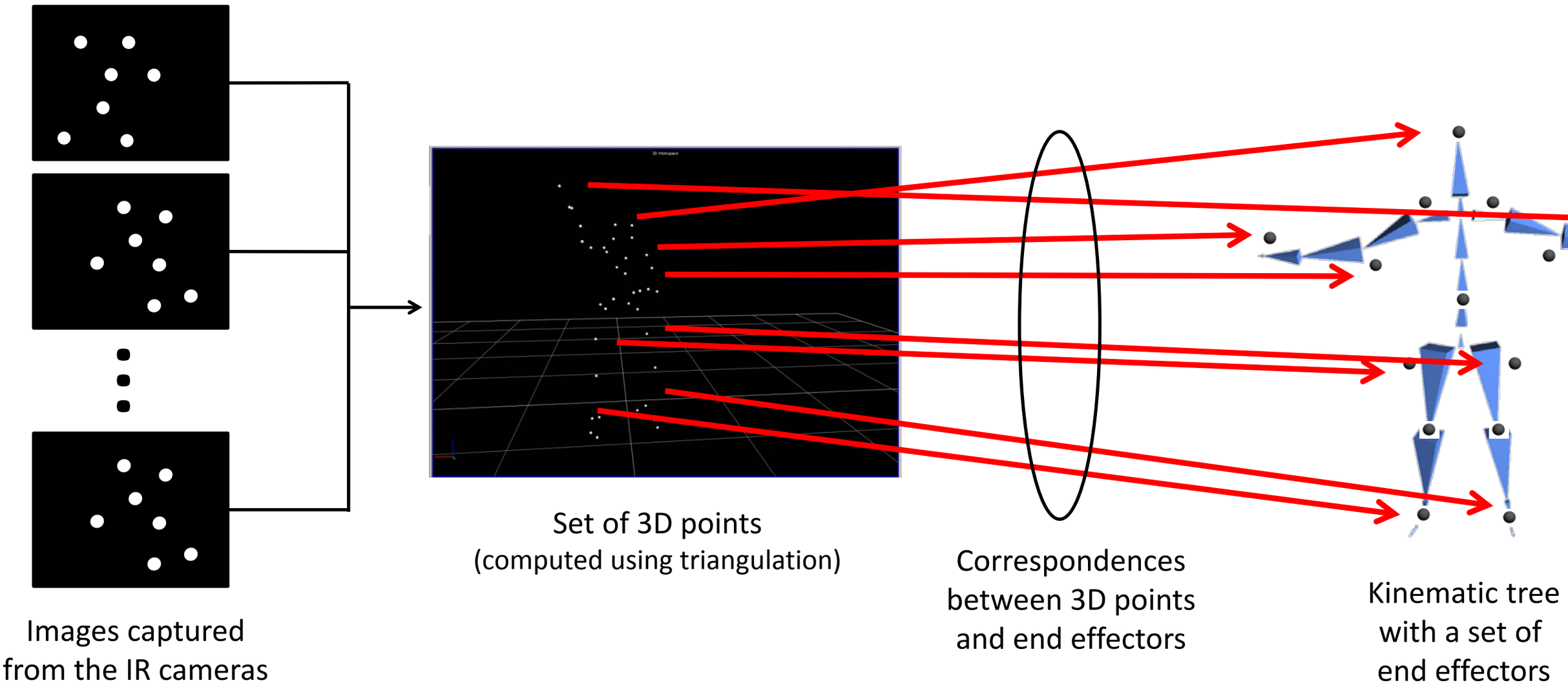
Marker-based Pose Estimation

Vicon system

Infrared cameras
with IR illuminator

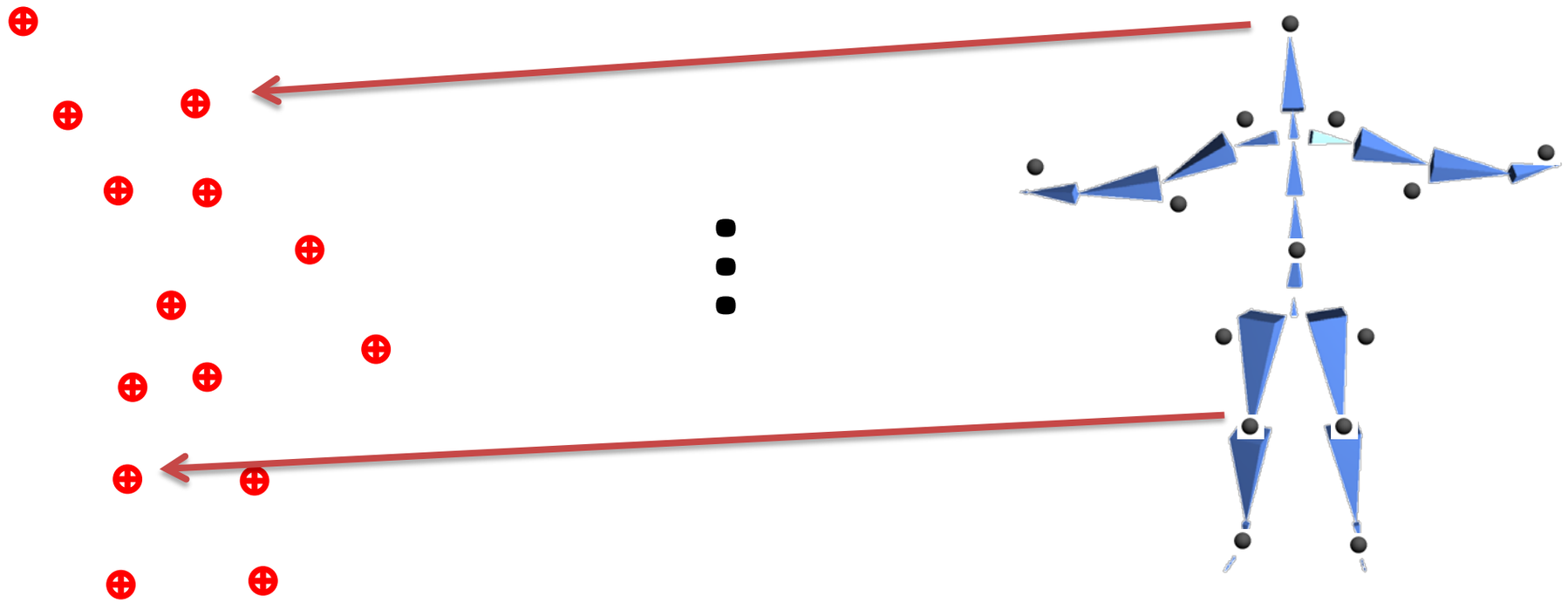
IR reflective markers

Marker-based Pose Estimation



- Triangulate 2D points -> 3D points
- Compute correspondences between 3D points and end effectors

Marker-based Pose Estimation



Targets

Kinematic tree with a set of end effectors

$$\arg\min_{x} \begin{cases} \|p_1(x) - q_1\| \\ \vdots \\ \|p_n(x) - q_n\| \end{cases}$$

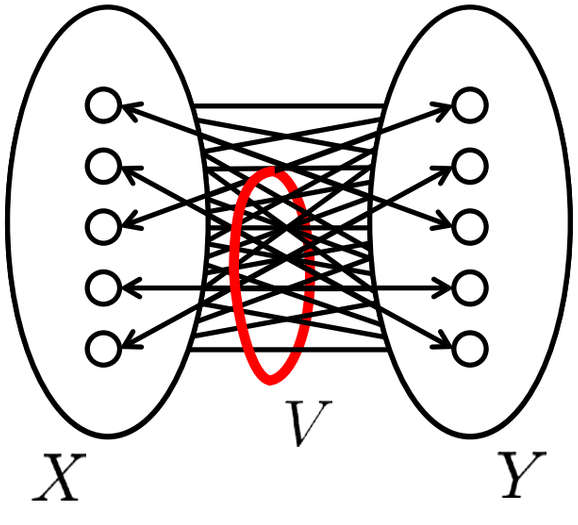
Multi-objective optimization

$$\arg\min_{x} \sum_{i=1}^{n} \|p_i(x) - q_i\|$$

Correspondences

- If correspondences between 3D points and targets are not known (or the heuristic used to compute them is not accurate), these need to be estimated together with the pose
- One of the possible solution to this problem is the ICP registration (ICP=Iterative Closest Points) [Besl & McKay 92]
- But how can we formulate mathematically this problem?

Correspondences



Complete bi-partite graph (with weights w_{ij})
(representing all the possible matching)

$$G = (X \cup Y, E)$$

Targets End effectors

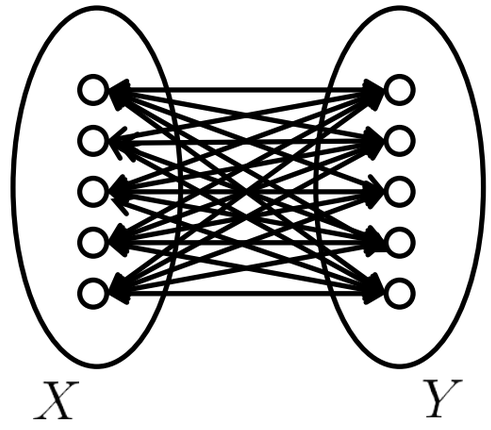
- All the edges on this graph are possible matching candidates
- We need to find the actual matches $V \subset E$ (unknown to the problem)

Correspondences

$e_{ij} = 1 \Leftrightarrow (i,j) \in V$ Target i in X is matched with end effector j in Y

w_{ij} cost of matching target i in X to end effector j in Y

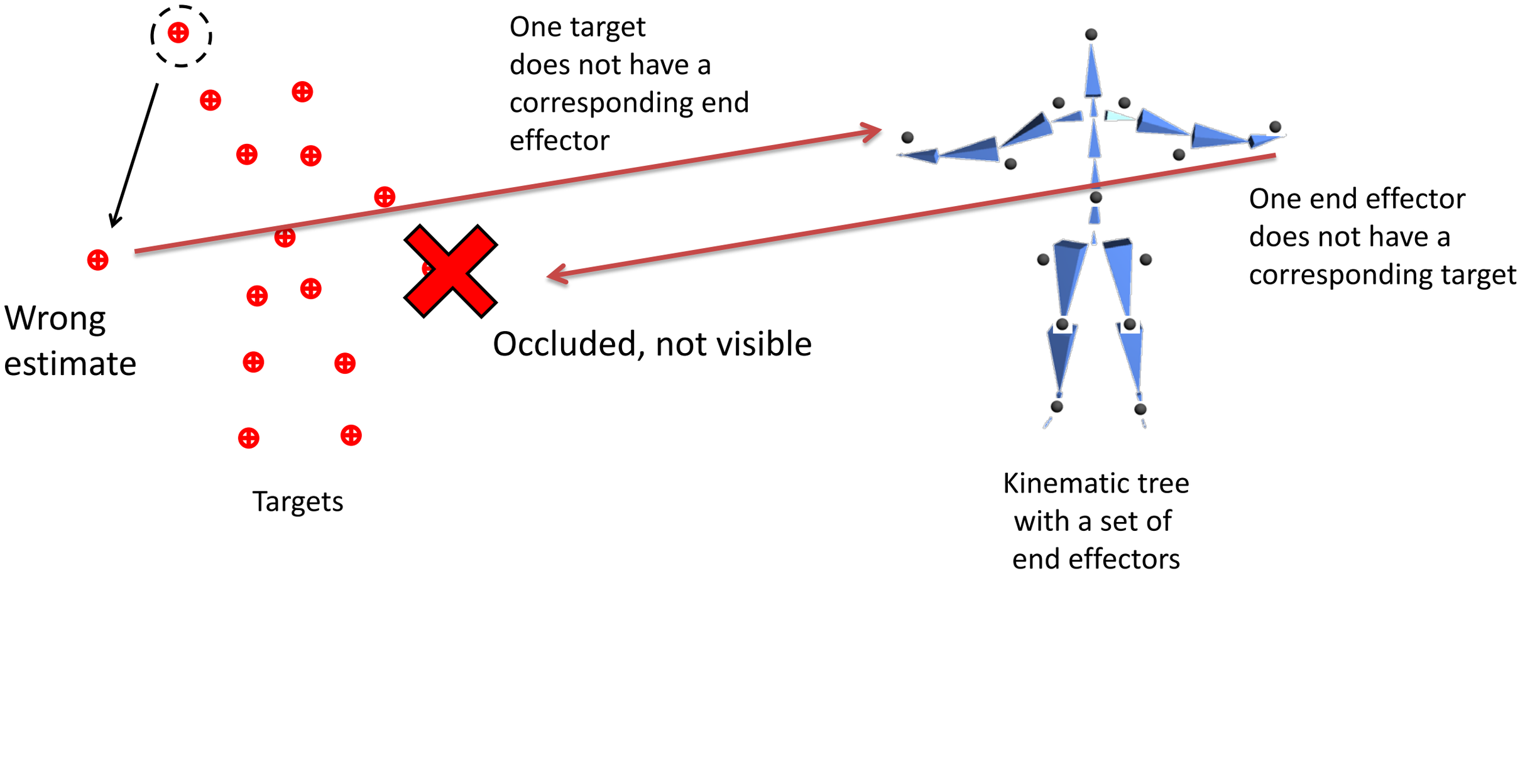
$$\begin{cases} \min\limits_{x,e_{ij}} & \sum_{ij} e_{ij} \|p_i(x) - q_j\| \quad +\lambda \sum_{ij} w_{ij} e_{ij} \\ \text{subject to} & \sum_i e_{ij} = 1 \quad \forall i \\ & \sum_j e_{ij} = 1 \quad \forall j \\ & e_{ij} \in \{0,1\} \quad \forall i,j \end{cases}$$



- Binary Integer problem + Continuous problem
- Alternating optimization approach
  - Integer Programming (Hungarian algorithm) + Gradient Descent

Correspondences

- It might happen that some targets are missing or that some targets are wrong

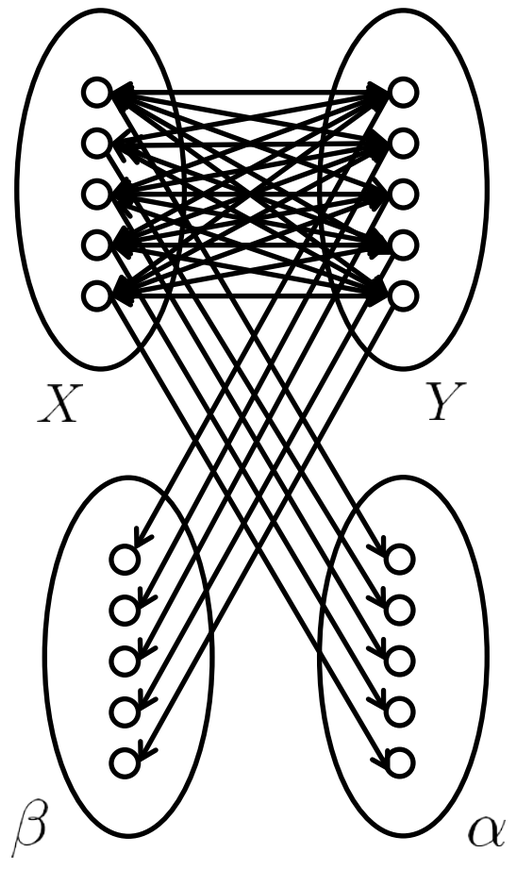


Correspondences

$$\begin{aligned}
\min_{x,e_{ij}} \quad & \textstyle\sum_{ij} e_{ij}\|p_i(x) - q_j\| \quad + \lambda \sum_{ij} w_{ij} e_{ij} + \lambda_\alpha \sum_i \alpha_i + \lambda_\beta \sum_j \beta_j \\
\text{subject to} \quad & \textstyle\sum_i e_{ij} + \alpha_i = 1 \qquad \forall i \\
& \textstyle\sum_j e_{ij} + \beta_j = 1 \qquad \forall j \\
& e_{ij}, \alpha_i, \beta_j \in \{0,1\} \qquad \forall i,j
\end{aligned}$$

additional unknowns

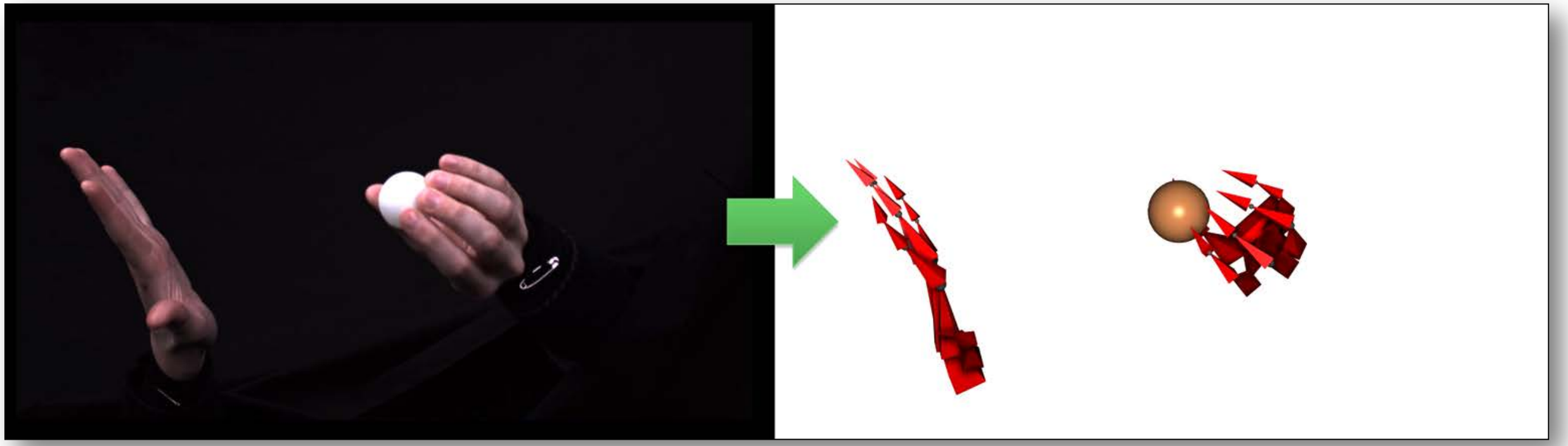
virtual nodes



Marker-Less Pose Estimation

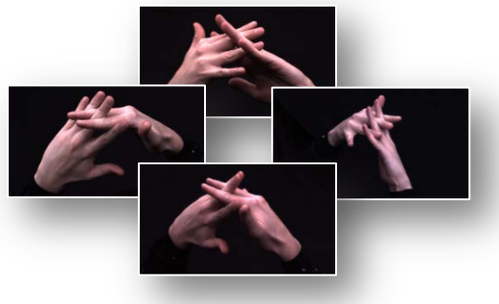
Input:

Output:

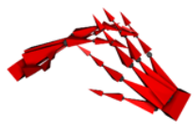


Scene Motion
(angles and positions)

Marker-Less Pose Estimation



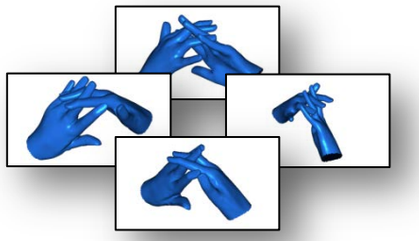
Input video frames
at time t



ξ

Pose

Generative
model



Rendered images

Comparison

Comparison Error

$$-\log\left[P\left(I=i|O=o\right)\right]$$

Correct
Pose

Minimum

$$\pi(i) = \arg\max_{o} P\left(I=i|O=o\right)$$

Maximum likelihood

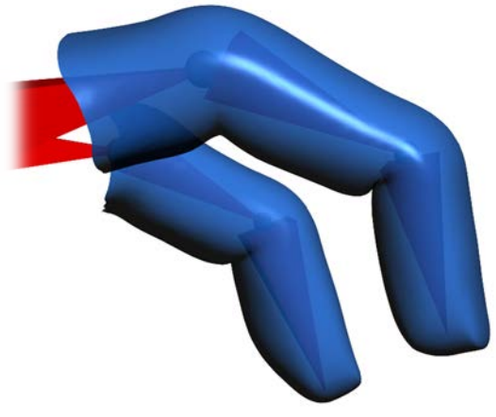
Generative Model

Kinematic tree + 3D model

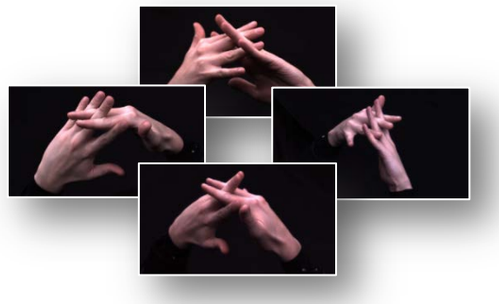
At a rigging pose

$$v_k(\boldsymbol{\xi}) = \sum_{j=1}^{m} \alpha_{k,j} T_j(\boldsymbol{\xi}) T_j(0)^{-1} v_k(0)$$

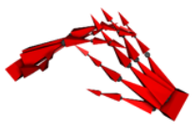
the motion of a vertex → the linear combination of all the motions that the vertex would undergo if rigidly attached to every bone, one at a time



Marker-Less Pose Estimation



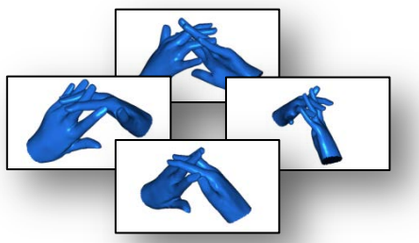
Input video frames
at time t



ξ

Pose

Generative
model



Rendered images

Comparison

Comparison Error

$$-\log\left[P\left(I=i|O=o\right)\right]$$

Correct
Pose

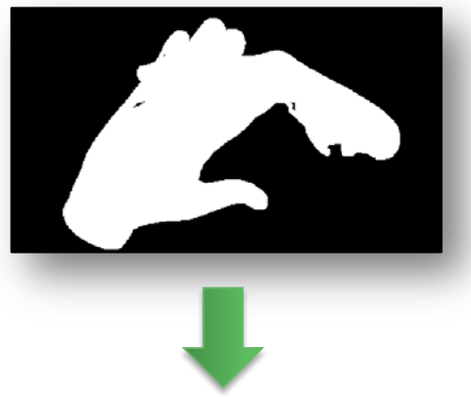
Minimum

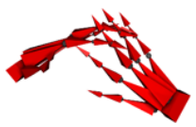
$$\pi(i)=\arg\max_{o} P\left(I=i|O=o\right)$$

Maximum likelihood

Comparison

Input video frame
at time t

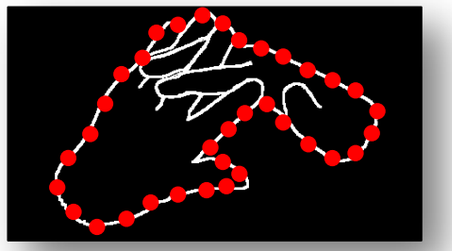


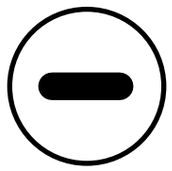


ξ

Pose

Generative
model





$$\sum_k d(v_k(\boldsymbol{\xi}), S)$$

$d(\cdot, S)$ = is a distance map in 3D between a point and S

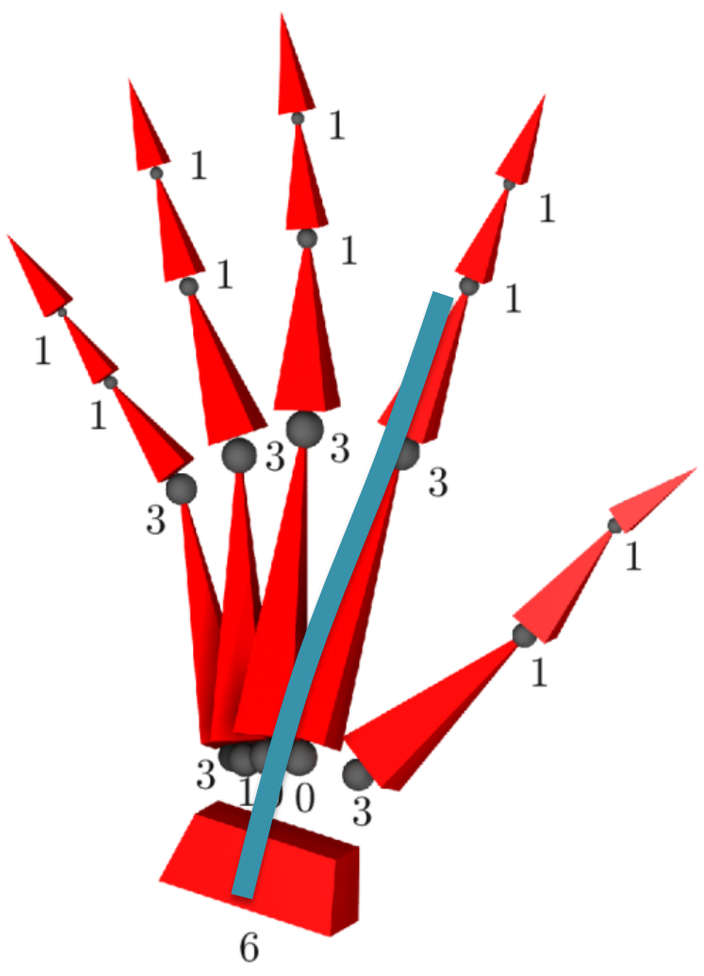
$$v_k(\boldsymbol{\xi}) = \sum_{j=1}^{m} \alpha_{k,j} T_j(\boldsymbol{\xi}) T_j(0)^{-1} v_k(0)$$

Solution

$$L(\boldsymbol{\xi}) = \sum_k d(v_k(\boldsymbol{\xi}), S)$$

$d(\cdot, S)$ = is a distance map in 3D

$$\frac{\partial L}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) = \sum_k \nabla d(v_k(\boldsymbol{\xi}), S) \cdot \frac{\partial v_k}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi})$$



$$v_k(\boldsymbol{\xi}) = \sum_{j=1}^{m} \alpha_{k,j} T_j(\boldsymbol{\xi}) T_j(0)^{-1} v_k(0)$$

$$\frac{\partial v_k}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) = \sum_{j=1}^{m} \alpha_{k,j} \frac{\partial T_j}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) T_j(0)^{-1} v_k(0)$$

$$T_j(\boldsymbol{\xi}) = e^{\xi_{c_1}} \ldots e^{\xi_{c_q}}$$

$$\frac{\partial T_j}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}) = \cdots$$

Final Examination

- Homework: 10% each
- Oral Exam: 60%
  - (60%-40%) about **40min** per person (20 Luca, 20 J.C.)
  - (100%) about **1h** per person (30 Luca, 30 J.C.)

- each person has to answer different questions/exercises spanning the **entire program**

- ETH will schedule the exam sometime in August

- For the ones
  - who didn't do the homework
  - who did not do **all** the homework
  - who is not satisfied with the homework grades and want to improve it

} he/she can do the 100% exam to recover the missing points (inform in advance)

Mathematical Foundations of Computer Graphics and Vision

Rigid Transformations

--- the geometry of SO(3) & SE(3) ---

Luca Ballan

Motivation

$$x^* = \arg\min_{x \in \mathbb{R}^k} L(x)$$

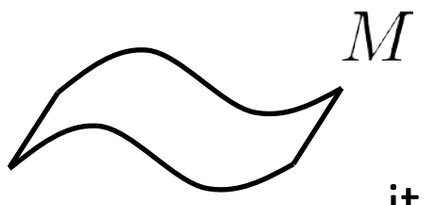
(**unconstrained minimization problem**)

$$u^* = \arg\min_{u \in \mathbb{F}(\mathbb{R}^k, \mathbb{R}^m)} L(u)$$

(**unconstrained minimization problem** with functions as domain)

$$x^* = \arg\min_{x \in M} L(x)$$

(**constrained minimization problem**)



it is very thin!!

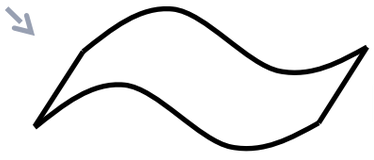
Motivation

- Many problems are formulated in the domain of a manifold
- Some in particular refers to **the set of the rigid motions** $SE(3)$

$$\begin{cases} \min f(x, \mathcal{O}) \\ x \in SE(3) \end{cases}$$

$\mathcal{O}$ = observations

optimization in the domain of a manifold

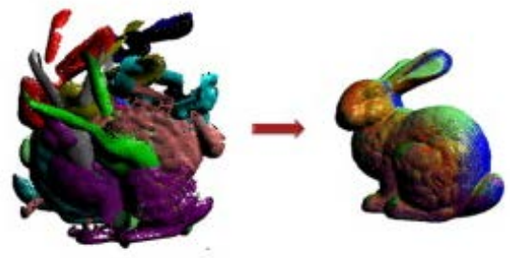


Manifold
(adding noise to a rotation matrix gives as result a non rotation matrix)

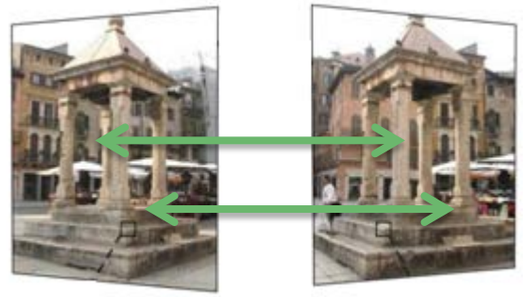
- **Reference book:** R. Murray, Z. Li and S. Sastry, “A Mathematical Introduction to Robotic Manipulation”, CRC Press 1994

Motivation

- Rigid Registration



- Camera pose estimation
  - **Input:** two images (with known intrinsics)
    - Compute correspondences between these images
    - Estimate the essential matrix $\mathbf{p}_i{}'^{\top} E \mathbf{p}_i = 0$
    - Factorize E in (R,t)
    - Compute the 3D structure
    - Bundle-Adjustment



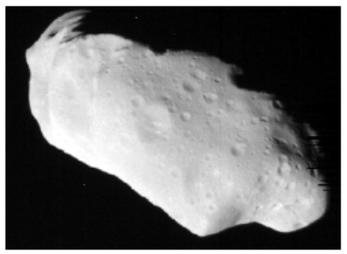
$$\min_{R\ ,\mathbf{t}\ ,\mathbf{M}^j} \sum_{j=1}^{n} d(K\ [R\ |\mathbf{t}\ ]\mathbf{M}^j, \mathbf{m}^j)^2$$

Motivation

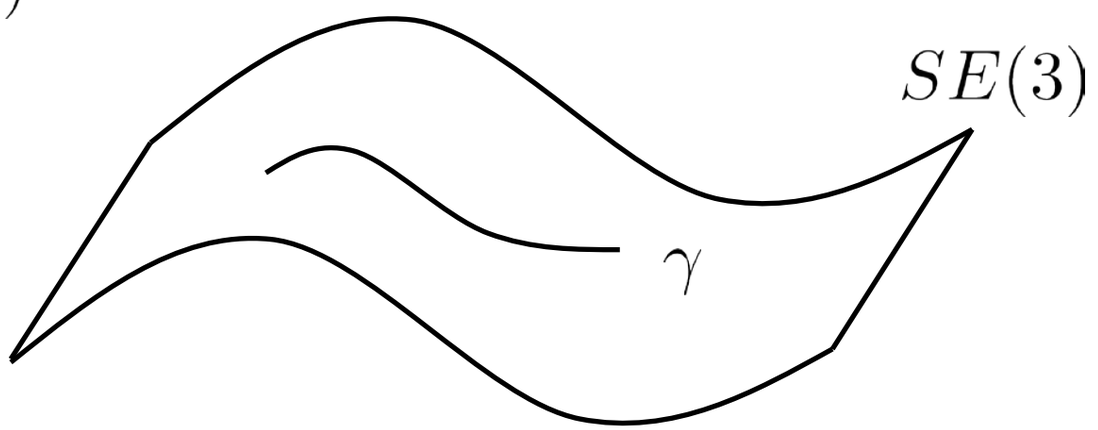
- The trajectory of a rigid object

$$\gamma : \mathbb{R} \to SE(3)$$ is a (smooth) curve in $SE(3)$

- 3D Rigid Object or Camera Tracking

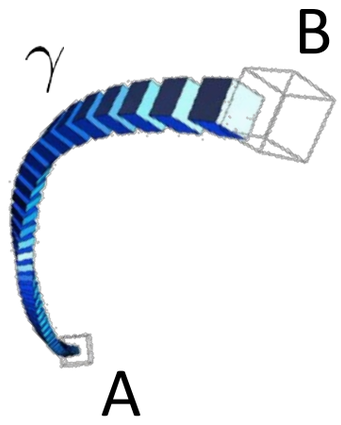


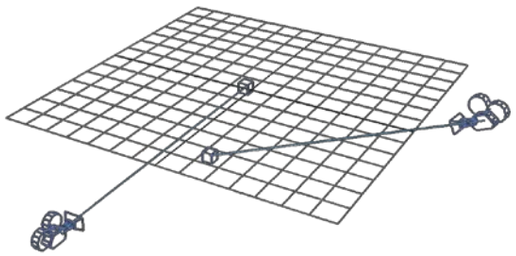
$$\begin{cases} \min f(\gamma, \mathcal{O}) \\ \quad \gamma : \mathbb{R} \to SE(3) \end{cases}$$

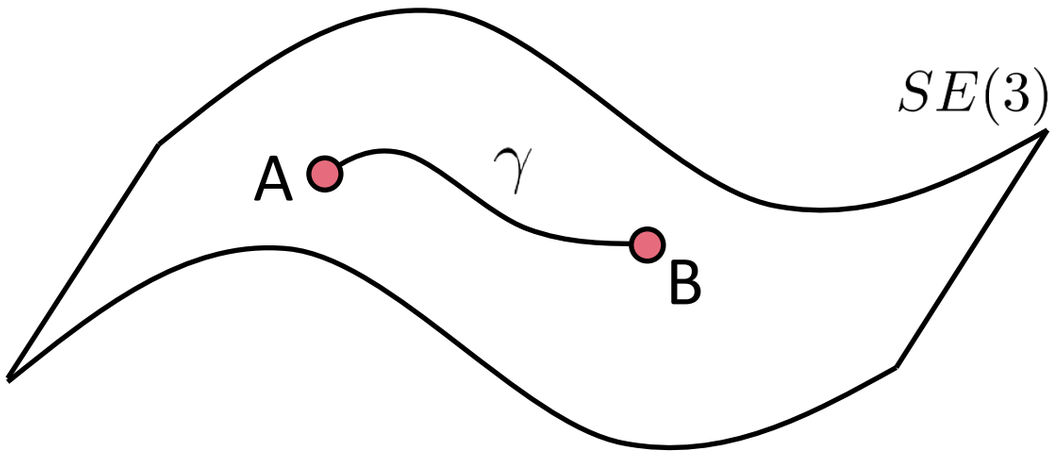


Motivation

- Rigid Motion Interpolation
  - Given two rigid motions: A and B $\in SE(3)$







- Find a smooth rigid motion γ connecting A and B
  (or find the shortest path between A and B)

Content

- **Rigid transformations**
- Linear Matrix Groups
- Manifolds
- Lie Groups/Lie Algebras
- Charts on SO(2) and SO(3)

Rigid Transformations

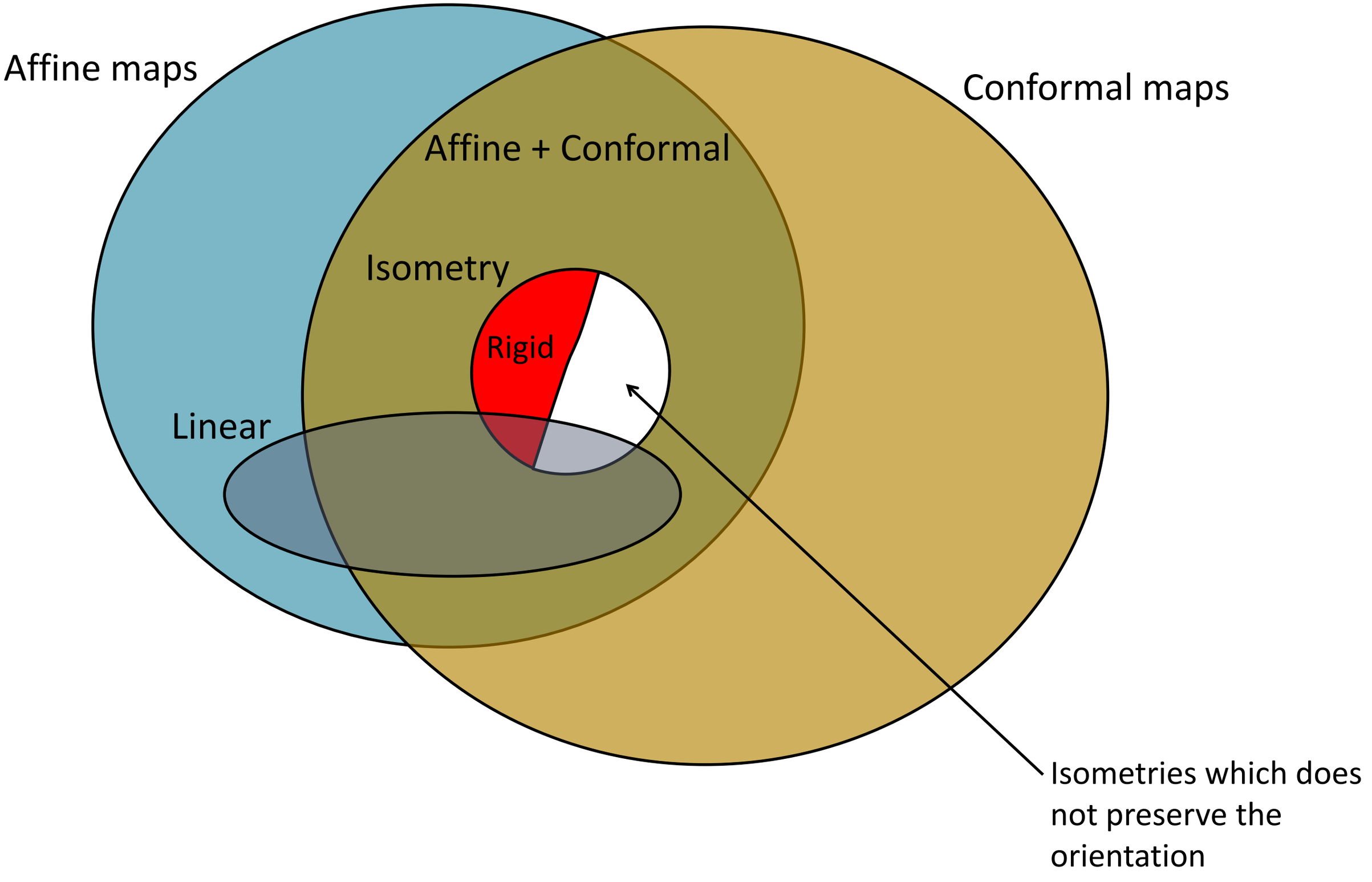
$F : A \to A$ is a transformation

Rigid Transformations

$F : A \to A$ is a rigid transformation iff,

- it preserves distances $d(x, y) = d(F(x), F(y)), \quad \forall x, y \in A$ (isometry)
- it preserves the space orientation (no reflection)

Taxonomy


Affine maps
Conformal maps
Affine + Conformal
Isometry
Rigid
Linear
Isometries which does not preserve the orientation

Representation

if A is a finite dimensional space (e.g. $\mathbb{R}^n$)

a rigid transformation $F : \mathbb{R}^n \to \mathbb{R}^n$

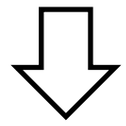
can be written as

$$F(x) = Rx + t \qquad x \in \mathbb{R}^n$$

$$t \in \mathbb{R}^n$$

$$R \in \mathbb{R}^{n \times n}$$

- R orthogonal (isometry)
- $det(R) = 1$ (preserve orientation)



Rotation matrix

$$F(x) = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} x \qquad x \in \mathbb{RP}^n$$

Projective space

Note: in this space, F is also linear

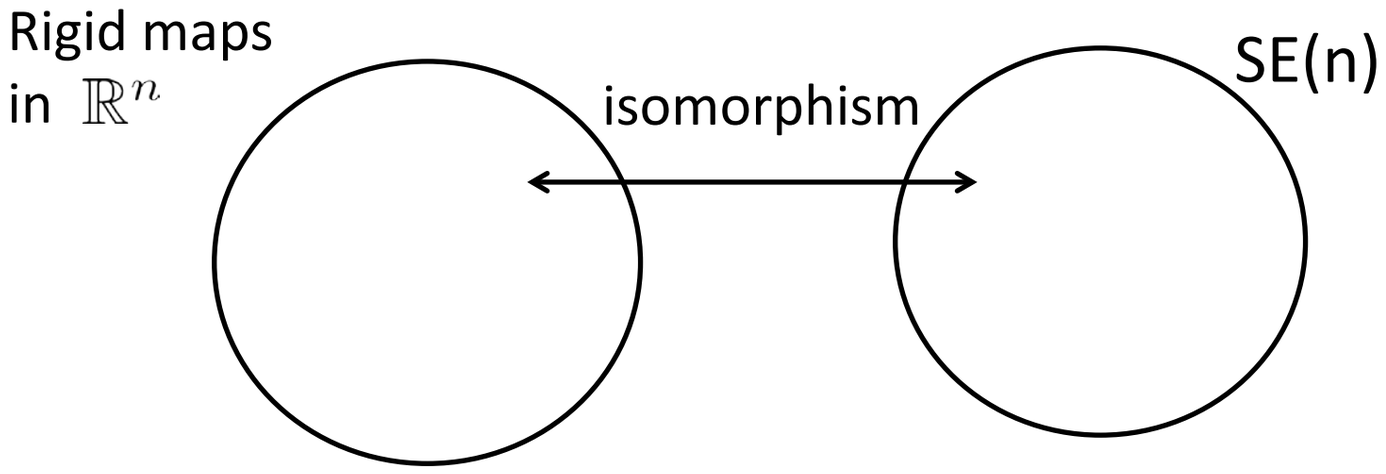
Rigid Transformations

- The set of all the rigid transformations in $\mathbb{R}^n$ is a **group** (not commutative) with the composition operation

$$(\{F : \mathbb{R}^n \to \mathbb{R}^n \mid F \text{ rigid } \}, \circ)$$

*

- This set is isomorphic to the **special Euclidean group SE(n)**



*

- The existence of an isomorphism is important because one can represent each rigid transformation as an element of SE(n) (bijective) and performs operations in this latter space (which will correspond to operations in the former space)

Content

- Rigid transformations
- **Matrix Groups**
- Manifolds
- Lie Groups/Lie Algebras
- Charts on SO(2) and SO(3)

Matrix Groups

- The set of all the nxn invertible matrices is a group w.r.t. the matrix multiplication

$$GL(n) = (\{M \in \mathbb{R}^{nxn} \mid det(M) \neq 0\}, \times)$$ **General linear group**

- GL(n) is isomorphic to the group of **linear and invertible transformations** in $\mathbb{R}^n$ with the composition as operation

$$(\{F : \mathbb{R}^n \to \mathbb{R}^n \mid F \text{ linear bijective}\}, \circ)$$

- It exists an isomorphism $\Psi(x \to Mx) = M$, such that

$$\Psi(F \circ G) = \Psi(F) \times \Psi(G)$$

*

Matrix Groups

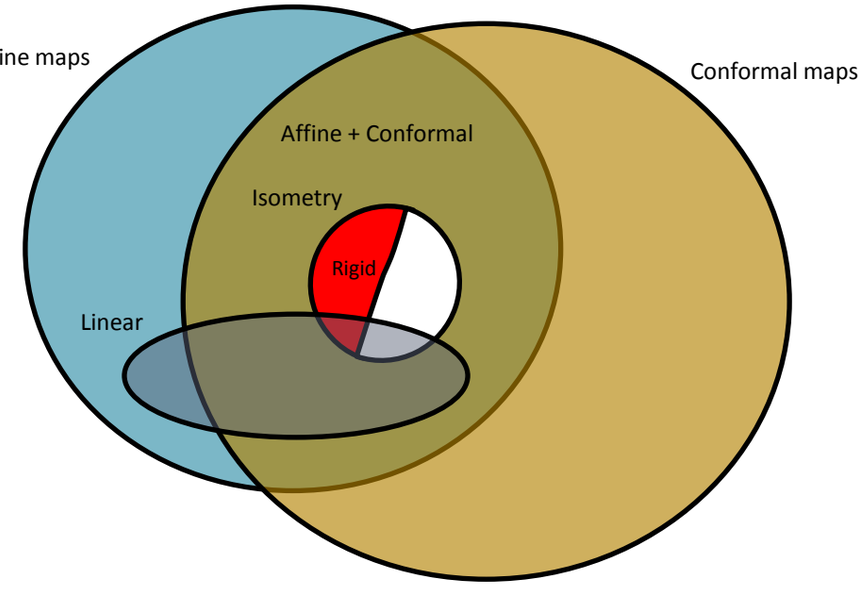
- The set of all the nxn orthogonal matrices is a group w.r.t. the matrix multiplication

$$O(n) = \left(\left\{A \in GL(n) \mid A^{-1} = A^T\right\}, \times\right)$$ **Orthogonal group**

- O(n) is isomorphic to the group of **linear isometries** in $\mathbb{R}^n$ with the composition as operation

$$(\{F : \mathbb{R}^n \to \mathbb{R}^n \mid F \text{ linear isometry}\}, \circ)$$

- PS: $A \in O(n) \Rightarrow \det(A) = \pm 1$



Matrix Groups

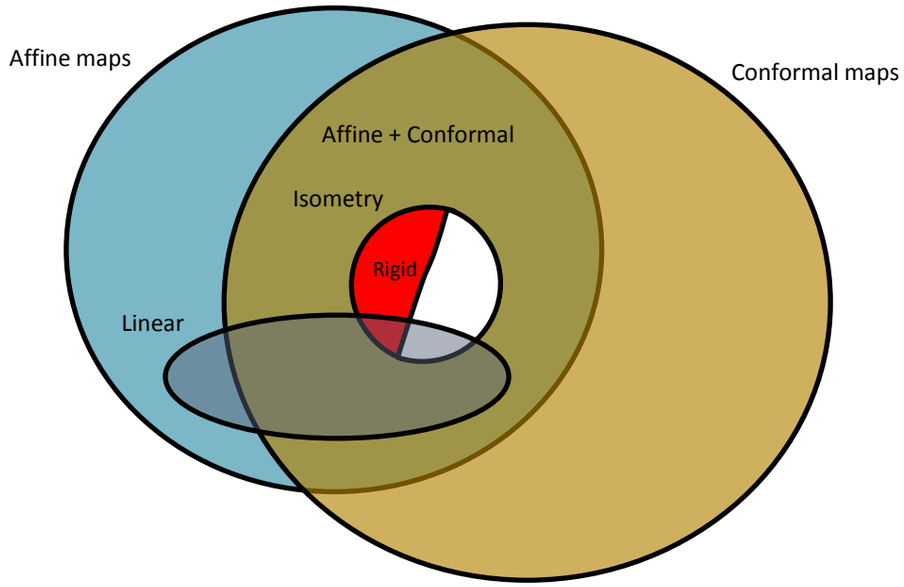
- The set of all the nxn orthogonal matrices with determinant equal to 1 is a group w.r.t. the matrix multiplication

$$SO(n) = (\{A \in O(n) \mid det(O) = +1\}, \times)$$ **Special orthogonal group**

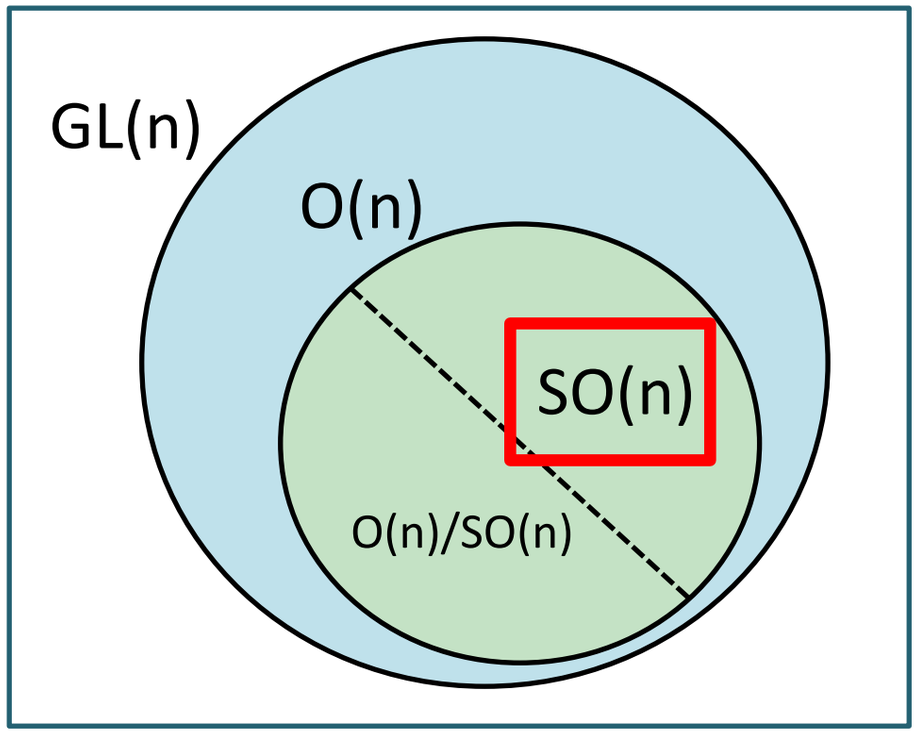
- SO(n) is isomorphic to the group of **linear rigid transformations** in $\mathbb{R}^n$ with the composition as operation

- It exists an isomorphism $\Psi(x \to Mx) = M$, such that

$$\Psi(F \circ G) = \Psi(F) \times \Psi(G)$$



Groups of Matrices: Summary



$\mathbb{R}^{nxn}$ = vector space of all the nxn matrices

$$GL(n) = (\{M \in \mathbb{R}^{nxn} \mid det(M) \neq 0\}, \times)$$ General linear group of order n

$$O(n) = (\{A \in GL(n) \mid A^{-1} = A^T\}, \times)$$ Orthogonal group of order n

$$SO(n) = (\{A \in O(n) \mid det(O) = +1\}, \times)$$ Special orthogonal group of order n

$$O(n)/SO(n) = \{A \in O(n) \mid det(O) = -1\}$$ Set of orthogonal matrices which do not preserve orientation (not a group)

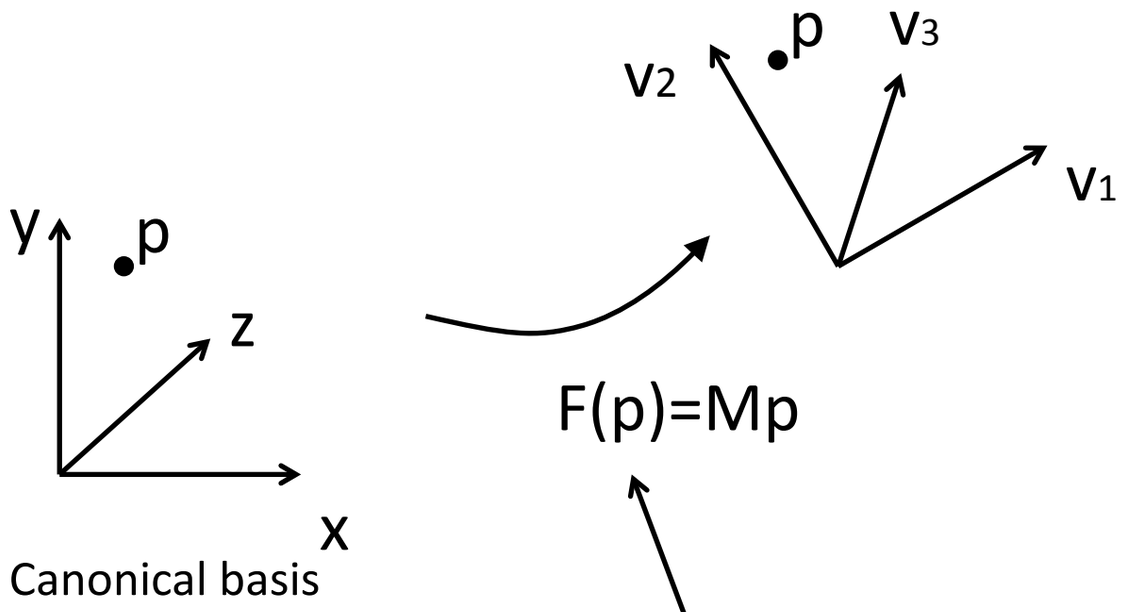
SO(n) in practice

$$M \in SO(3) \quad \Rightarrow \quad M = \begin{bmatrix} . & . & . \\ v_1 & v_2 & v_3 \\ . & . & . \end{bmatrix}$$

Orthogonality:

$$< v_i, v_j >= 0$$

$$|v_i| = 1$$



Special Euclidean group

- The Cartesian product $SO(n) \times \mathbb{R}^n$ is a group w.r.t. a "weird" operation

$$SE(n) = (SO(n) \times \mathbb{R}^n, \times)$$

Special Euclidean group

$$(M,t) \times (S,q) = (MS, Mq + t)$$

- The "weird" operation is define in such a way that the group SE(n) is isomorphic to the group of **rigid transformations** in $\mathbb{R}^n$ with the composition as operation

- It exists an isomorphism $\Psi(x \to Rx + t) = (R,t)$, such that

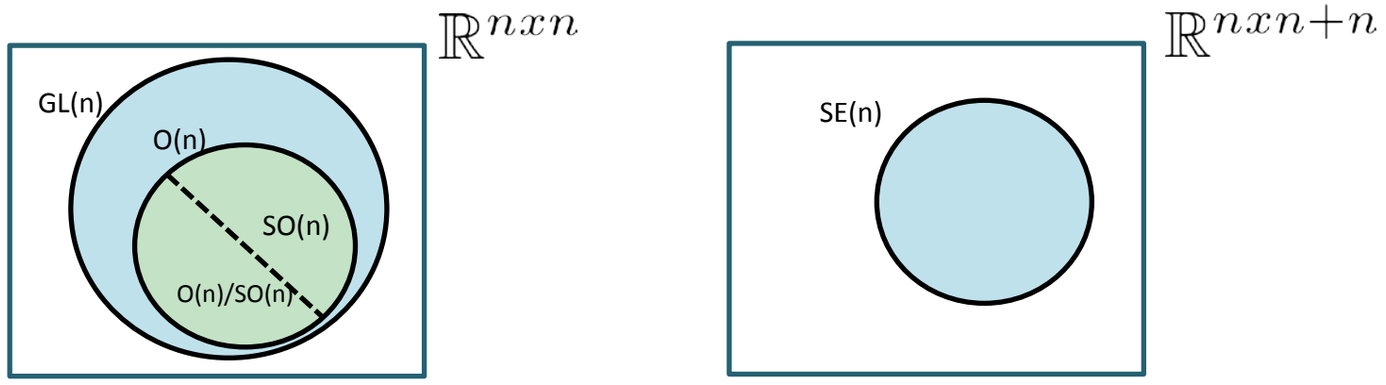
$$\Psi(F \circ G) = \Psi(F) \times \Psi(G)$$

$$\begin{aligned} F(x) &= Mx + t \\ G(x) &= Sx + q \end{aligned} \quad \Rightarrow \quad \Psi(F \circ G) = (M,t) \times (S,q)$$

Commutative??

The Geometry of these Groups

- GL(n), O(n), SO(n) and SE(n) are all subset of a vector space

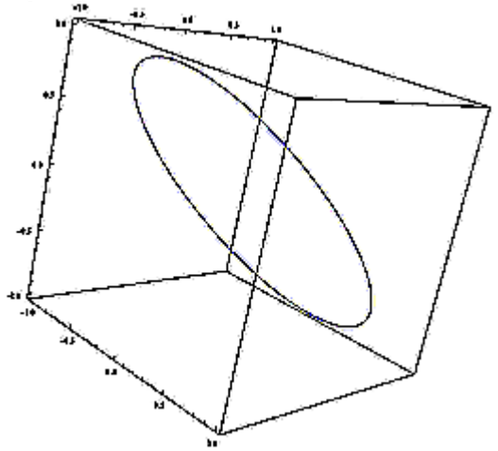


- GL(n), O(n), SO(n) and SE(n) are all **smooth manifolds**
  (surfaces, curves, solids, etc... immerse in some big vector space)

SO(2) and SO(3): Shape

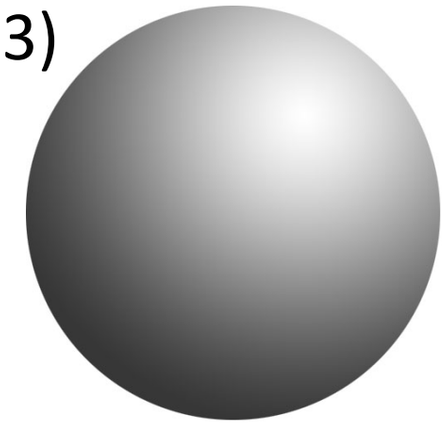
- What are the shapes of these two manifolds?

SO(2)



1-manifold

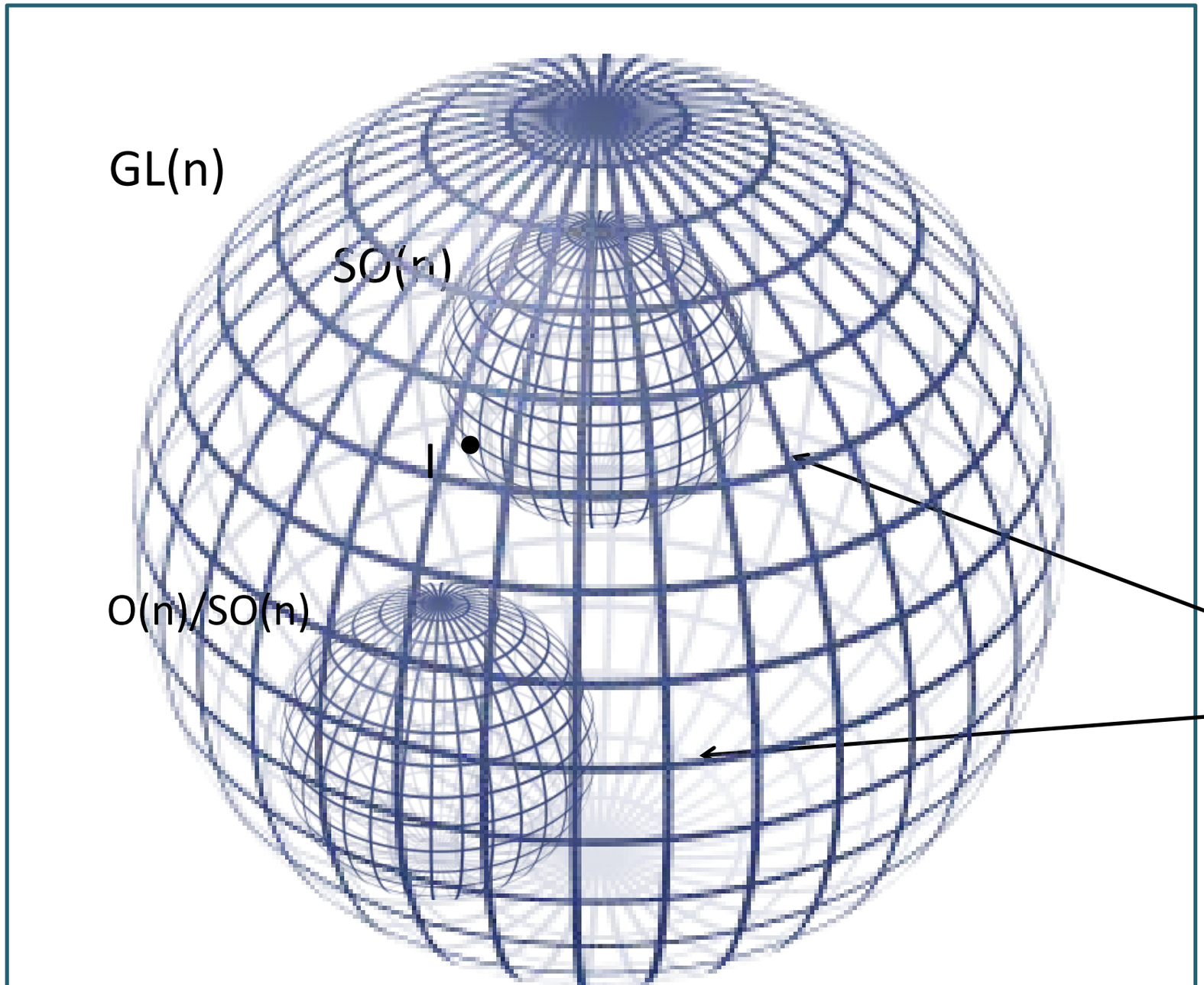
SO(3)



3-manifold

*

GL(N), O(N) and SO(N)



$\mathbb{R}^{nxn}$

O(n) is the union of these two manifolds

Content

- Rigid transformations
- Matrix Groups
- **Manifolds**
- Lie Groups/Lie Algebras
- Charts on SO(2) and SO(3)

Manifold

- The concept of manifold generalizes
  - the concepts of **curve**, **area**, **surface**, and **volume** in the Euclidean space/plane
  - … but not only …
- A manifold does not have to be a subset of a bigger space, it is an object on its own.
- A manifold is one of the most generic objects in math..
- Almost everything is a manifold

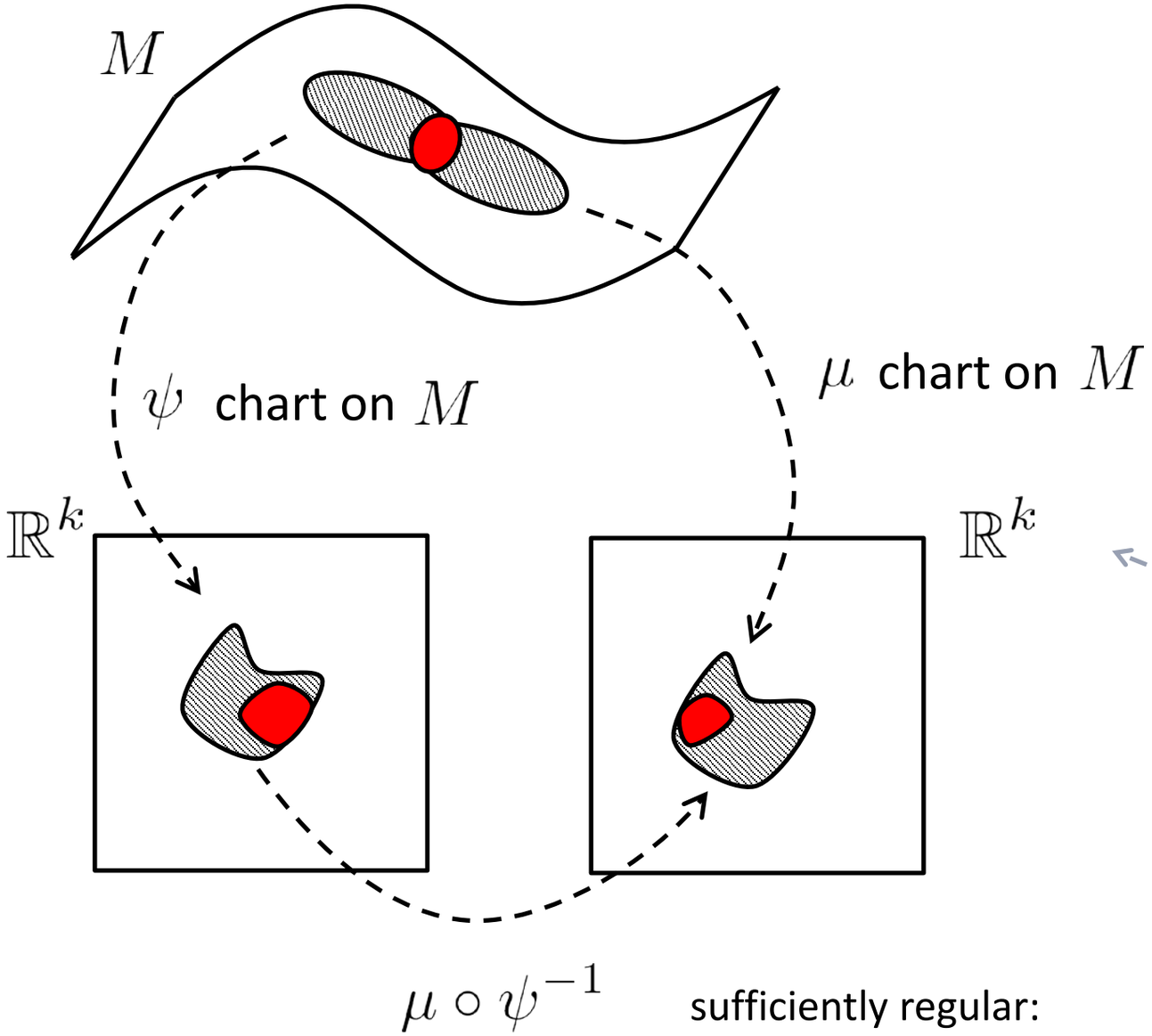
Differential Manifold

- **Manifold =** topological set + a set of charts

$$M = (S, \mathcal{T}, \mathcal{A})$$

topological set

Atlas = set of charts

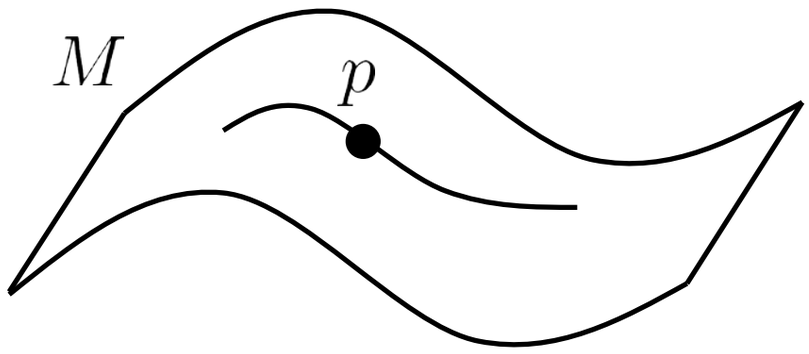


M is a **k**-manifold

sufficiently regular:
Bijective, derivable s times
with inverse derivable s times

Chart: bijective, continuous,
and with continuous inverse

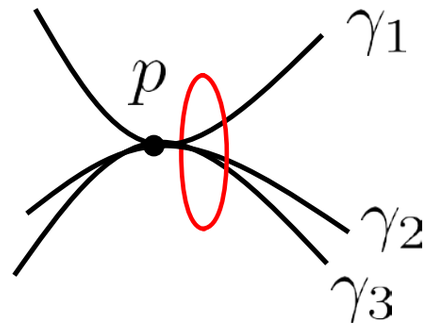
Tangent Space



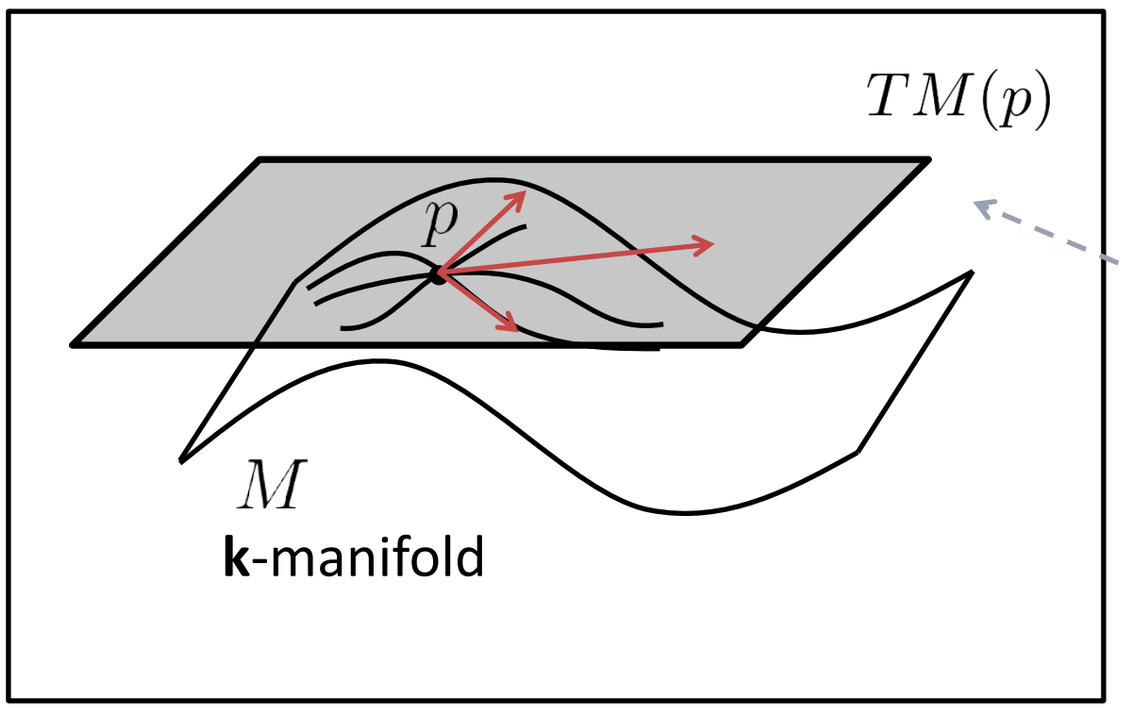
- The **tangent space of** M **in** p is the set of all the smooth curves in M of type

$$\begin{cases} \gamma : \mathbb{R} \to M \\ \gamma \in C^0 \\ \gamma(0) = p \end{cases}$$

- grouped accordingly to their first derivative in p

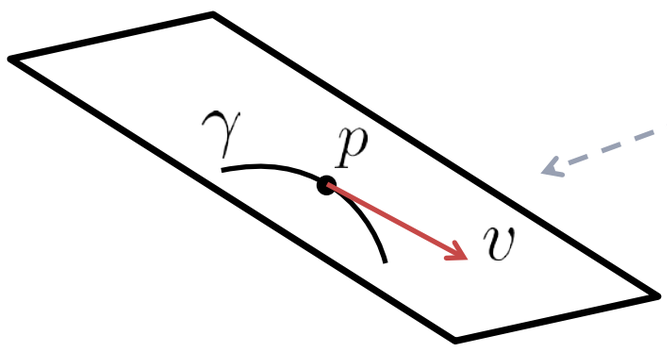


Tangent Space



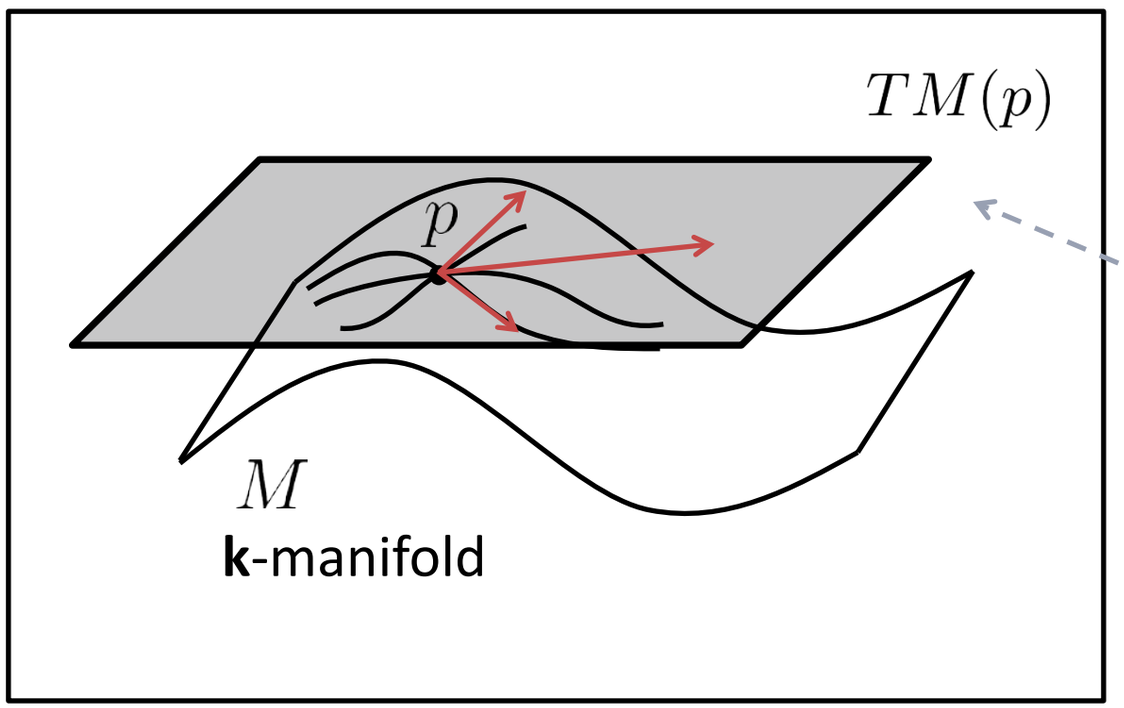
V = Vector space

The tangent space of M in p is isomorphic to a subspace of V



It corresponds to the velocity of γ in p
(direction and speed)

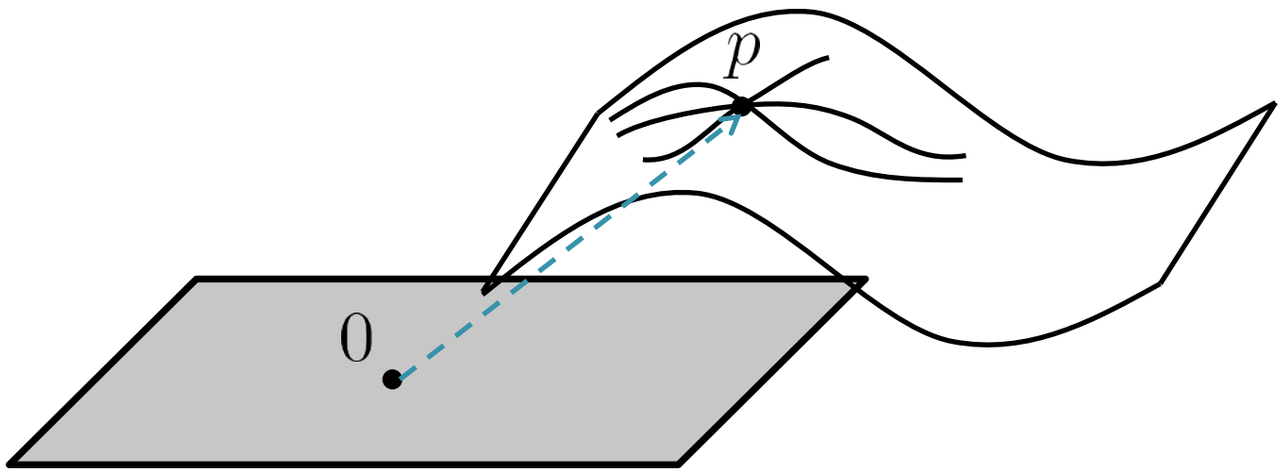
Tangent Space



V = Vector space

The tangent space of M in p is isomorphic to a subspace of V

- $TM(p)$ is a vector space (subspace of V)
  has dimension k



1-manifold → 1 dim TM
(curves) (lines)

2-manifold → 2 dim TM
(surfaces) (planes)

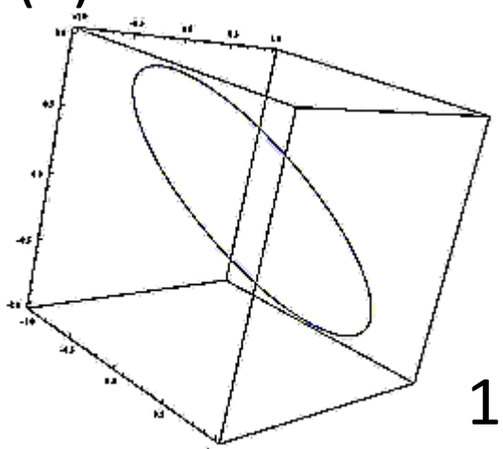
3-manifold → 3 dim TM
(volumes) (full volumes)

SO(2) and SO(3): Tangent Spaces

- What are the tangent spaces of these two manifolds?

$\mathbb{R}^{2x2}$

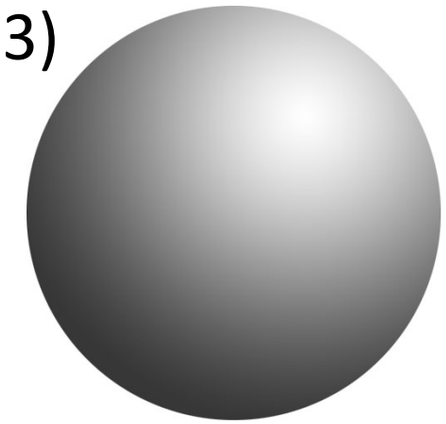
SO(2)



1-manifold

$\mathbb{R}^{3x3}$

SO(3)



3-manifold

$T_{SO(2)}$ vector space with
1 dimension
subspace of $\mathbb{R}^{2x2}$

$T_{SO(3)}$ vector space with
3 dimensions
subspace of $\mathbb{R}^{3x3}$

They are matrices

Skew-Symmetric Matrix

M is skew-symmetric matrix iff $M^T = -M$

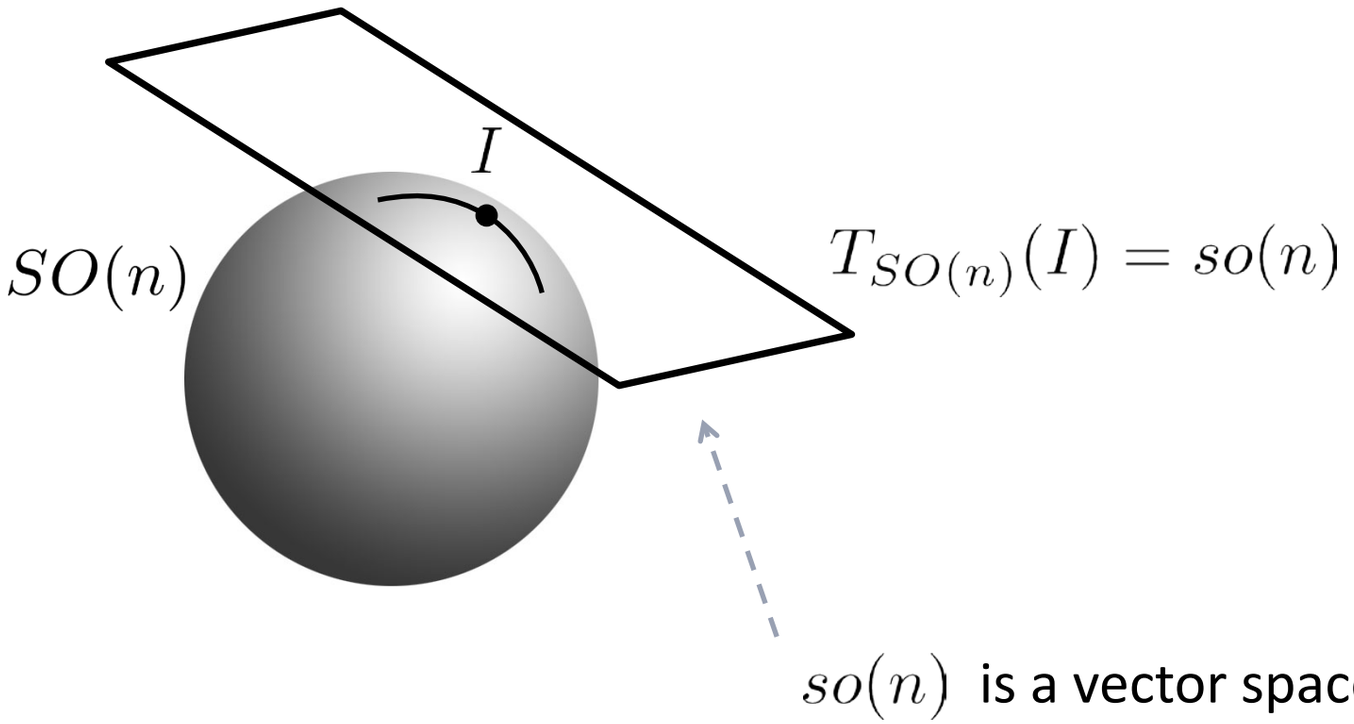
$$\begin{bmatrix} 0 & 3 & 6 \\ -3 & 0 & -1 \\ -6 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 4 \\ -4 & 0 \end{bmatrix}$$

$$so\,(n) = (\left\{M \in \mathbb{R}^{nxn} \mid M^T = -M\right\}, +, \cdot_e, [\,])$$

Special orthogonal Lie algebra
(vector space with Lie brackets)

$$[A, B] = AB - BA$$

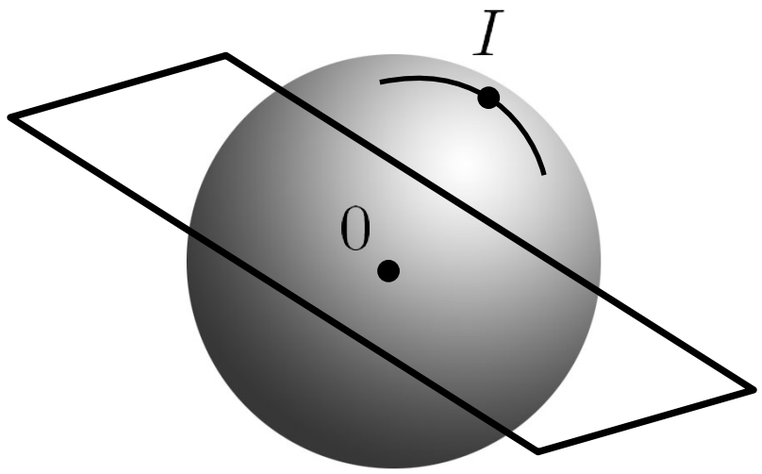
Skew-Symmetric Matrix



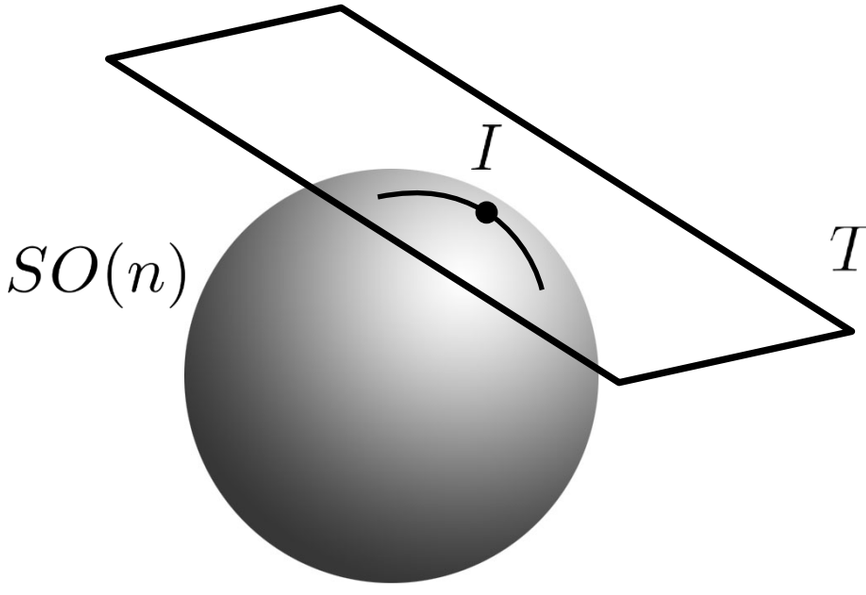
The Special orthogonal Lie algebra is the tangent space of $SO(n)$ at the identity

$so(n)$ is a vector space so it passes through the null matrix

so in reality



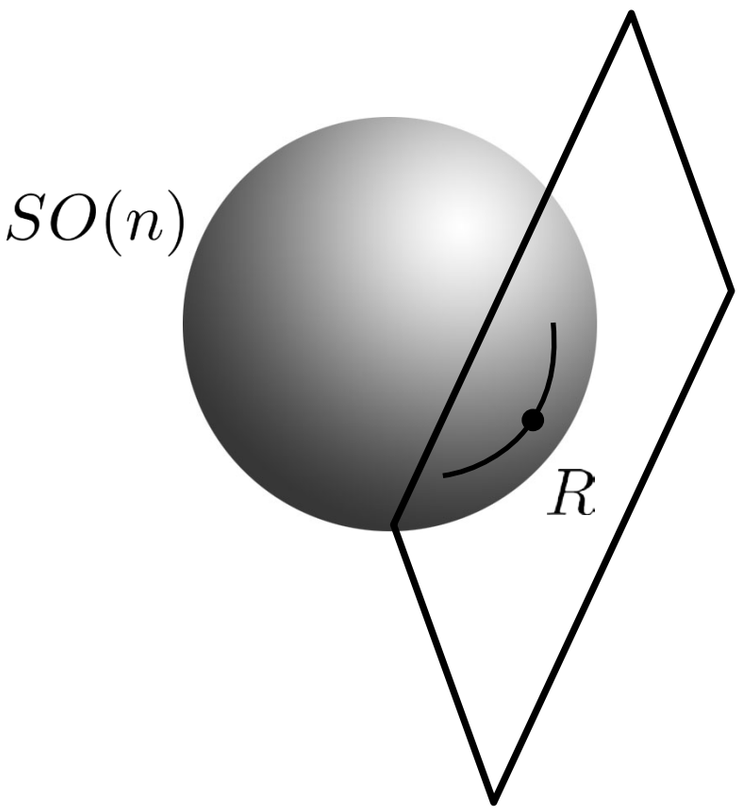
Skew-Symmetric Matrix



$$T_{SO(n)}(I) = so(n)$$

The Special orthogonal Lie algebra is the tangent space of $SO(n)$ at the identity

valid only at the identity



$$T_{SO(n)}(R) = R \times so(n)$$

The tangent space of $SO(n)$ in any other point R is a rotated version of $so(n)$

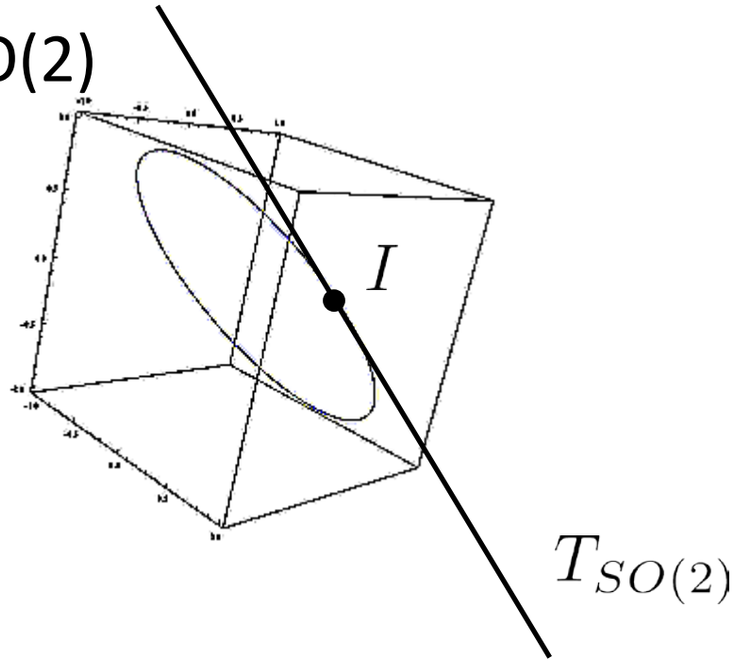
they are no more skew-symmetric matrices but rotations of them

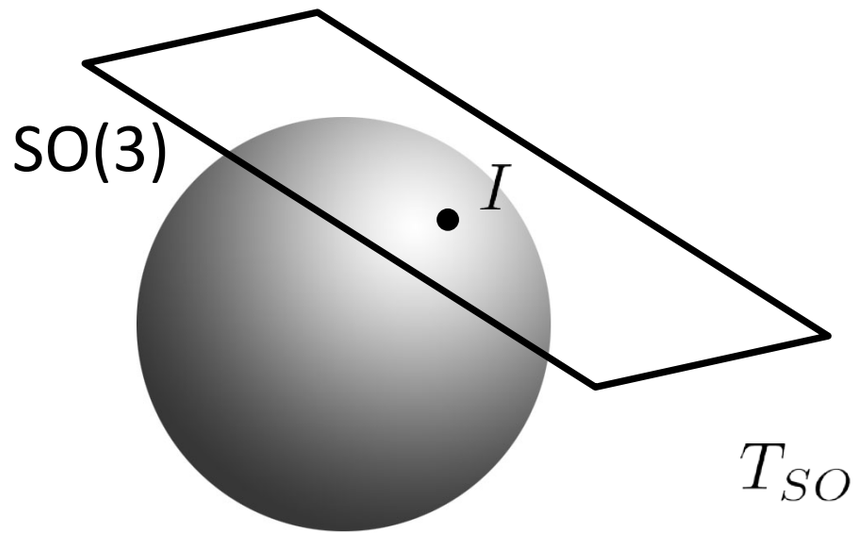
so(2) and so(3)

$$\begin{bmatrix} 0 & 3 & 6 \\ -3 & 0 & -1 \\ -6 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 4 \\ -4 & 0 \end{bmatrix}$$

- $so(3)$ is a vector space of dimension 3
- $so(2)$ is a vector space of dimension 1

SO(2)





$T_{SO(3)}$ is a volume

an element in so(3) or so(2) represents an infinitesimal rotation from the identity matrix

The hat operator

- The hat operator in $so(3)$

$$\widehat{\cdot}: \mathbb{R}^3 \to so(3)$$

$$\widehat{(x,y,z)} \to \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

- it is an isomorphism from $so(3)$ to $\mathbb{R}^3$
  (it maps + into +)

- The hat operator in $so(2)$

$$\widehat{\cdot}: \mathbb{R} \to so(2)$$

$$\widehat{x} \to \begin{bmatrix} 0 & -x \\ x & 0 \end{bmatrix}$$

The hat operator

- The hat operator is used to define cross-product in matrix form:

$$a \times b = \widehat{a}b \qquad \forall a, b \in \mathbb{R}^3$$

- The hat operator maps cross products into [.,.]

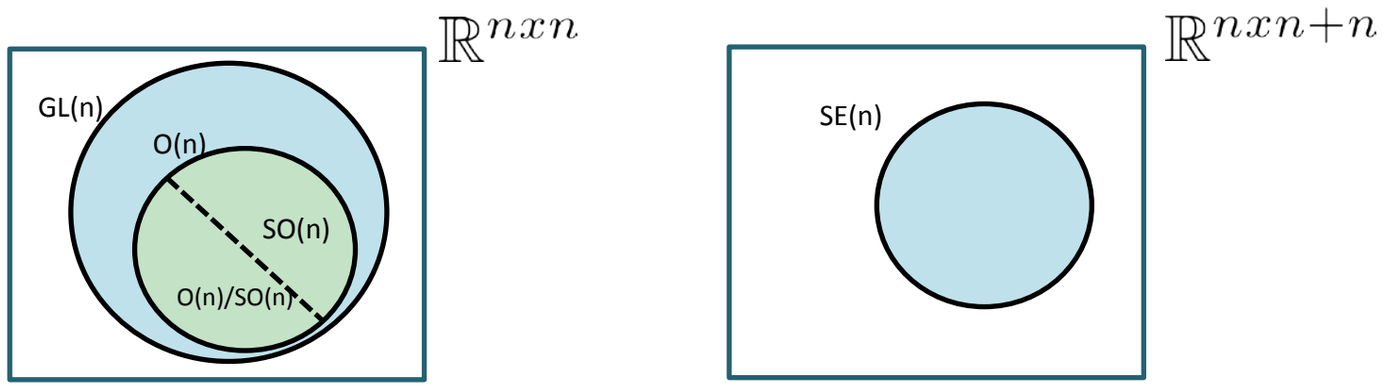
$$\widehat{a \times b} = \left[\widehat{a}, \widehat{b}\right]$$

Content

- Rigid transformations
- Matrix Groups
- Manifolds
- **Lie Groups/Lie Algebras**
- Charts on SO(2) and SO(3)

Lie Groups

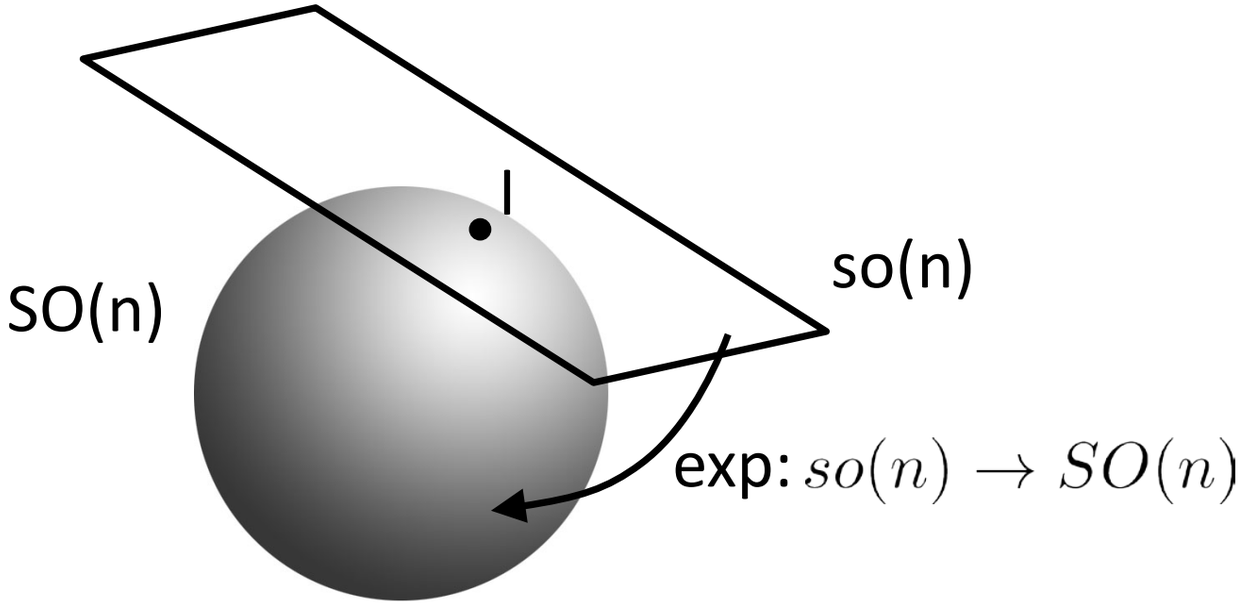
- GL(n), O(n), SO(n) and SE(n) are all **Lie groups**
  (groups which are also smooth manifold where the operation is a differentiable function between manifolds)



Exponential Map

- Given a Lie group **G**, with its related Lie Algebra **g = TG(I)**, there always exists a smooth map from Lie Algebra **g** to the Lie group **G** called **exponential map**

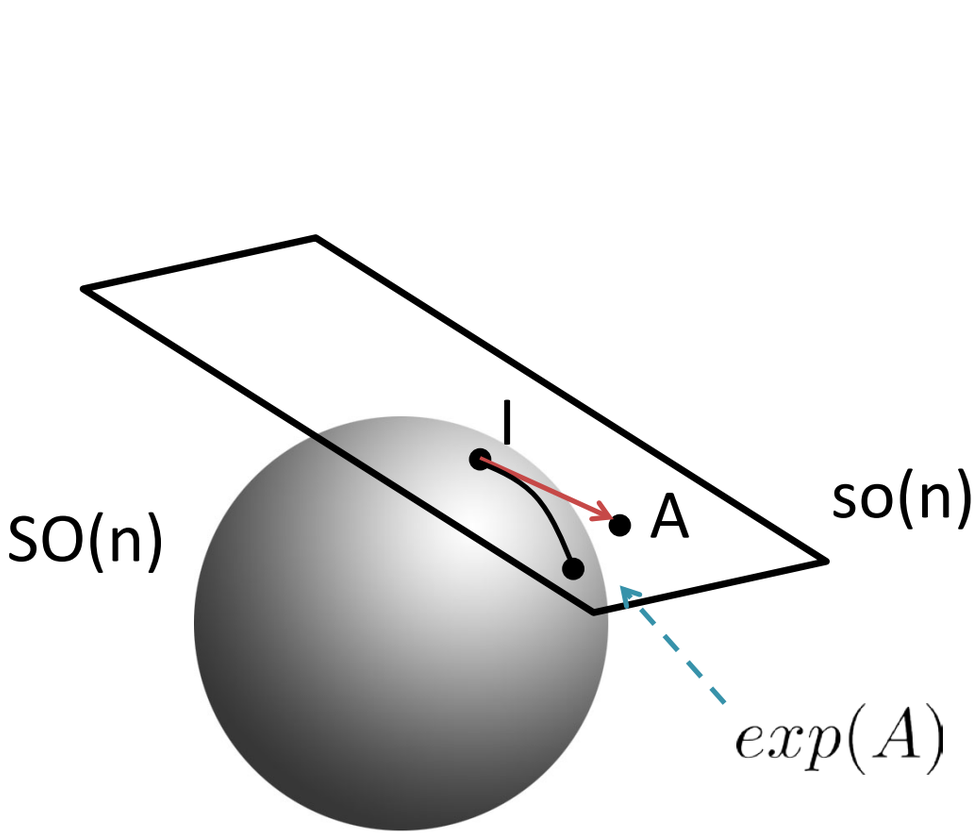
$$\exp : g \to G$$



Exponential Map

- Given a Lie group **G**, with its related Lie Algebra **g = TG(I)**, there always exists a smooth map from Lie Algebra **g** to the Lie group **G** called **exponential map**

$$\exp : g \rightarrow G$$



$exp(A) =$ **is the point in G that can be reached by traveling along the geodesic passing through the identity I in direction A, for a unit of time**

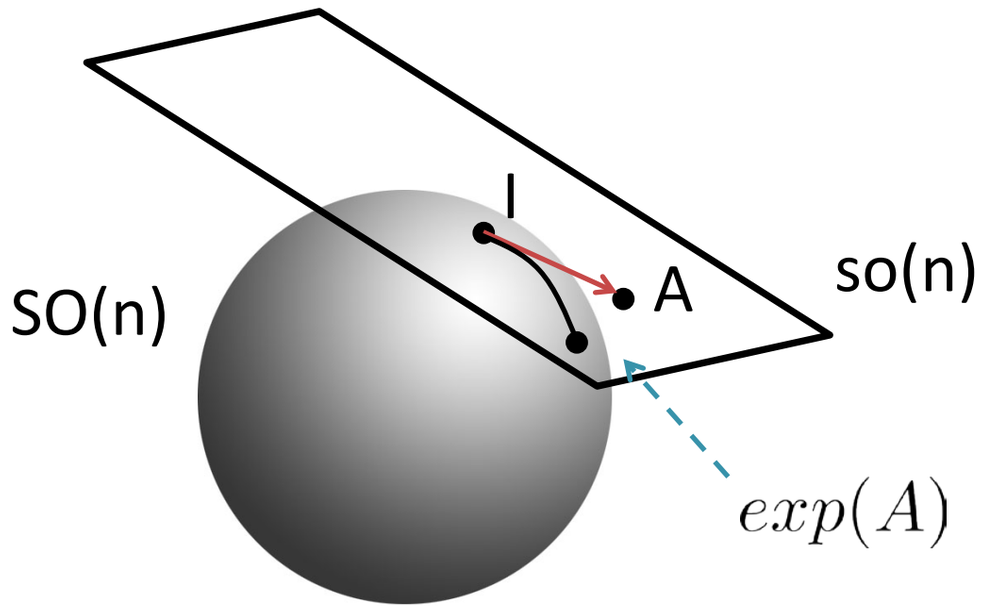
(Note: A defines also the traveling speed)

Exponential Map

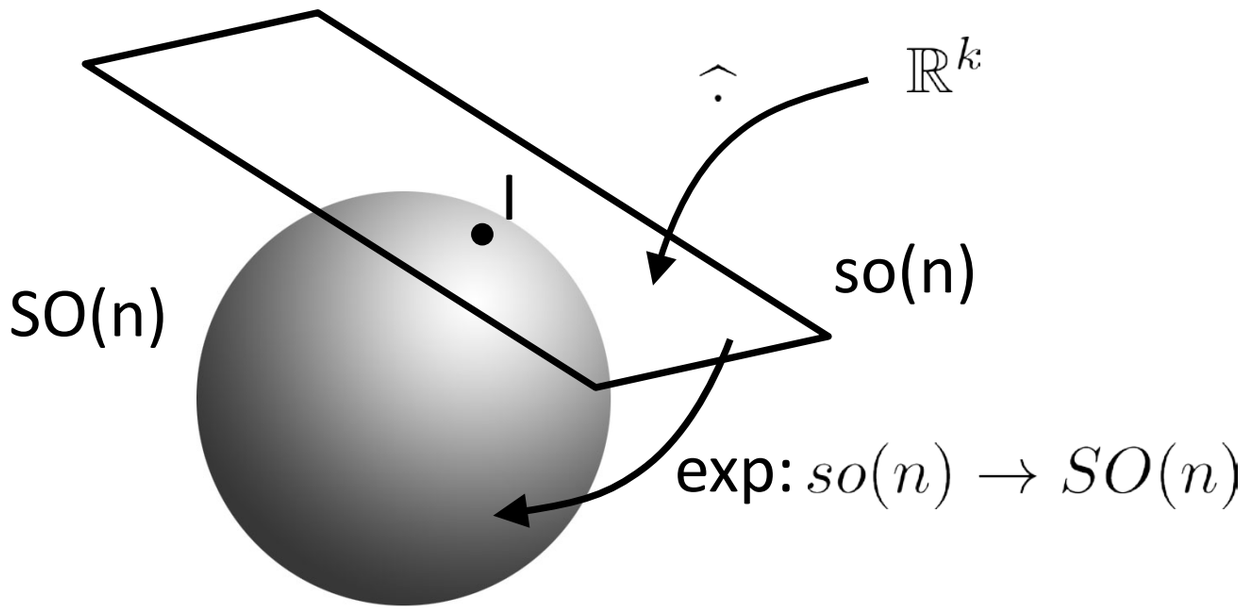
- The exponential map for a any matrix Lie group (GL(n), O(n), and SO(n)) coincides with the matrix exponential:

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

- it is **smooth map**
- it is **surjective** (it covers the Lie Group entirely)
- it is **not injective** (is a many to one map)



Exponential Map and Hat Operator



$$\omega \in \mathbb{R}^k$$

$$\widehat{\omega} \in so(n)$$

$$\exp(\widehat{\omega}) = \sum_{k=0}^{\infty} \frac{1}{k!} \widehat{\omega}^k \quad \in SO(n)$$

- composed with the hat operator, it is a **smooth** and **surjective** map from $\mathbb{R}^k$ to SO(n) (k = the dimension of the tangent space)

$$\omega \in \mathbb{R}^k \rightarrow \exp(\widehat{\omega})$$ **Angle-Axis representation**

*

Properties

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

$$e^{\widehat{0}} = e^0 = I$$

Identity

$$e^{-X} = (e^X)^{-1}$$

Inverse $\longrightarrow$ $e^{\widehat{-\omega}} = e^{-\widehat{\omega}} = (e^{\widehat{\omega}})^{-1}$

$$e^{X+Y} \neq e^X e^Y$$

in general not "Linear" **(different from the standard exp in $\mathbb{R}$)**

$$e^X e^Y \neq e^Y e^X$$

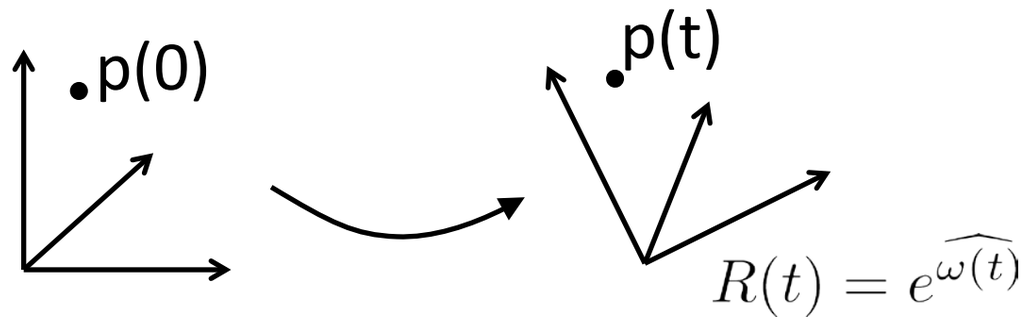
$$e^{sX+tX} = e^{sX} e^{tX} \qquad \forall t, s \in \mathbb{R}$$

$$\partial e^X = \partial X e^X = e^X \partial X$$

Derivative

*

Physical meaning of so(3)



$$p(t) = e^{\widehat{\omega(t)}} p(0)$$

Position

$$\frac{\partial p}{\partial t}(t) = \frac{\partial \widehat{\omega}}{\partial t}(t)\, e^{\widehat{\omega(t)}} p(0)$$

Velocity

$$= \frac{\partial \widehat{\omega}}{\partial t}(t)\; p(t)$$

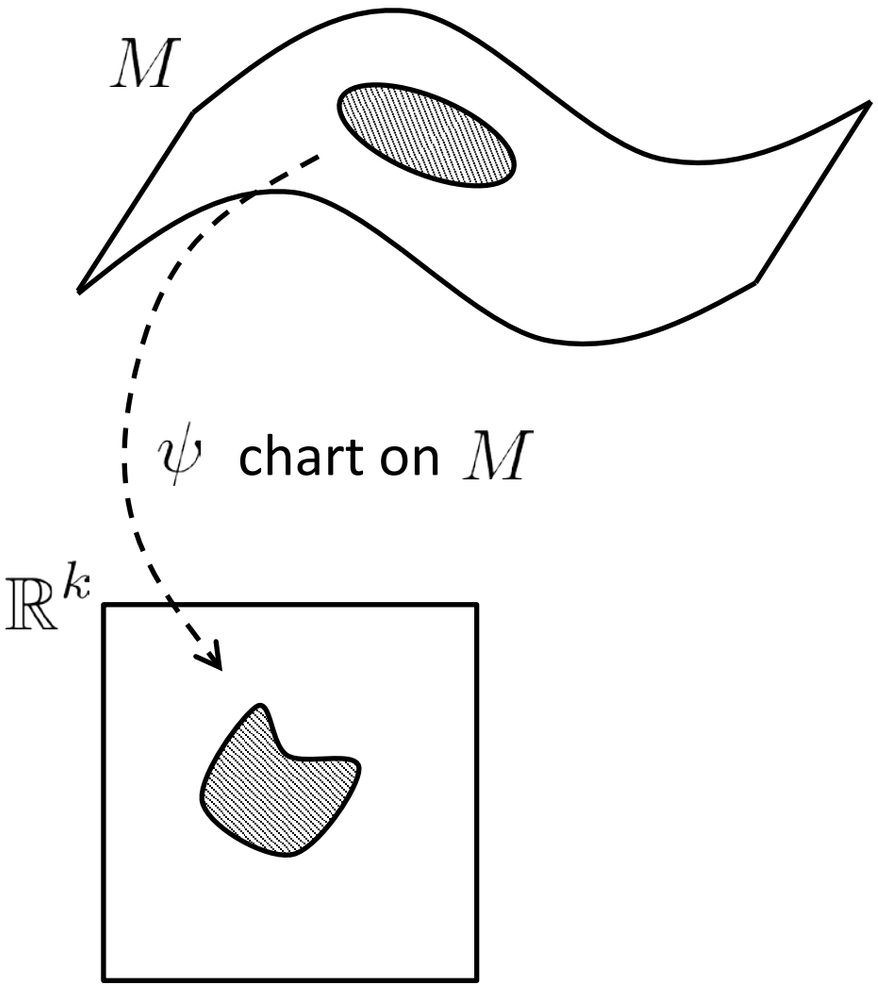
Spatial (angular) velocity $\in so(3)$

(transform each point in $\mathbb{R}^3$ into the corresponding speed that that point undergoes during the rotation at time t)

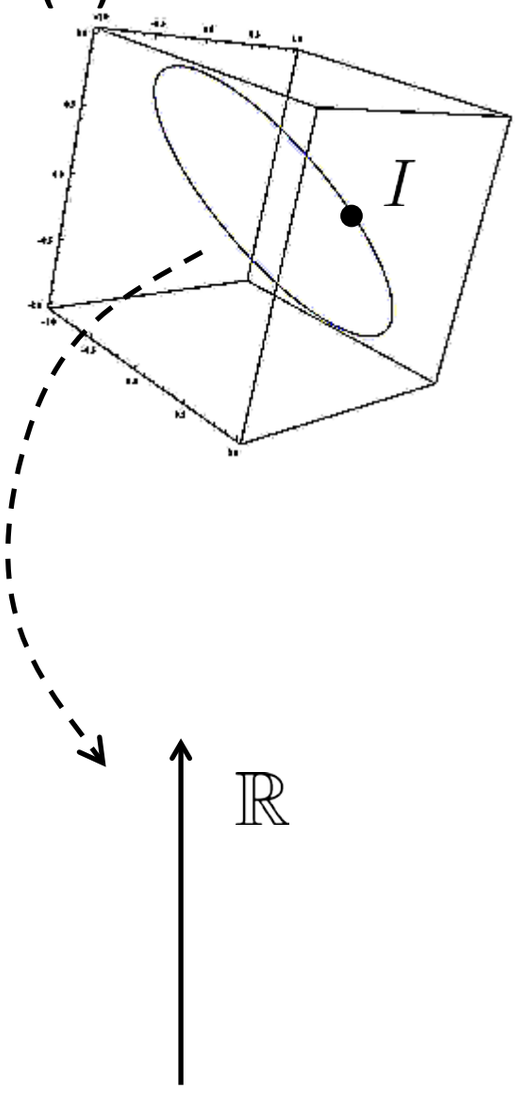
Content

- Rigid transformations
- Matrix Groups
- Manifolds
- Lie Groups/Lie Algebras
- **Charts on SO(2) and SO(3)**

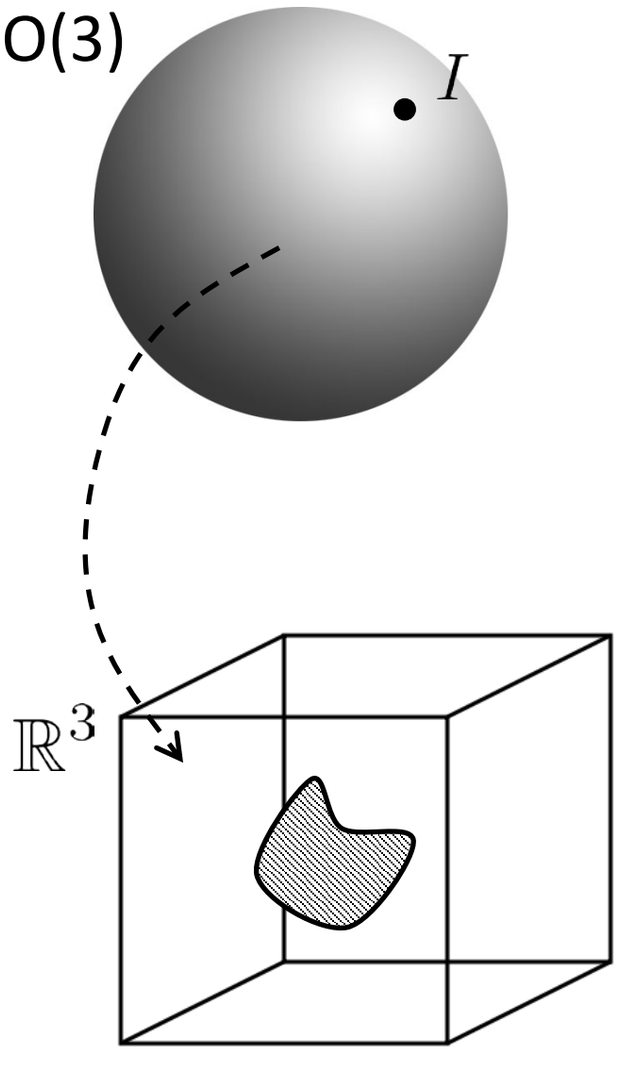
Charts on SO(2) and SO(3)



SO(2)

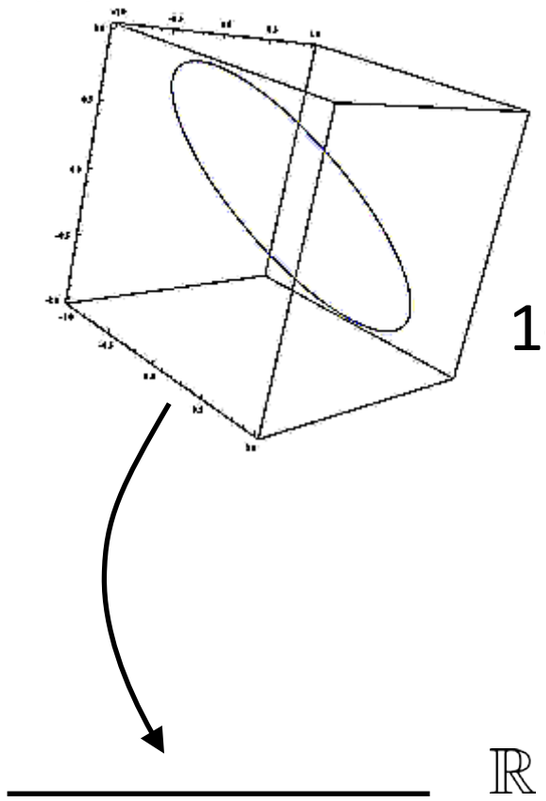


SO(3)



Charts on SO(2)

SO(2)



$$\gamma : [0, 2\pi) \rightarrow SO(2)$$

$$\gamma(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \in SO(2)$$

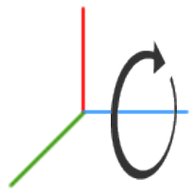
- Chart of SO(2), that cover the entire SO(2) using a single parameter

$$\min_{T \in SO(2)} f(T, \mathcal{O}) \quad \Rightarrow \quad \min_{\theta \in [0, 2\pi)} f(\gamma(\theta), \mathcal{O})$$

Charts on SO(3)

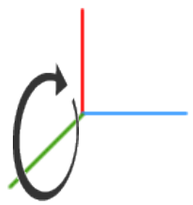
- **Euler's Theorem for rotations**: Any element in SO(3) can be described as a sequence of three rotations around the canonical axes, where no successive rotations are about the same axis.

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



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

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



$\in SO(3)$

- For any $R \in SO(3)$ there $\exists \alpha, \beta, \gamma \mid R = R_x(\alpha)R_y(\beta)R_z(\gamma)$

- α, β, γ are called **Euler angles** of R according to the XYZ representation

SO(3): Euler angles

- Given M there are 12 possible ways to represent it

$$M \in SO(3) \iff \exists \alpha, \beta, \gamma \mid M = R_x(\alpha) R_y(\beta) R_z(\gamma) \qquad \text{XYZ}$$

$$M \in SO(3) \iff \exists \alpha, \beta, \gamma \mid M = R_x(\alpha) R_z(\beta) R_y(\gamma) \qquad \text{XZY}$$

$$M \in SO(3) \iff \exists \alpha, \beta, \gamma \mid M = R_x(\alpha) R_z(\beta) R_x(\gamma) \qquad \text{XZX}$$

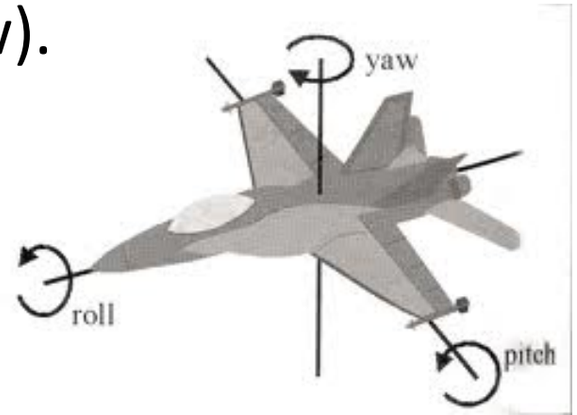
$$M \in SO(3) \iff \exists \alpha, \beta, \gamma \mid M = R_z(\alpha) R_x(\beta) R_z(\gamma) \qquad \text{ZXZ}$$

....

Remarks: multiplication is not commutative

- Unfortunately, all of them have the same drawbacks!! (see later)

- A common representation is ZYX corresponding to a rotation first around the x-axis (roll), then the y-axis (pitch) and finally around the z-axis (yaw).



SO(3): Euler angles

- The parameterization is non-linear
- The parameterization is modular $R_x(\alpha + 2k\pi) = R_x(\alpha)$
  (but this is something that we need to live with in any representation of SO(3))
- Beside the modularity, the parameterization is not unique:

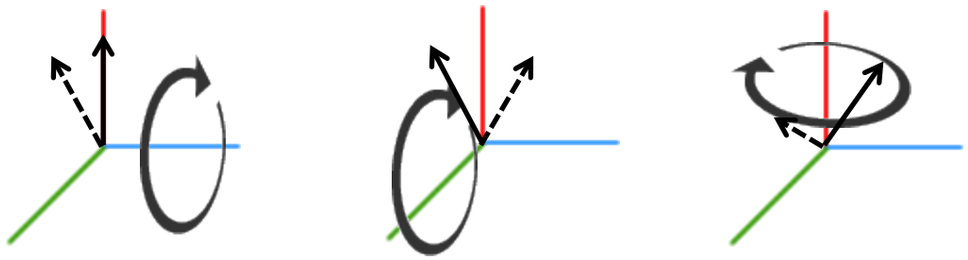
for some R in SO(3), $\exists \alpha_1, \beta_1, \gamma_1$ and $\alpha_2, \beta_2, \gamma_2$ such that

$$M = R_x(\alpha_1) R_y(\beta_1) R_z(\gamma_1)$$
$$M = R_x(\alpha_2) R_y(\beta_2) R_z(\gamma_3)$$

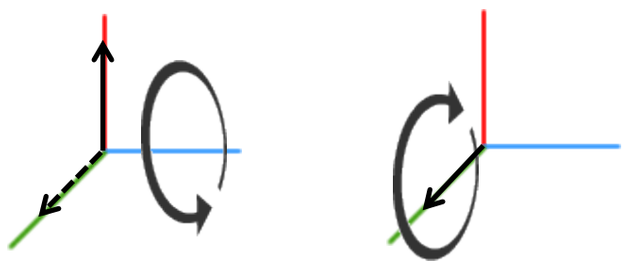
SO(3): Euler angles

- The parameterization have some singularities, called **gimbal lock**
- a gimbal lock happens when after a rotation around an axis, two axes align, resulting in a loss of one degree of freedom

(a)



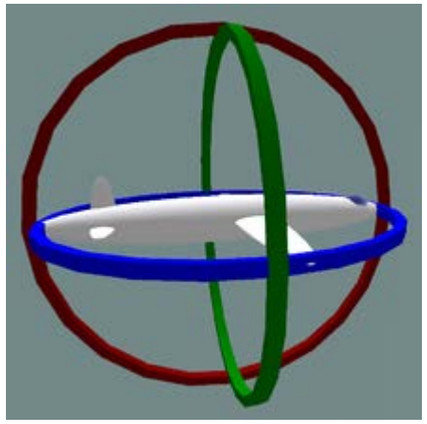
(b)



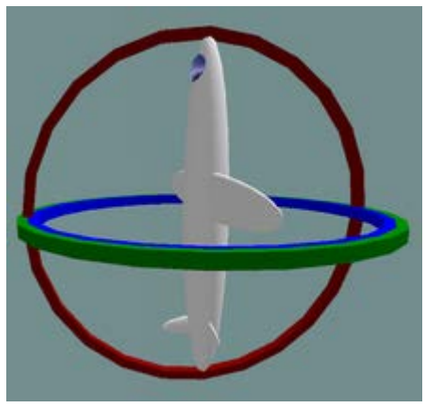
cannot move (lock)

SO(3): Euler angles

- The parameterization have some singularities, called g**imbal lock**
- a gimbal lock happens when after a rotation around an axis, two axes align, resulting in a loss of one degree of freedom
- The name gimbal lock derives from the gimbal



normal



lock [wikipedia]

- Even the most advanced modeling software uses Euler angles to parameterize the orientation of the rendering window. This is because Euler angles are more intuitive to the user. As a drawback, the gimbal lock is often noticeable.

Euler Angles and Angle/Axis

- The Euler angle representation say

$$R \in SO(3) \iff \exists \alpha, \beta, \gamma \mid R = R_x(\alpha) R_y(\beta) R_z(\gamma)$$

XYZ representation

$$= e^{\alpha \hat{x}} e^{\beta \hat{y}} e^{\gamma \hat{z}}$$

possible Gimbal lock

$$\neq e^{\alpha \hat{x} + \beta \hat{y} + \gamma \hat{z}}$$

no Gimbal lock

- while Euler angle define 3 rotation matrices, the angle/axis representation define a single rotation matrix identified by an element of R3