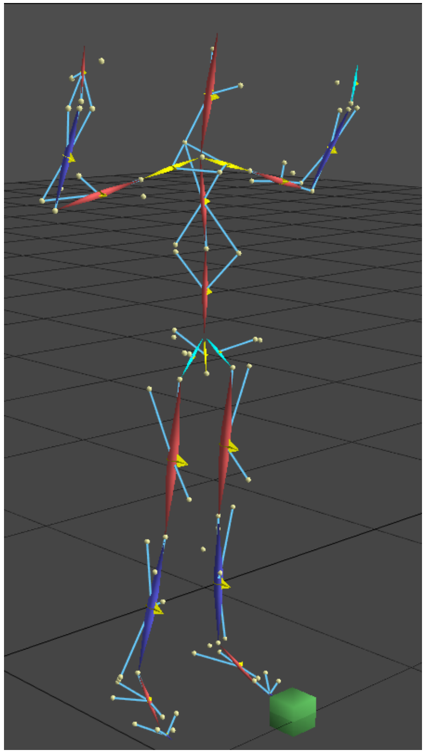
Model Based Pose Estimation

Gerard Pons-Moll and
Bodo Rosenhahn

Institute for Information Processing

- Motion Analysis, Vicon (55+ Marker, 1000fps, Strobe-lights)



Overview

Human pose estimation algorithms can be classified in:

- Generative Models
  - *„explain the image“*

- Discriminative Models
  -*„condition on the image“*

Is it hard ?

- Many degrees of freedom
- Highly Dynamic / Skinning/ Clothing / Outdoor
- Large variability and individuality of Motion patterns

http://www.google.de/images

Is it hard ?

- Let us assume a ***model*** then:
  - How to parameterize/represent the model ?
  - How to optimize the model parameters ?
  - What features are suited ?




http://www.google.de/images

Overview

1) Kinematic parameterization

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

2) Subject model

- Geometric primitives
- Detailed Body Scans
- Human Shape models

3) Inference

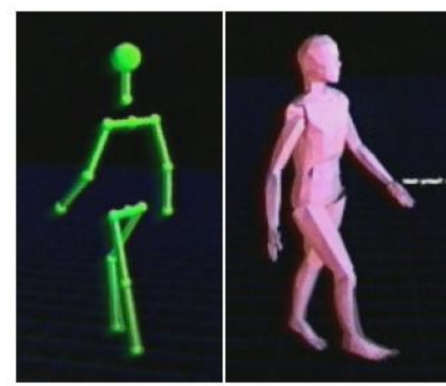
- Observation likelihood
- Local optimization
- Particle Based optimization


Karate

Kinematic Chains

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

The model parameters to optimize for are rigid body motions.



Bregler et.al. CVPR-98

How to model RBM ?

Parameterization

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

2) Singularities can be avoided during optimization

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

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

5) Simple rules for concatenating motions

Definition

A rigid body motion is an affine transformation that preserves distances and orientations

Euclidean : (non linear)

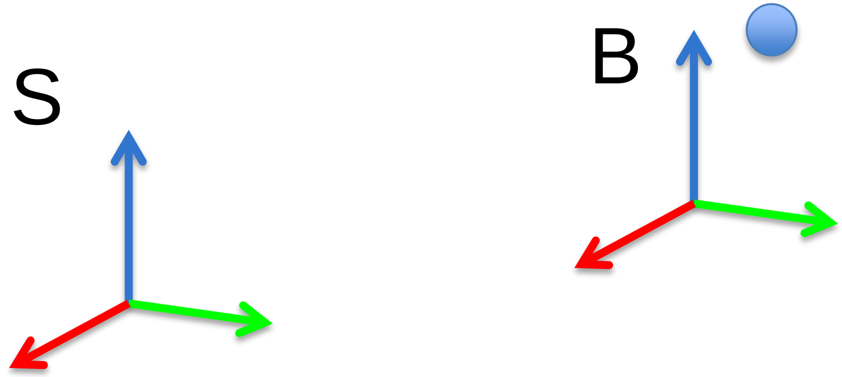
$$X = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \rightarrow X' = RX + t, R \in \mathcal{R}^{3 \times 3}, t \in \mathcal{R}^3$$

$$RR^T = I, \det(R) = 1$$

Affine: (linear)

$$X = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ 1 \end{pmatrix} \rightarrow X' = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} X$$

Rotation Matrices



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

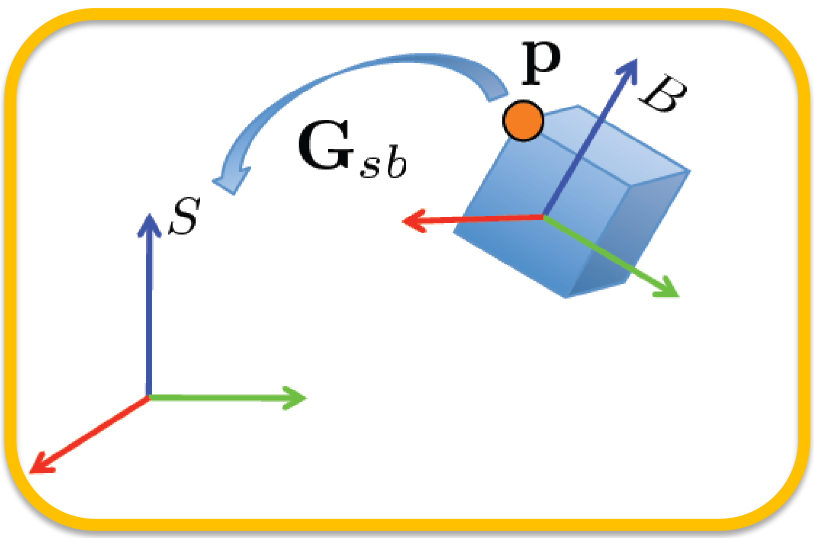
$$\mathbf{p}_s = \lambda_x \mathbf{x}_s^B + \lambda_y \mathbf{y}_s^B + \lambda_z \mathbf{z}_s^B$$

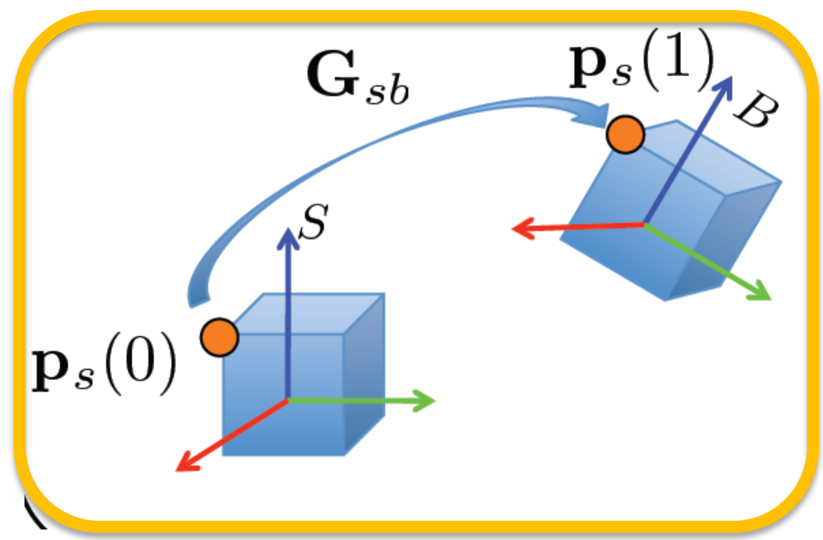
$$\mathbf{p}_s = \mathbf{R}_{sb}\,\mathbf{p}_b \quad \Rightarrow \quad \mathbf{R}_{sb} = \begin{bmatrix} \mathbf{x}_s^B & \mathbf{y}_s^B & \mathbf{z}_s^B \end{bmatrix}$$

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

2 Views of Rotations

Rotations can be interpreted either as





Coordinate transformation

Relative motion in time

Rotation matrix drawbacks

- Need for **9 numbers**
- **6 additional constrains** to ensure that the matrix is orhtornormal
- Suboptimal for optimization

Euler Angles

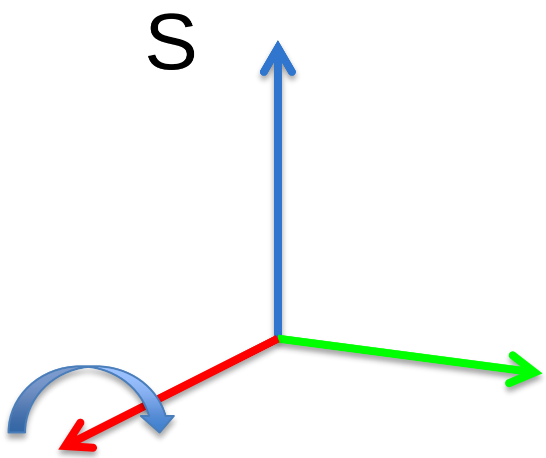
- One of the most **popular** parameterizations

- Rotation is encoded as **the successive rotations** about three principal axis

- Only **3 parameters** to encode a rotation

- **Derivatives** easy to compute

Euler Angles



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

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

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

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

Euler Angles: confusion

Careful: Euler angles are a typical source of confusion

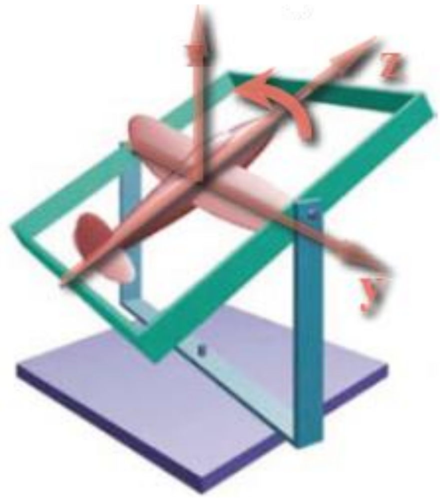
When using Euler angles **2 things** have to be specified

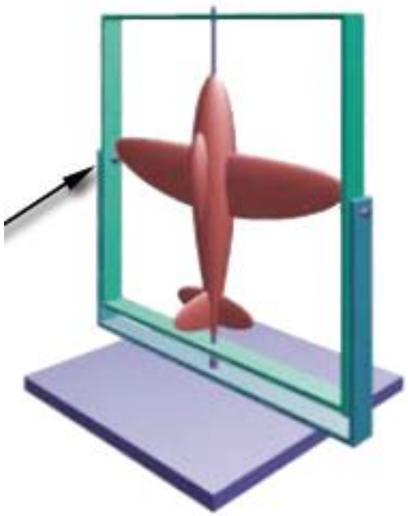
1) Convention: X-Y-Z, Z-Y-X, Z-Y-Z …

2) Rotations about the static spatial frame or the moving body frame

Euler Angles: drawbacks I

- **Gimbal lock:** When two of the axis align one degree of freedom is lost !
- Parameterization is not unique
- Lots of conventions for Euler angles





Quaternions

- A quaternion has 4 components:

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

- They generalize complex numbers

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

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

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

Quaternions

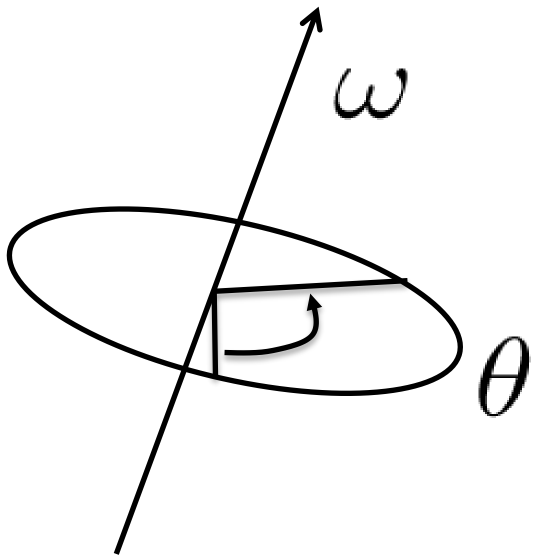
- Quaternions can also be interpreted as a scalar plus a 3-vector

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

Where

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

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



Quaternions

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

  - Concatenation of rotations:

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

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

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

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

Quaternions

- ✓ Quaternions have no singularities
- ✓ Derivatives exist and are linearly independent
- ✓ Quaternion product allows to perform rotations
- ✗ But all this comes at the expense of using 4 numbers instead of 3

- Enforce quadratic constrain $||\mathbf{q}||_2 = 1$

Axis-angle

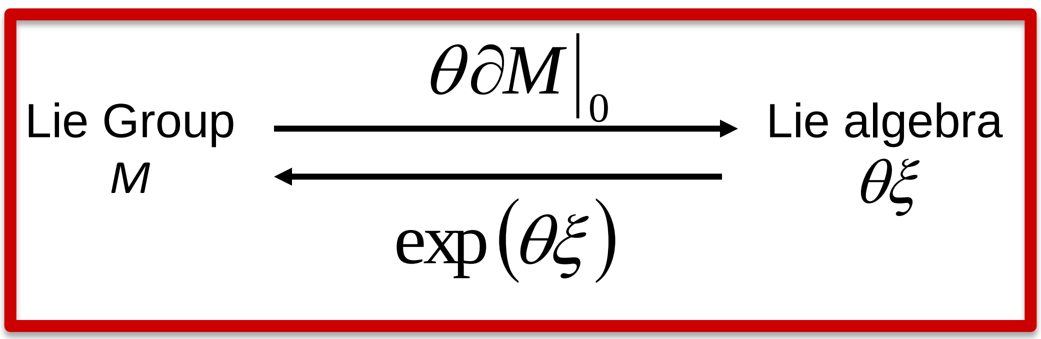
For human motion modeling it is often needed to specify the axis of rotation of a joint

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

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

Lie Groups / Lie Algebras

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



Lie Groups / Lie Algebras

$$so(2) = \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix}, \theta\partial \begin{pmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{pmatrix} \Big|_0$$

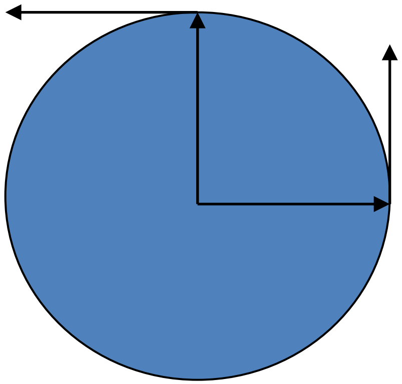
$$= \theta \begin{pmatrix} -sin(\phi) & -cos(\phi) \\ cos(\phi) & -sin(\phi) \end{pmatrix} \Big|_0 = \theta \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \theta\hat{\omega}$$

$$so(2) = \{A \in \mathcal{R}^{2\times 2} | A = -A^T\}$$

If a body rotates at constant velocity about an axis, the velocity can be written as

$$\dot{q}(t) = \hat{\omega}q(t) \quad (1)$$

Example: $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$



(1) Is a time invariant linear differential equation which may be integrated to give:

$$q(t) = \exp(\hat{\omega}t)q(0)$$

Axis-angle

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

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

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

It is the matrix form of the cross-product:

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

Exponential map

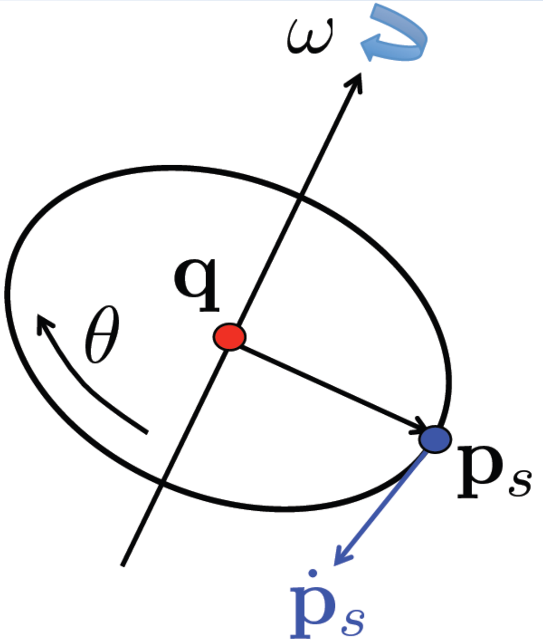
Proof: exponential map

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

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

If we rotate θ units of time

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



Exponential map

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

Exploiting the properties of skew symetric matrices

Rodriguez formula

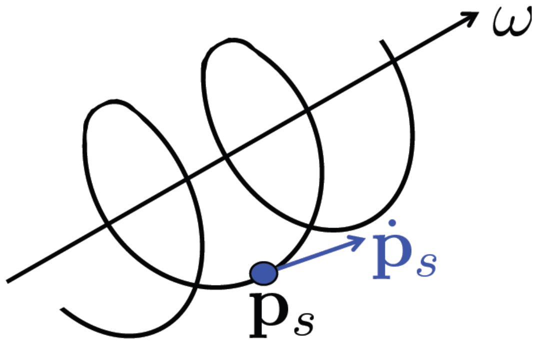
$$\exp(\theta\widehat{\omega}) = I + \widehat{\omega}\sin(\theta) + \widehat{\omega}^2(1-\cos(\theta))$$

Closed form!

What about translation ?

The **twist coordinates** are defined as

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



And the **twist** is defined as

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

Exponential map

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

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

From the following formula

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

Ranking

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

Overview

1) Kinematic parameterization

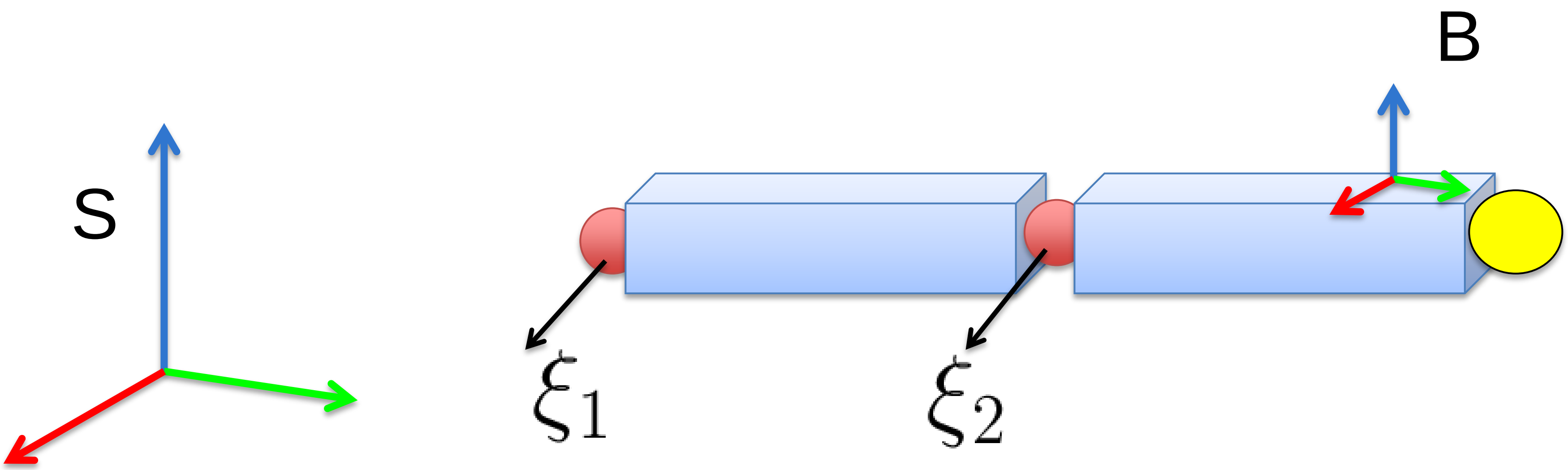
- Rotation Matrices
- Euler Angles
- Quaternions
- Twists and Exponential maps
- Kinematic chains

2) Subject model

- Geometric primitives
- Detailed Body Scans
- Human Shape models

3) Inference

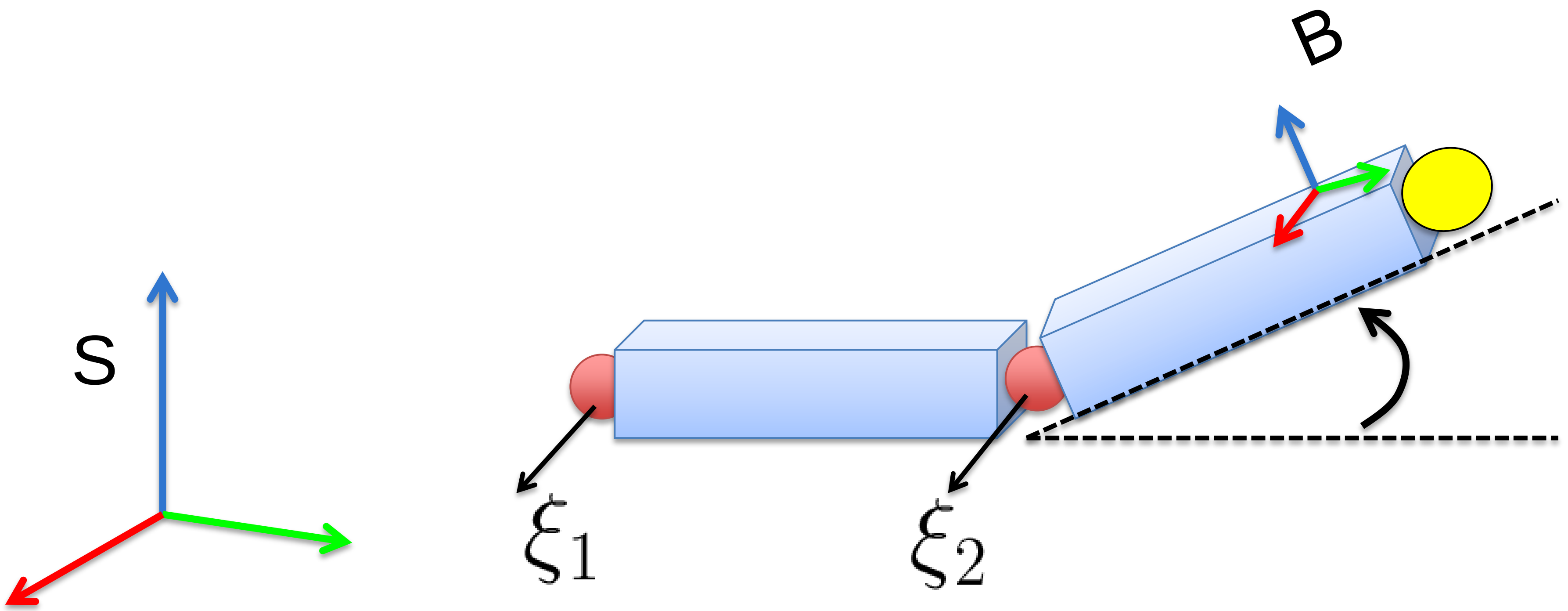
- Observation likelihood
- Local optimization
- Particle Based optimization



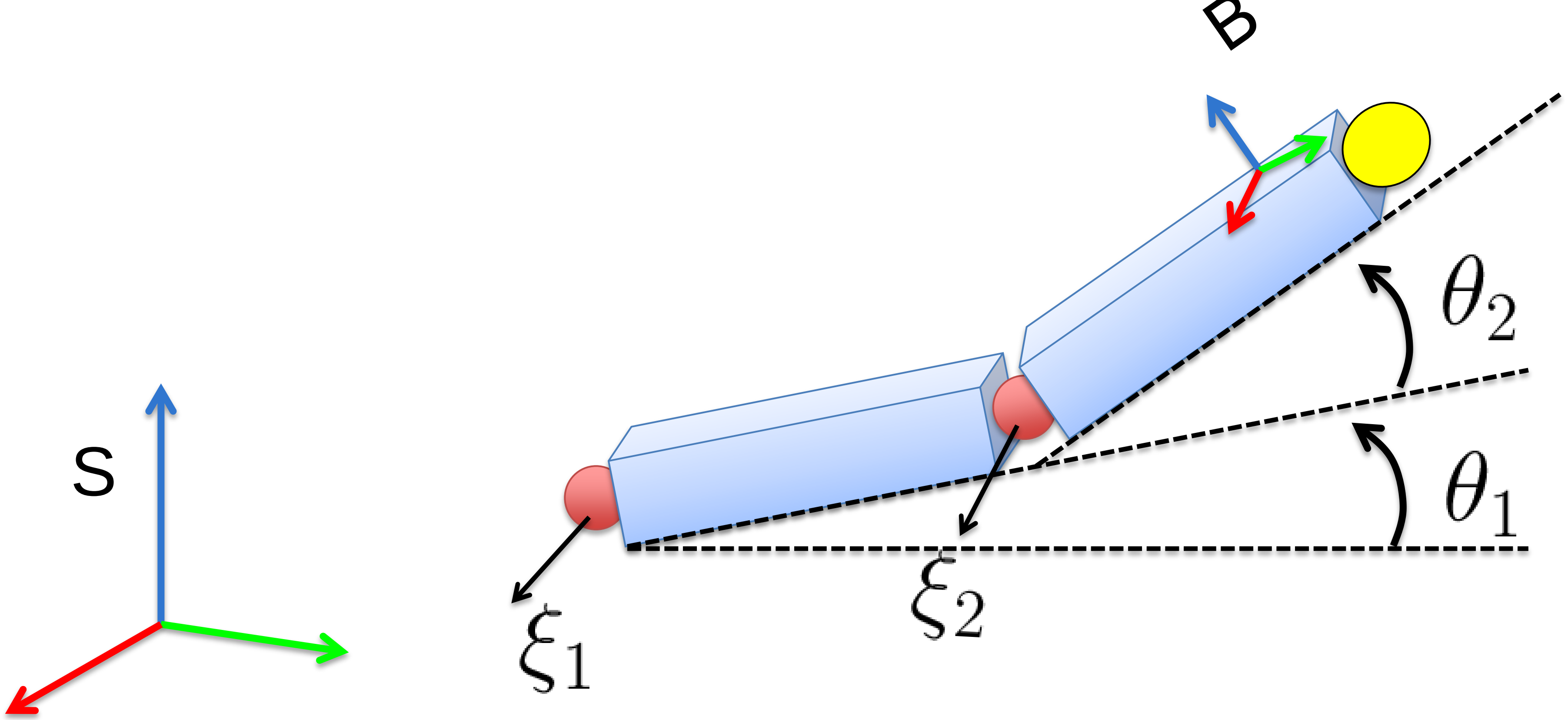
In a rest position we have

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

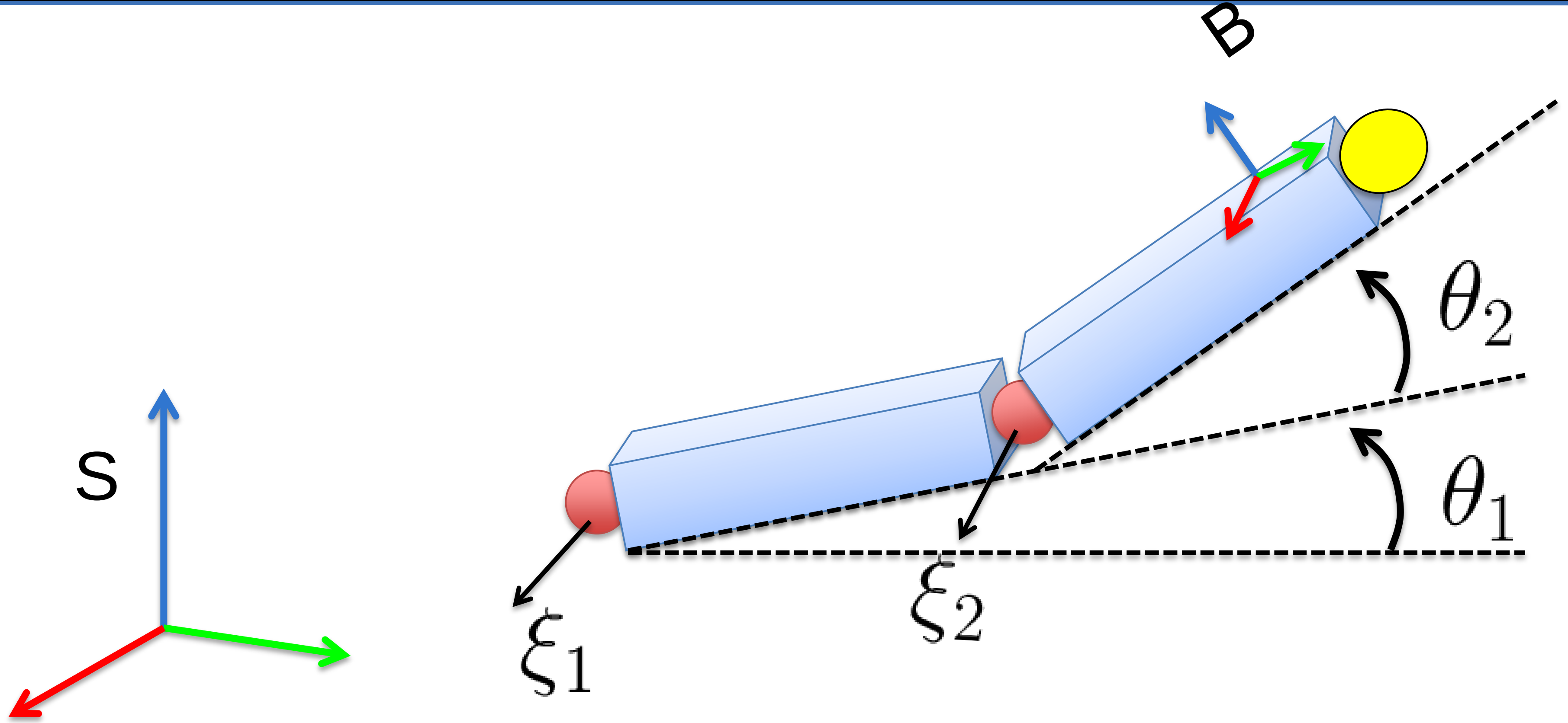
Articulation



Articulation



Articulation



The coordinates of the point in the spatial frame

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

Product of exponentials

Product of exponentials formula

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

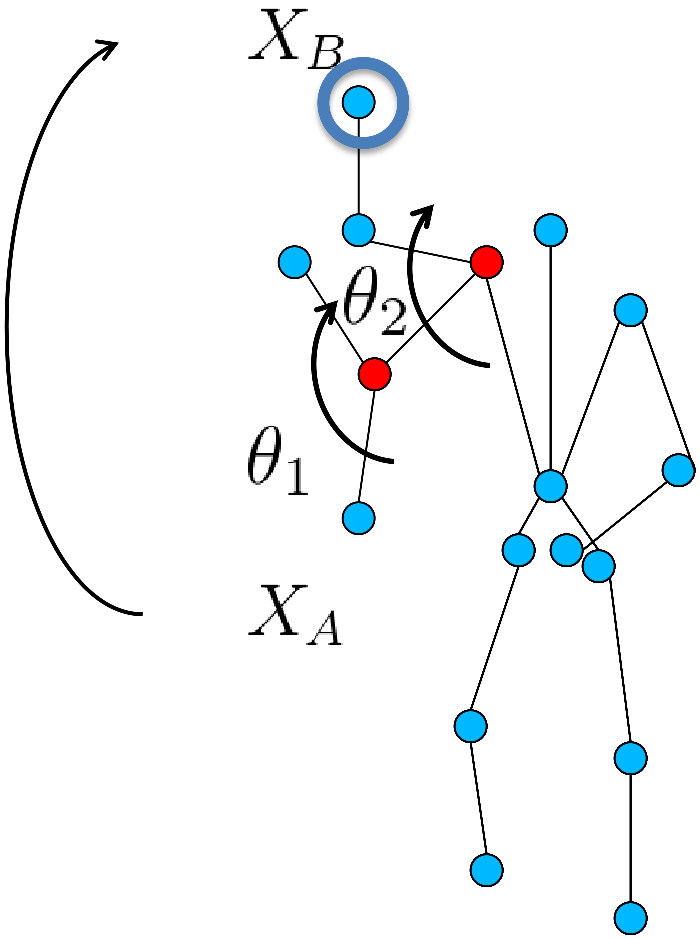
$\mathbf{G}_{sb}(\Theta)$ is the mapping from coordinate B to coordiante S

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

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

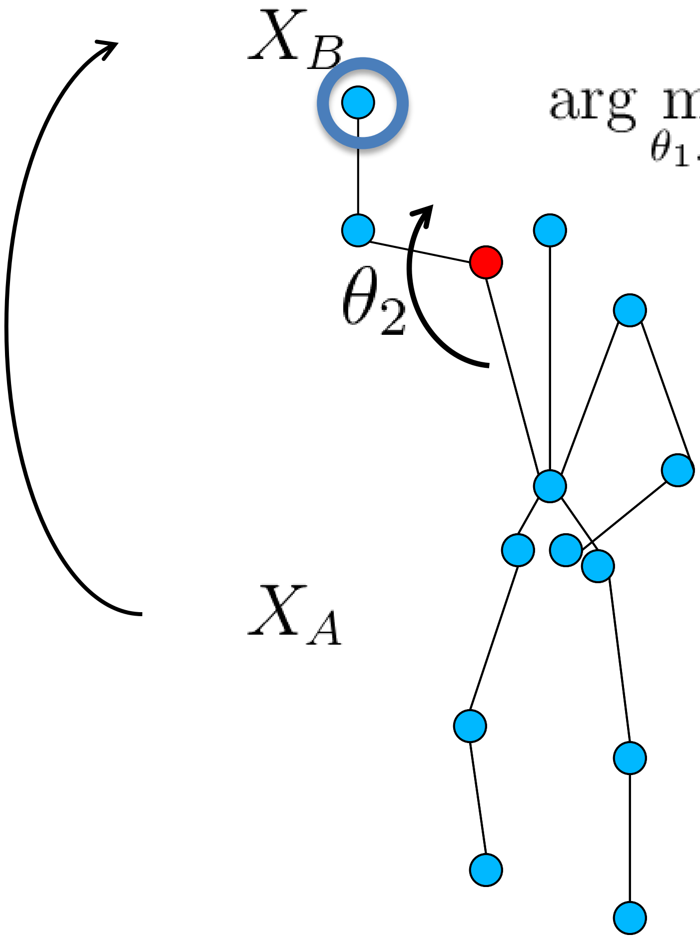
Inverse Kinematics

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



Inverse Kinematics

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



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

- The problem is non-linear

- Linearize with the articulated **Jacobian**

Articulated Jacobian

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

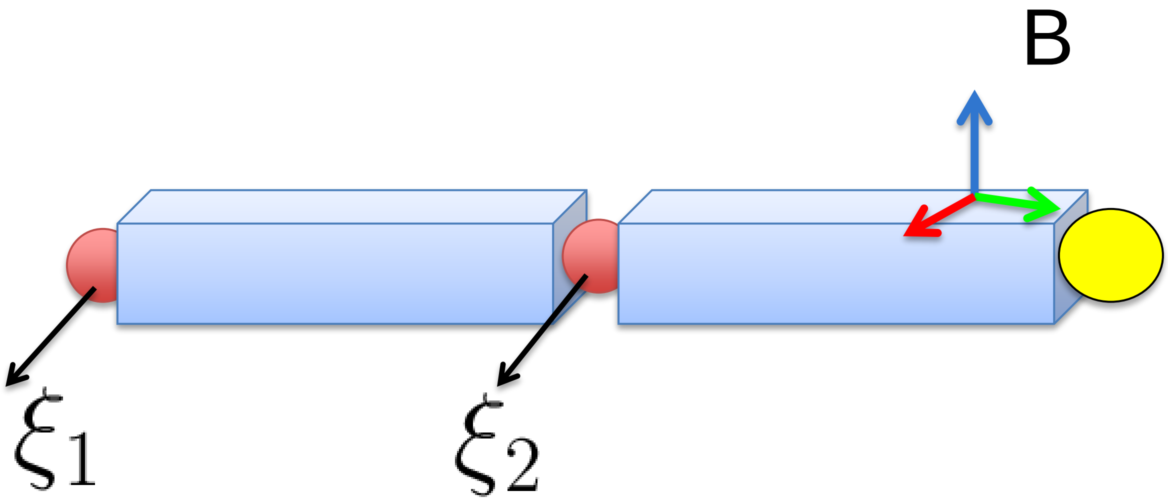
$$\mathbf{J}_{\Theta} = \begin{bmatrix} \xi_1 & \xi_2' & \ldots & \xi_n' \end{bmatrix}$$

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

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

Articulated Jacobian

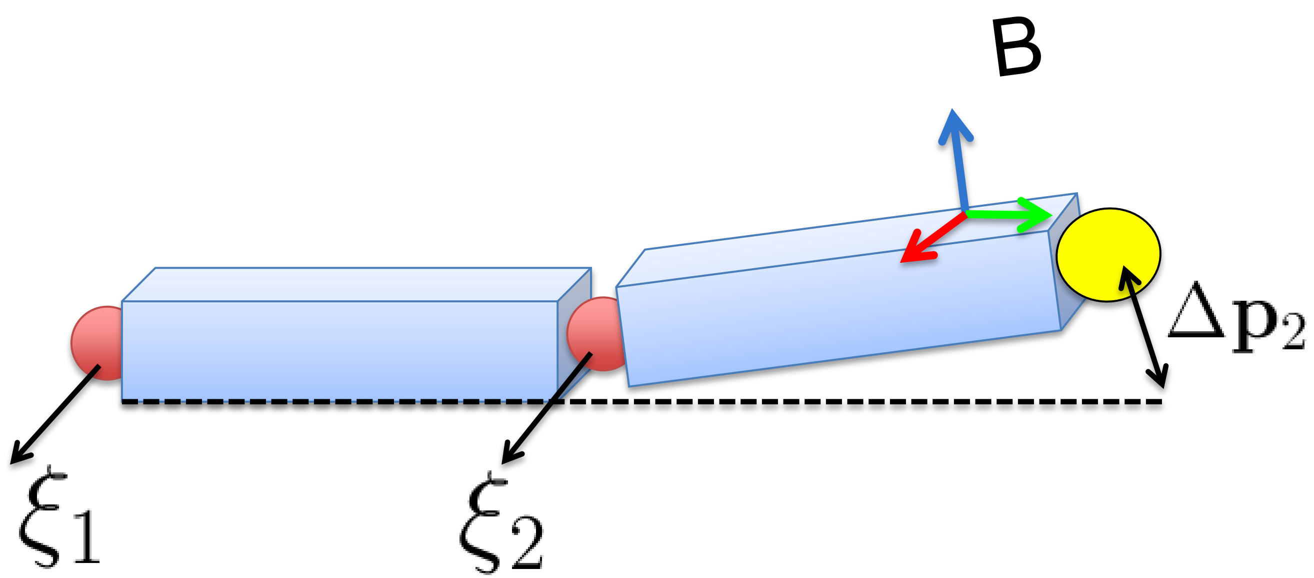
Intuition: Linear combination of twists



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

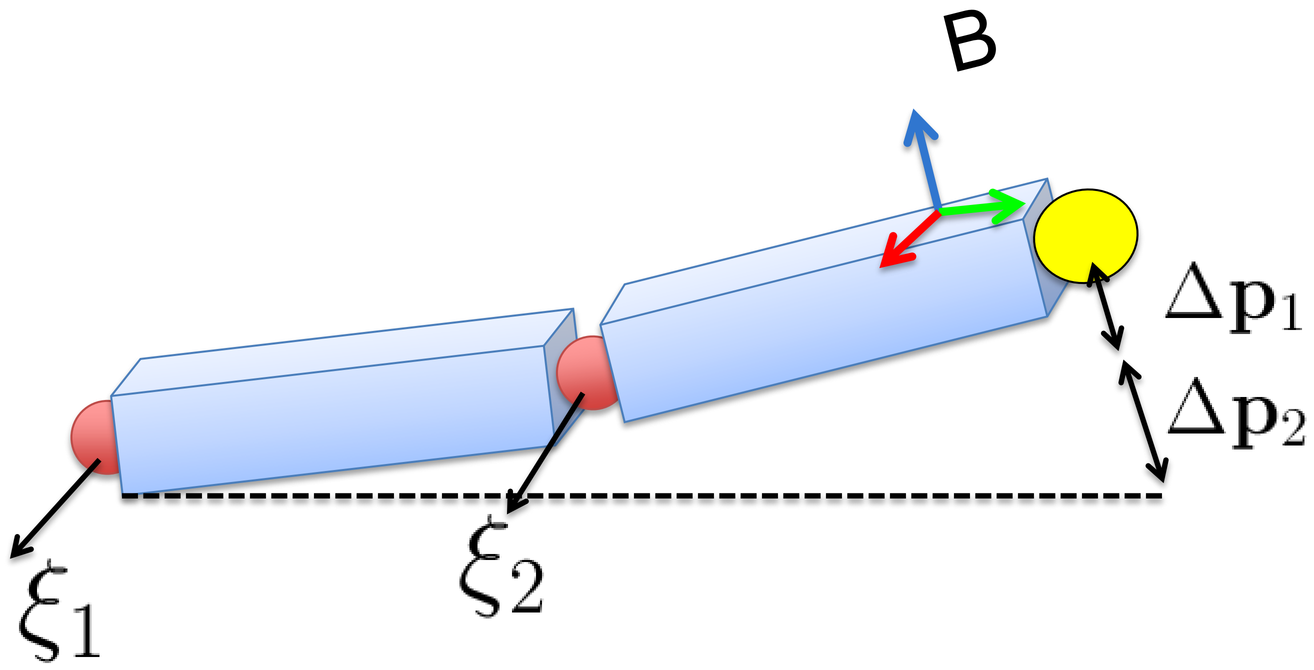
Articulated Jacobian

Intuition: Linear combination of twists



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

Intuition: Linear combination of twists

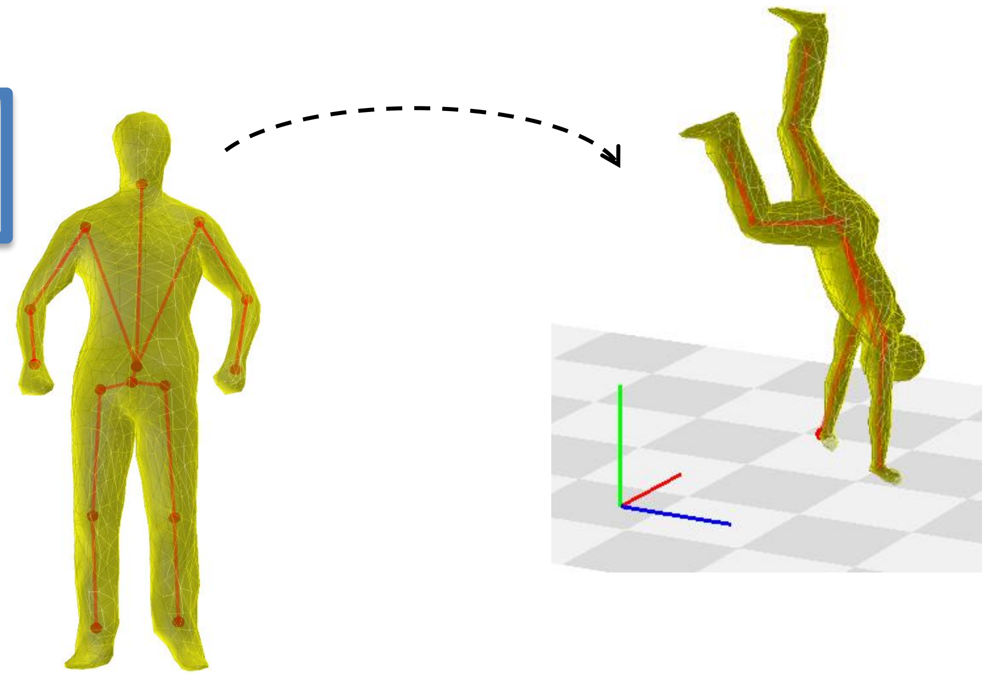


$$\Delta\bar{\mathbf{p}}_s = [\,\mathbf{J}_\Theta\ \cdot \Delta\Theta\,]^\wedge\,\bar{\mathbf{p}}_s = [\xi_1 \Delta\theta_1 + \xi_2' \Delta\theta_2 + \ldots + \xi_n' \Delta\theta_n]^\wedge \bar{\mathbf{p}}_s$$

Pose Parameters

Pose parameters: root + joint angles

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



Pose Jacobian

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

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

$$\mathbf{J}_{\mathbf{x}}(\mathbf{p}_s) = \left[\begin{array}{cccccc} \mathbf{I}_{[3\times3]} & -\mathbf{p}_s^{\wedge} & \widehat{\xi}_1\,\bar{\mathbf{p}}_s & \widehat{\xi}'_2\,\bar{\mathbf{p}}_s & \cdots & \widehat{\xi}'_n\,\bar{\mathbf{p}}_s \end{array} \right]$$

6 columns of Root

N columns for one per joint

Overview

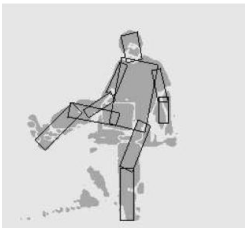
1) Kinematic parameterization
- Rotation Matrices
- Euler Angles
- Quaternions
- Twists and Exponential maps
- Kinematic chains

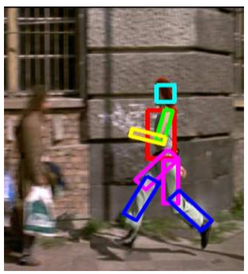
2) Subject model
- Geometric primitives
- Detailed Body Scans
- Human Shape models

3) Inference
- Observation likelihood
- Local optimization
- Particle Based optimization

Geometric primitives

2D

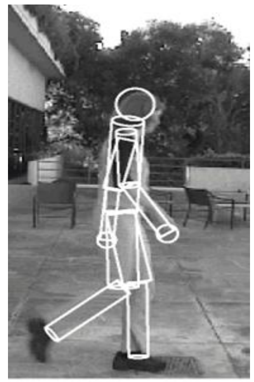


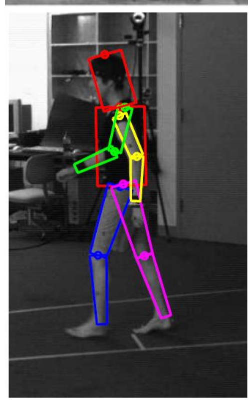


Felzenszwalb et.al
Ramanan et.al.
Andriluka et.al.

3D

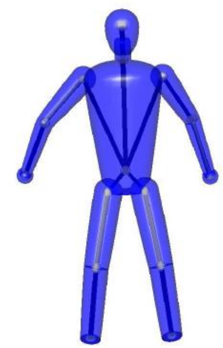
Cylinders

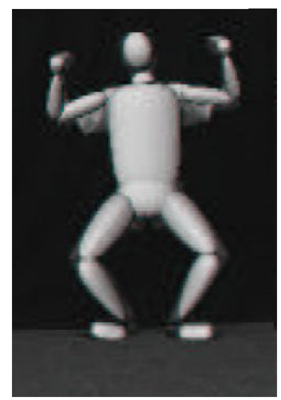




Kjellström et.al.
Sigal et.al.

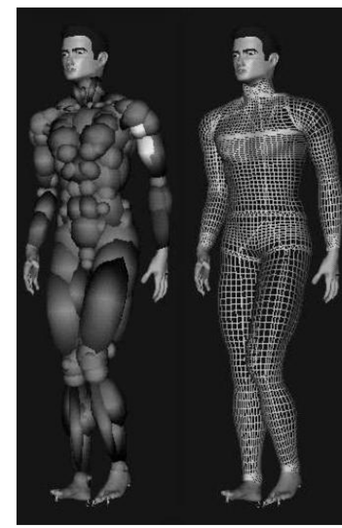
Ellipsoids





Kehl and Van Gool
Sminchisescu and Triggs

Gaussian Blobs



Plaenkers and Fua

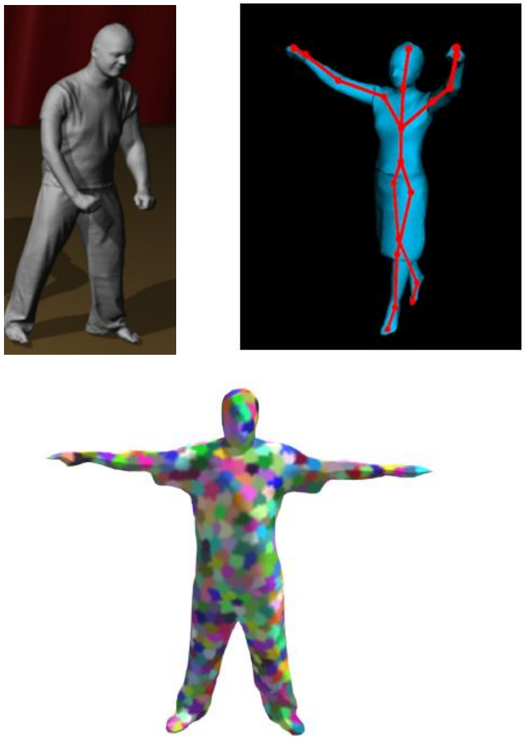
Detailed models

Rigged Subject Scan

Pons-Moll et.al.
Rosehnahn et.al.
Hasler et.al.

~ 30 DoF
- Kinematic model

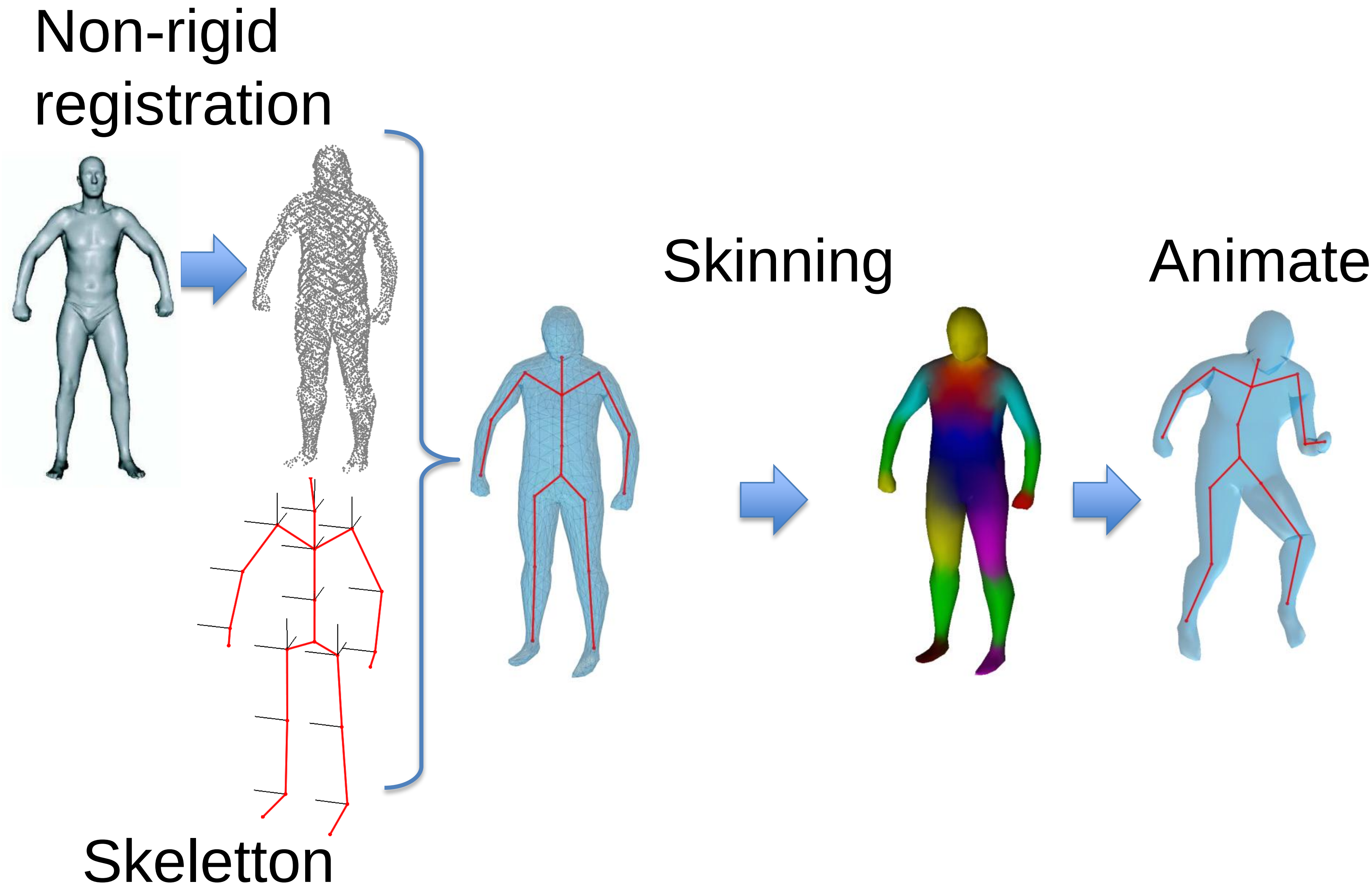
Free form Surface



Aguiar et.al.
Gall et.al
Cagniart et.al.

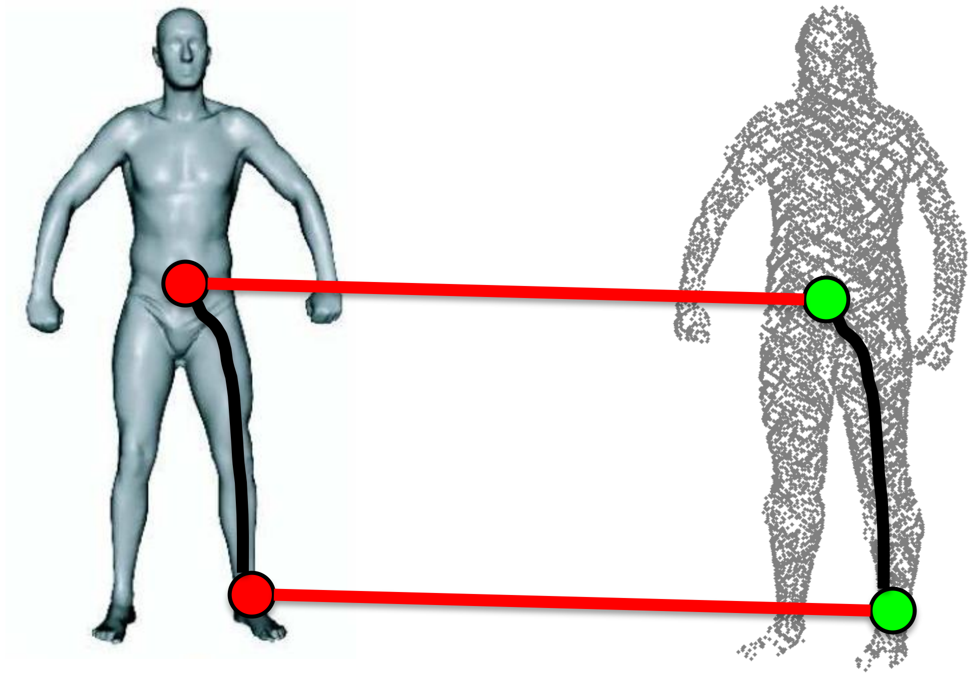
- > 1000 DoF
- with ++ constrains

Model Rigging



Fit a template

Correspondences of **pairwise** points with similar **local regions** and similar **geodesic distances**



Template   Point cloud

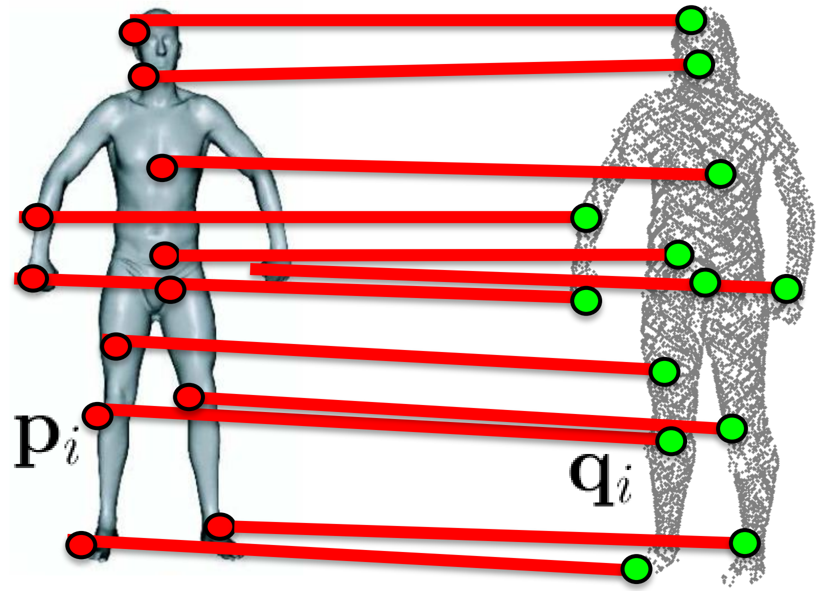
Loopy belief propagation

Anguelov et.al

Non-rigid registration

$$\mathbf{X} = [\mathbf{T}_1\,\mathbf{T}_2 \ldots \mathbf{T}_n]$$

$\mathbf{T}_i$ 3x4 affine matrix



Least squares

$$\mathrm{E}(\mathbf{X}) = \alpha \underbrace{\sum_i \|\mathbf{T}_i \mathbf{p}_i - \mathbf{q}_i\|^2}_{\text{Distance term}} + \beta \underbrace{\sum_i \sum_j w_{ij} \|\mathbf{T}_i - \mathbf{T}_j\|_F^2}_{\text{Smoothness term}}$$

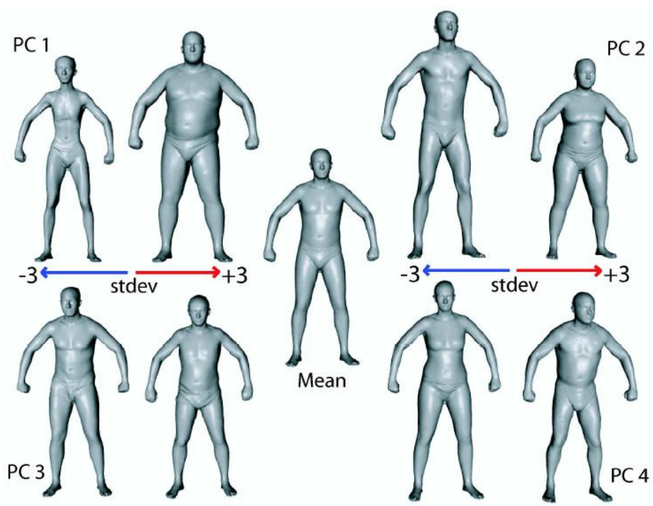
Distance term

Smoothness term

Shape and Pose models

Learn a PCA model of shape

Hasler et.al



Anguelov et.al

SCAPE

Infer model parameters from images

Ballan et.al

Guan et.al

Hasler et.al

Overview

1) Kinematic parameterization

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

2) Subject model

- Geometric primitives
- Detailed Body Scans
- Human Shape models

3) Inference

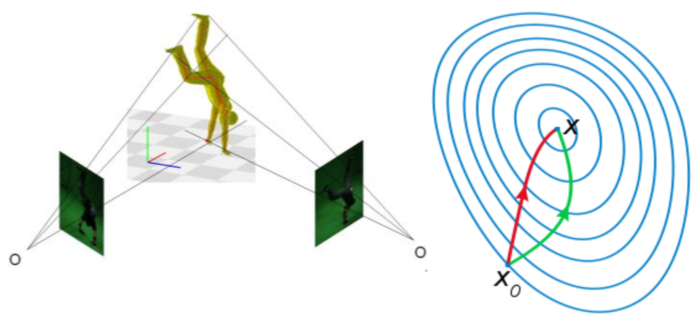
- Observation likelihood
- Local optimization
- Particle Based optimization

Inference

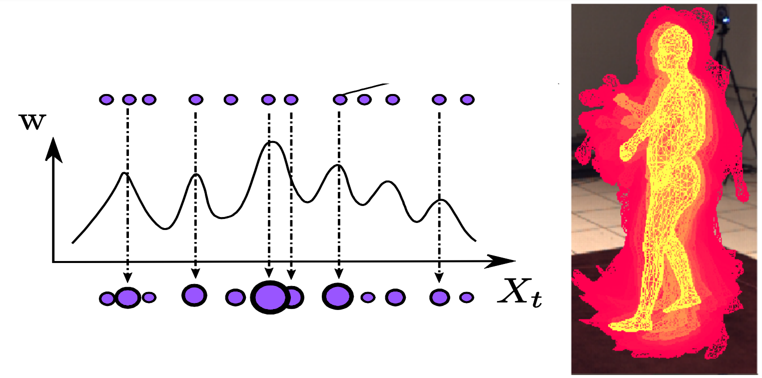
Generative models

$$p(\mathbf{x}|\mathbf{I}) \quad \propto \quad p(\mathbf{I}|\mathbf{x}) \quad \times \quad p(\mathbf{x})$$

Posterior Likelihood × Prior



Optimization



Bayesian models

Map of $p(\mathbf{x}|\mathbf{I})$

Approx. $p(\mathbf{x}|\mathbf{I})$ with weighted samples

Optimization

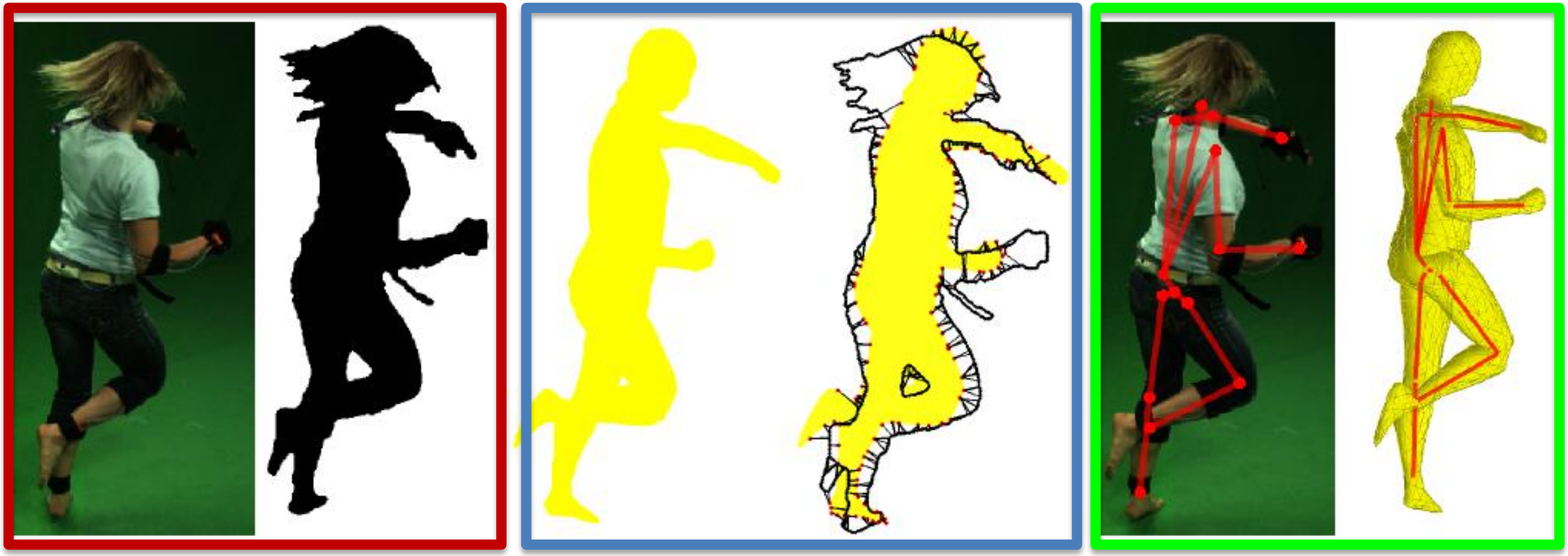
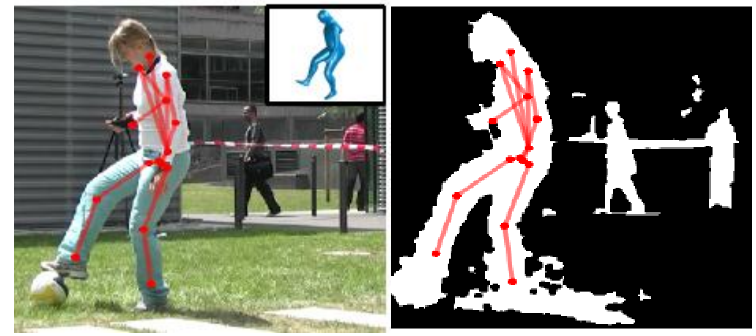
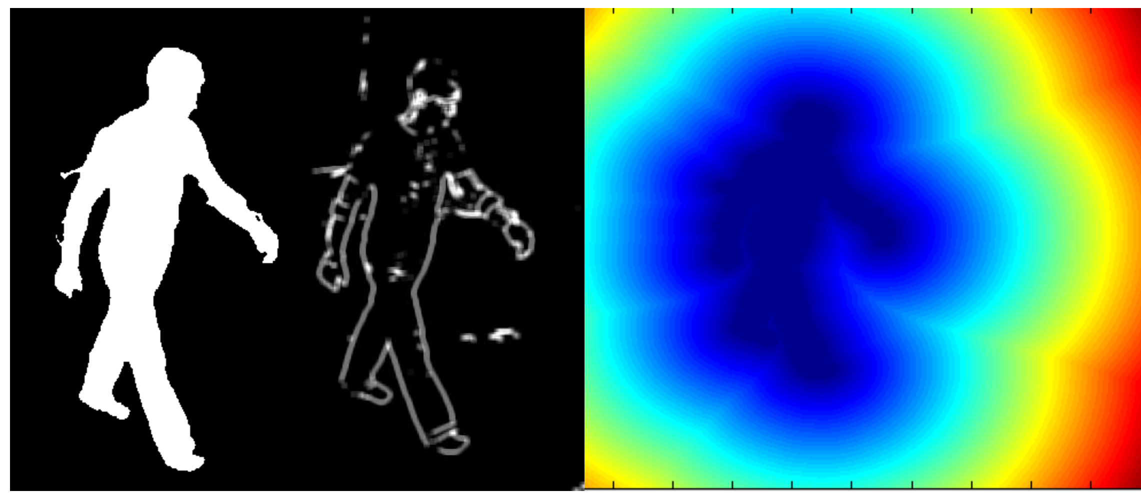


Image features

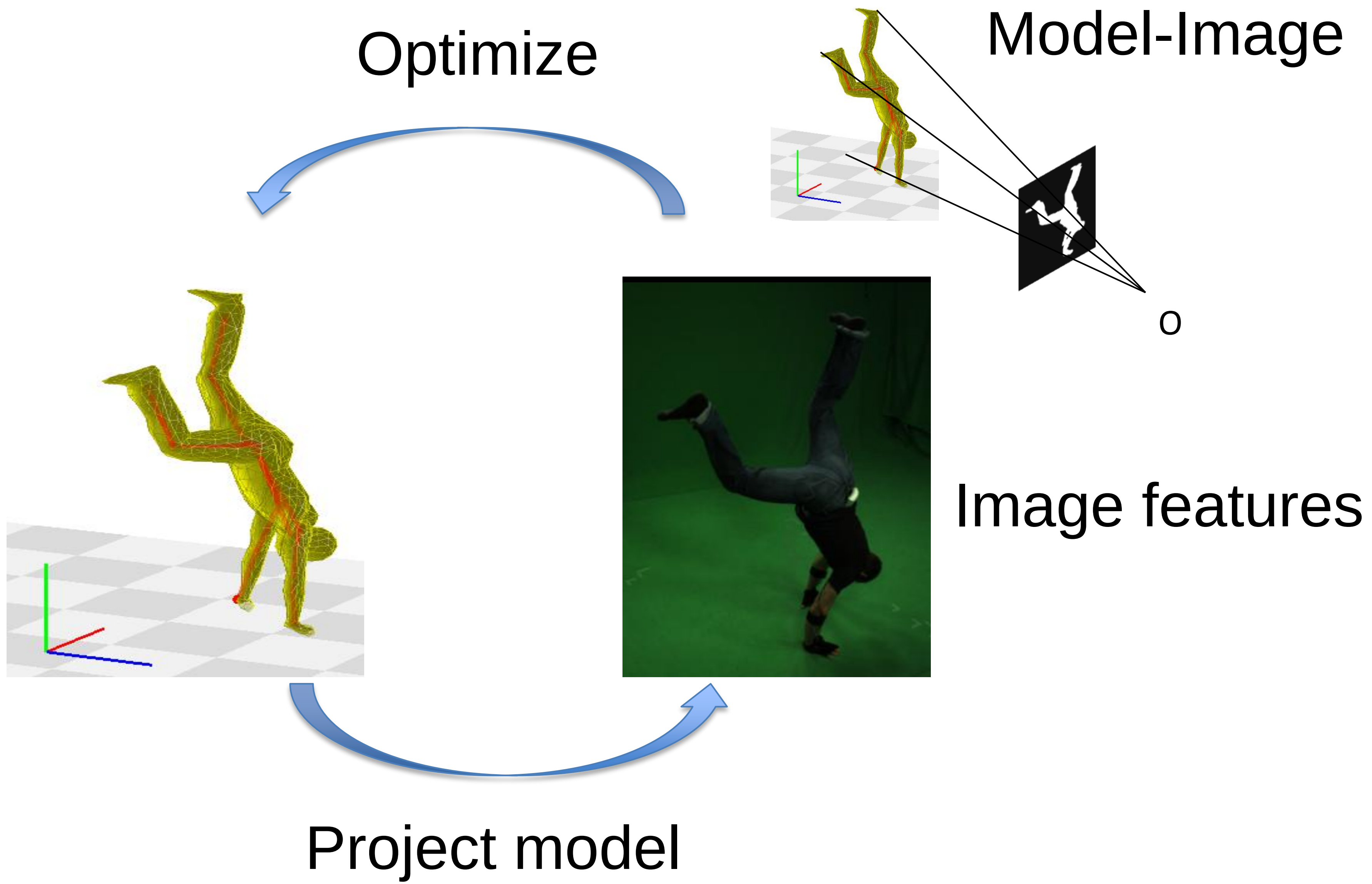
- Silhouettes
- Edges
- Distance transforms
- SIFT
- Optic flow
- Appearance
- …





Any feature that can be predicted from the model and is fast to compute

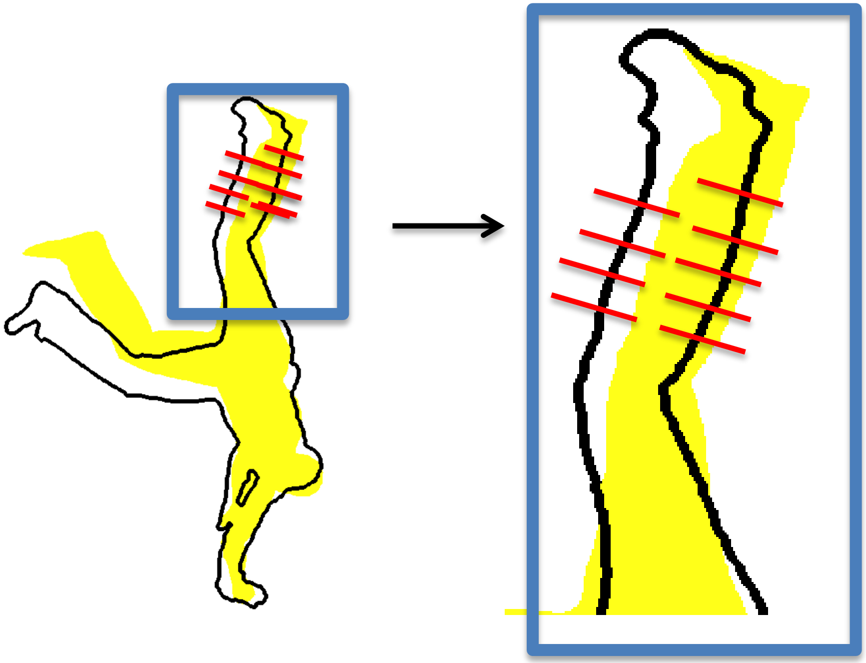
Optimization



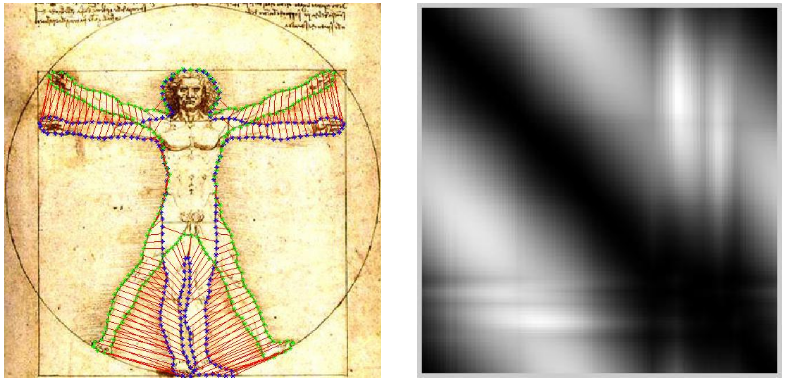
Matching

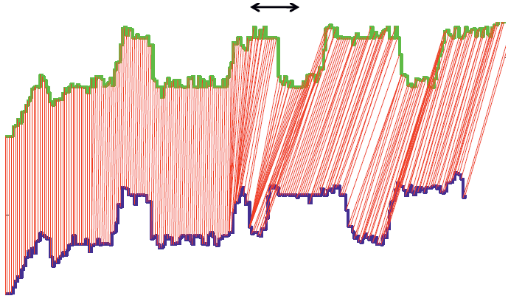
TIPS:
1) Match image to model and model to image
2) Careful removing outliers

1) Look along normal model contour directions



2) Discretize and match





Least squares

$$e(\mathbf{x}_t) = \sum_i^N \mathbf{e}_i^2(\mathbf{x}_t) = \sum_i^N \|\tilde{\mathbf{r}}_i(\mathbf{x}_t) - \mathbf{r}_i\|^2$$

Model predictions ($\tilde{\mathbf{r}}_i(\mathbf{x}_t)$)

Image observations ($\mathbf{r}_i$)

Assuming Gaussian distribution it is equivalent to a MAP estimate

$$p(\mathbf{x}_t|\mathbf{y}_t) = \boxed{p(\mathbf{y}_t|\mathbf{x}_t)}\, p(\mathbf{x}_t) \propto \boxed{\exp\left(-\sum_i^N \mathbf{e}_i^2(\mathbf{y}_t^i|\mathbf{x}_t)\right)}\, p(\mathbf{x}_t)$$

Least squares

Express the problem in vector form

$$e(\mathbf{x}_t) = \mathbf{e}^T \mathbf{e} \qquad \mathbf{e} \in \mathbb{R}^{2N}$$

$$\mathbf{e} = (\mathbf{e}_1^T, \mathbf{e}_2^T, \ldots, \mathbf{e}_N^T)$$

$$e(\mathbf{x}_t) = \begin{bmatrix} \Delta r_{1,x} & \Delta r_{1,y} & \ldots & \Delta r_{N,x} & \Delta r_{N,y} \end{bmatrix} \begin{bmatrix} \Delta r_{1,x} \\ \Delta r_{1,y} \\ \vdots \\ \vdots \\ \Delta r_{N,x} \\ \Delta r_{N,y} \end{bmatrix}$$

Residual for match 1

Local Optimization

$$\Delta\mathbf{x} = \arg\min_{\Delta\mathbf{x}} \frac{1}{2}\,\mathbf{e}^T(\mathbf{x}_t + \Delta\mathbf{x})\;\mathbf{e}(\mathbf{x}_t + \Delta\mathbf{x})$$

$$= \arg\min_{\Delta\mathbf{x}} \frac{1}{2}(\mathbf{e} + \mathbf{J}_t\Delta\mathbf{x})^T(\mathbf{e} + \mathbf{J}_t\Delta\mathbf{x})$$

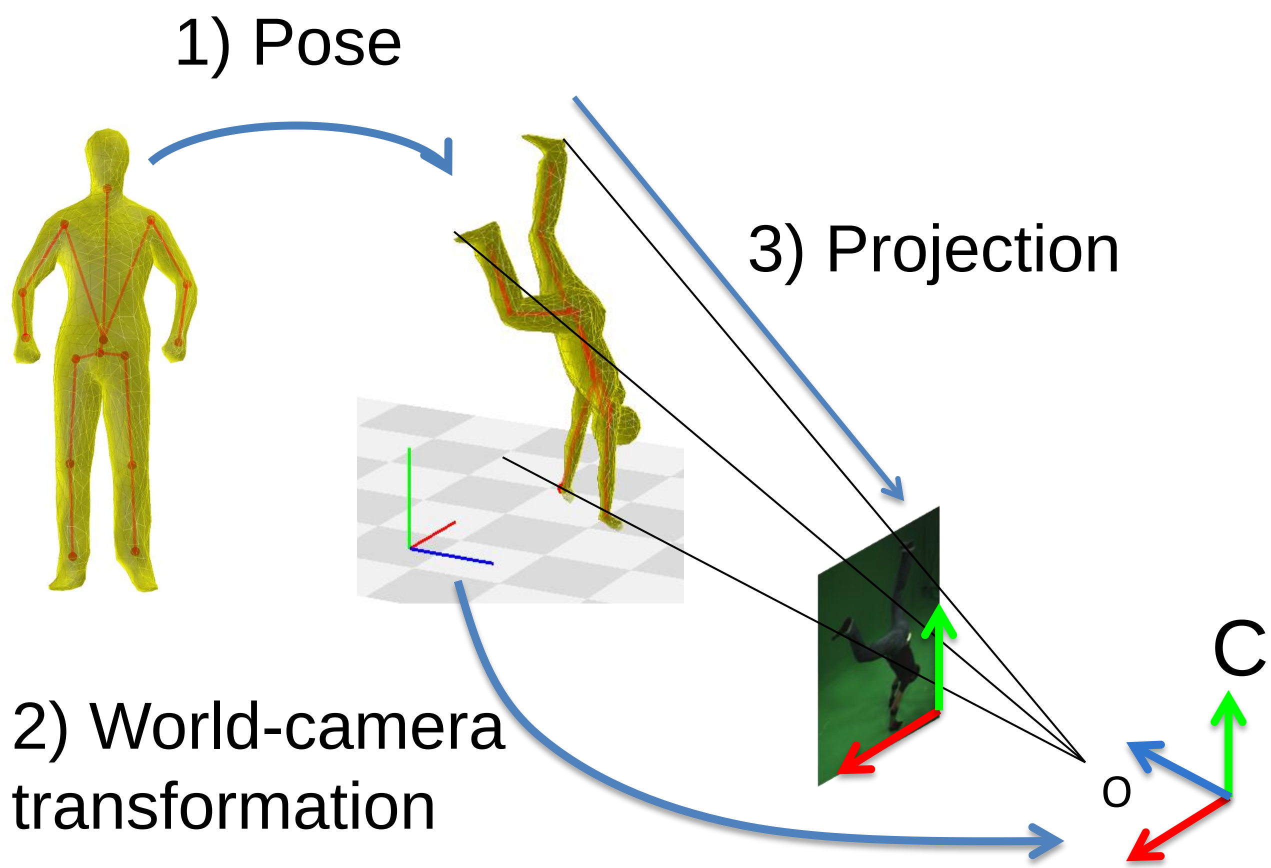
$$= \arg\min_{\Delta\mathbf{x}} \frac{1}{2}\mathbf{e}^T\mathbf{e} + \Delta\mathbf{x}^T\underbrace{\mathbf{J}_t^T\mathbf{e}}_{\text{Gradient}} + \frac{1}{2}\Delta\mathbf{x}^T\underbrace{\mathbf{J}_t^T\mathbf{J}_t}_{\text{\textasciitilde Hessian}}\Delta\mathbf{x}$$

$$\Delta\mathbf{x} = -\left(\mathbf{J}_t^T\mathbf{J}_t + \mu\mathbf{I}\right)^{-1}\mathbf{J}_t^T\mathbf{e}$$

Take a step in that direction

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta\mathbf{x}$$

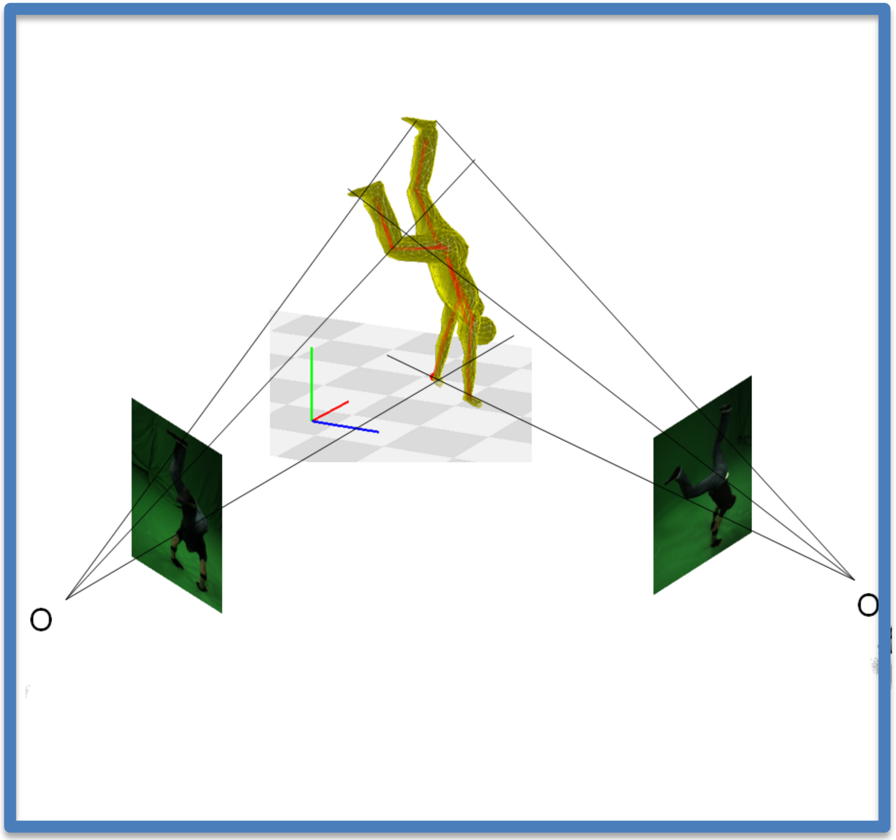
Jacobian



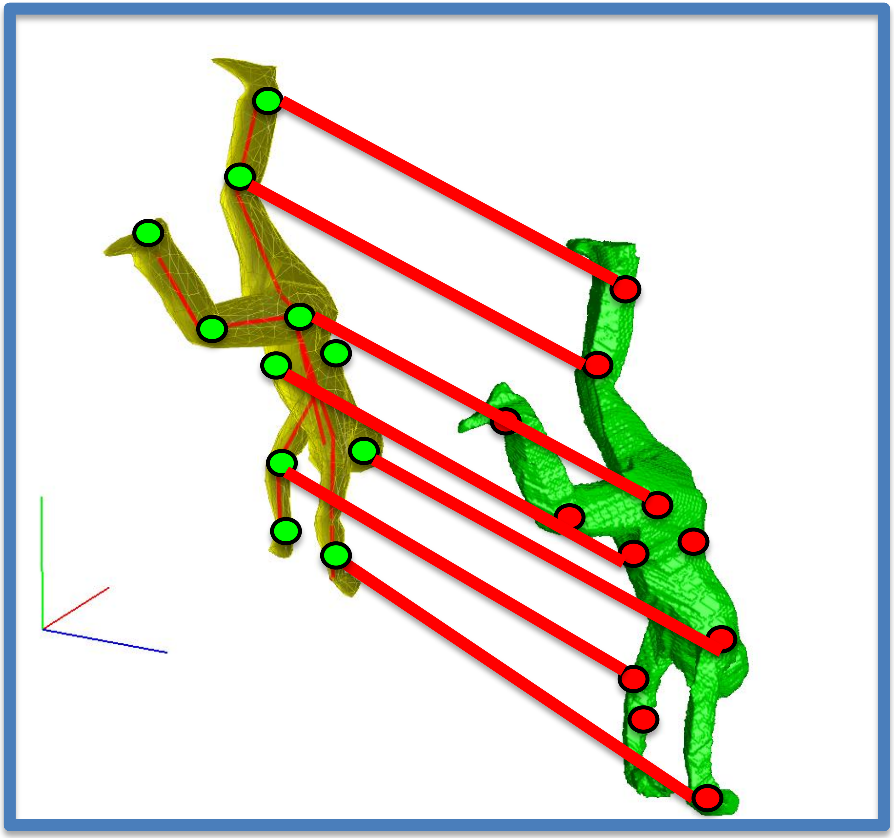
$$\mathbf{J}_{t,i} = \frac{\Delta \tilde{\mathbf{r}}_i}{\Delta \mathbf{x}_t} = \frac{\Delta \tilde{\mathbf{r}}_i}{\Delta \mathbf{p}_c} \cdot \frac{\Delta \mathbf{p}_c}{\Delta \mathbf{p}_s} \cdot \frac{\Delta \mathbf{p}_s}{\Delta \mathbf{x}_t} = \mathbf{J}_p \, \mathbf{R}_{cs} \, \mathbf{J}_{\mathbf{x}}(\mathbf{p}_s^i)$$

3 2 1

Other likelihoods

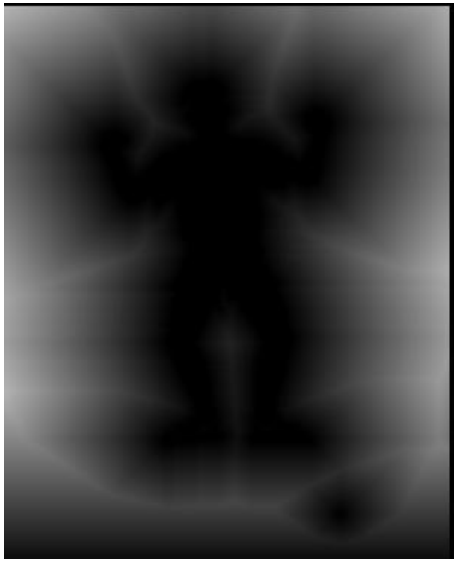


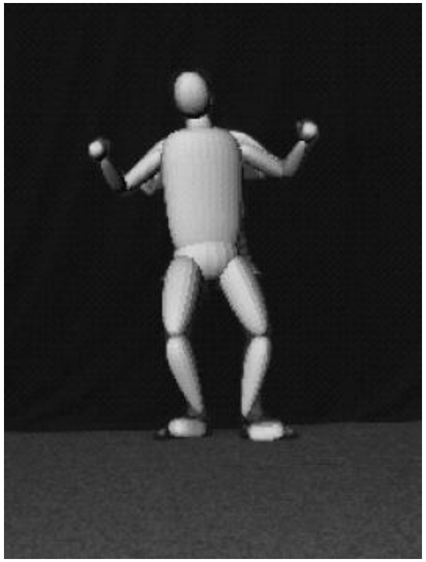
2D-3D error
point-to-line distance



3D-3D error
point-to-point distance

Distance transforms





inconsistent consistent

1) Push model inside silhouette
2) Force the model to explain the image

Distance transform + overlapp term

Sminchisescu F & G 2001

Region-based

Rosenhahn et.al.

Use model as **region mask** Q that separates foreground from background

$$e(\mathbf{x}) = -\int_{\Omega} Q(\mathbf{x}, \mathbf{r}) \log p_1 + (1 - Q(\mathbf{x}, \mathbf{r})) \log p_2) d\mathbf{r}$$

Optical flow

Bregler and Malik

Parameterize **flow** with human motion model

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} - I_t = 0,$$

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \cdot \Pr_c(\Delta \mathbf{p}_s) - I_t = 0,$$

Local optimization

- ✓ It is fast and accurate

- ✗ Prone to local minima
- ✗ Requires initialization
- ✗ Matching cost is ambiguous
- ✗ Single hypothesis propagated

Overview

1) Kinematic parameterization

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

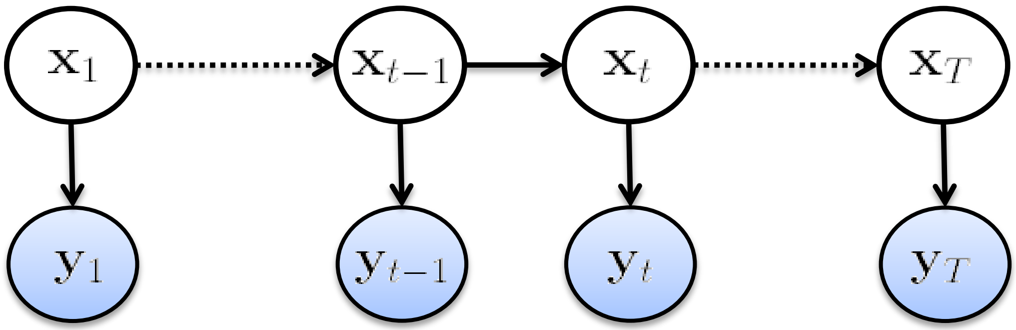
2) Subject model

- Geometric primitives
- Detailed Body Scans
- Human Shape models

3) Inference

- Observation likelihood
- Local optimization
- Particle Based inference

Particle Filter



First order Markov process

$\mathbf{y}_t = (\mathbf{r}_1 \ldots \mathbf{r}_N)$ Image observations at t

$\mathbf{x}_t$ State space, pose parameters at t

- Once I know $\mathbf{x}_{t-1}$, $\mathbf{x}_t$ is independent on previous measurements

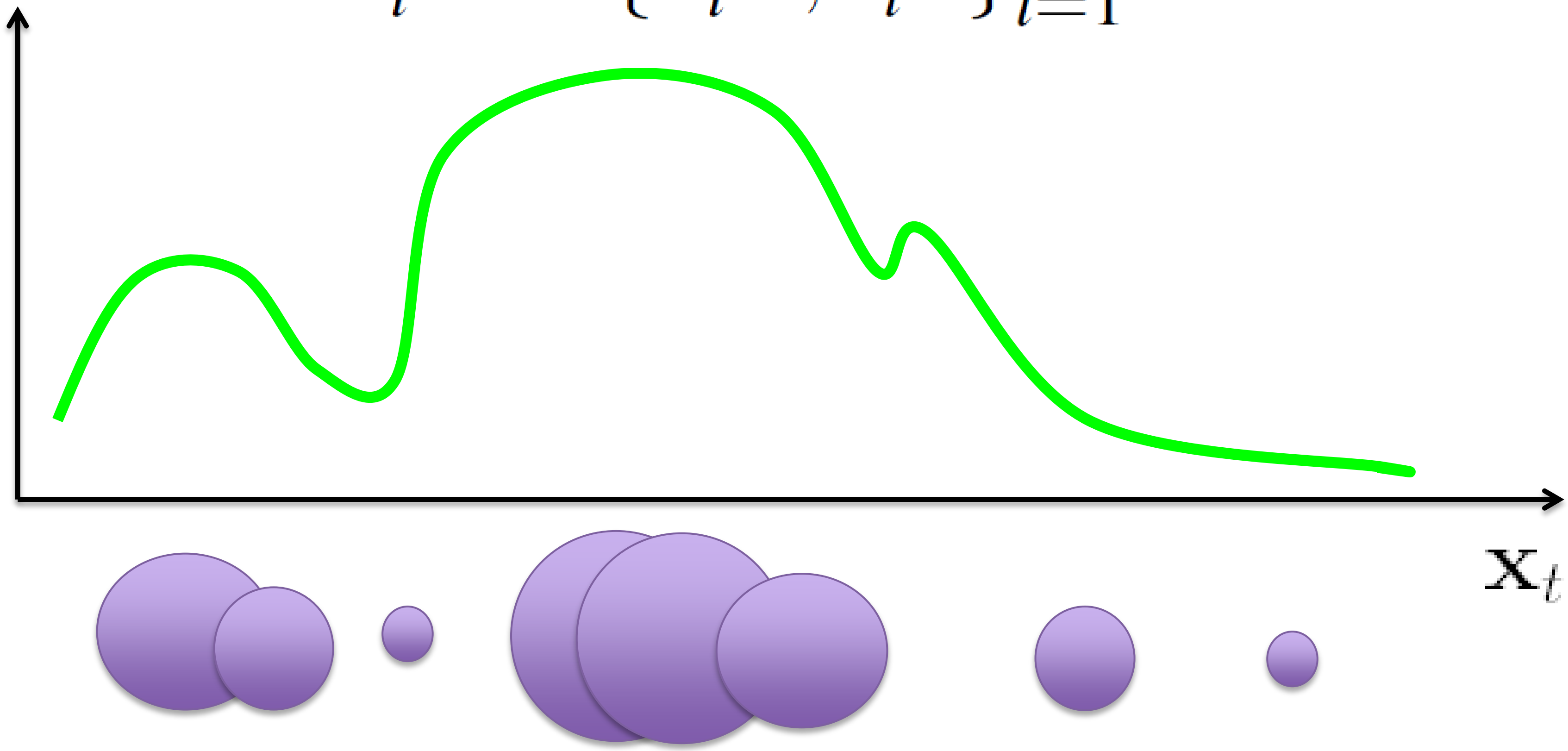
$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$$

- Once I know the state, the new measurement becomes independent on the others

$$p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{y}_{1:t-1}) = p(\mathbf{y}_t|\mathbf{x}_t)$$

Particle Filter

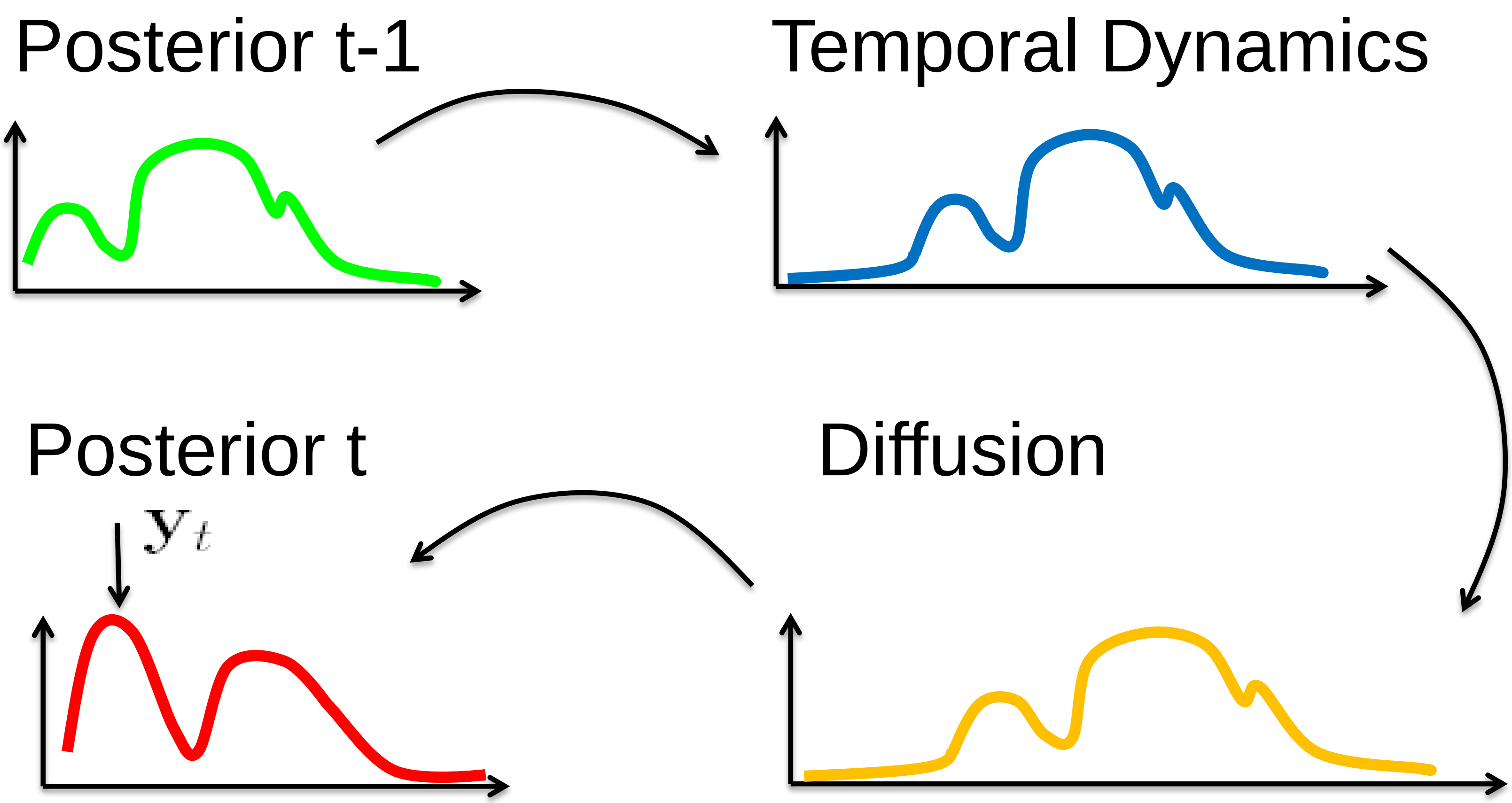
$$\mathscr{P}_t^+ := \{\pi_t^{(i)}, \mathbf{x}_t^{(i)}\}_{i=1}^N$$



Distribution approximated with a set of **weighted samples**

Particle Filter

$$\underline{p(\mathbf{x}_t|\mathbf{y}_{1:t})} = p(\mathbf{y}_t|\mathbf{x}_t) \boxed{\int p(\mathbf{x}_t|\mathbf{x}_{t-1})\underline{p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})}d\mathbf{x}_{t-1}}$$



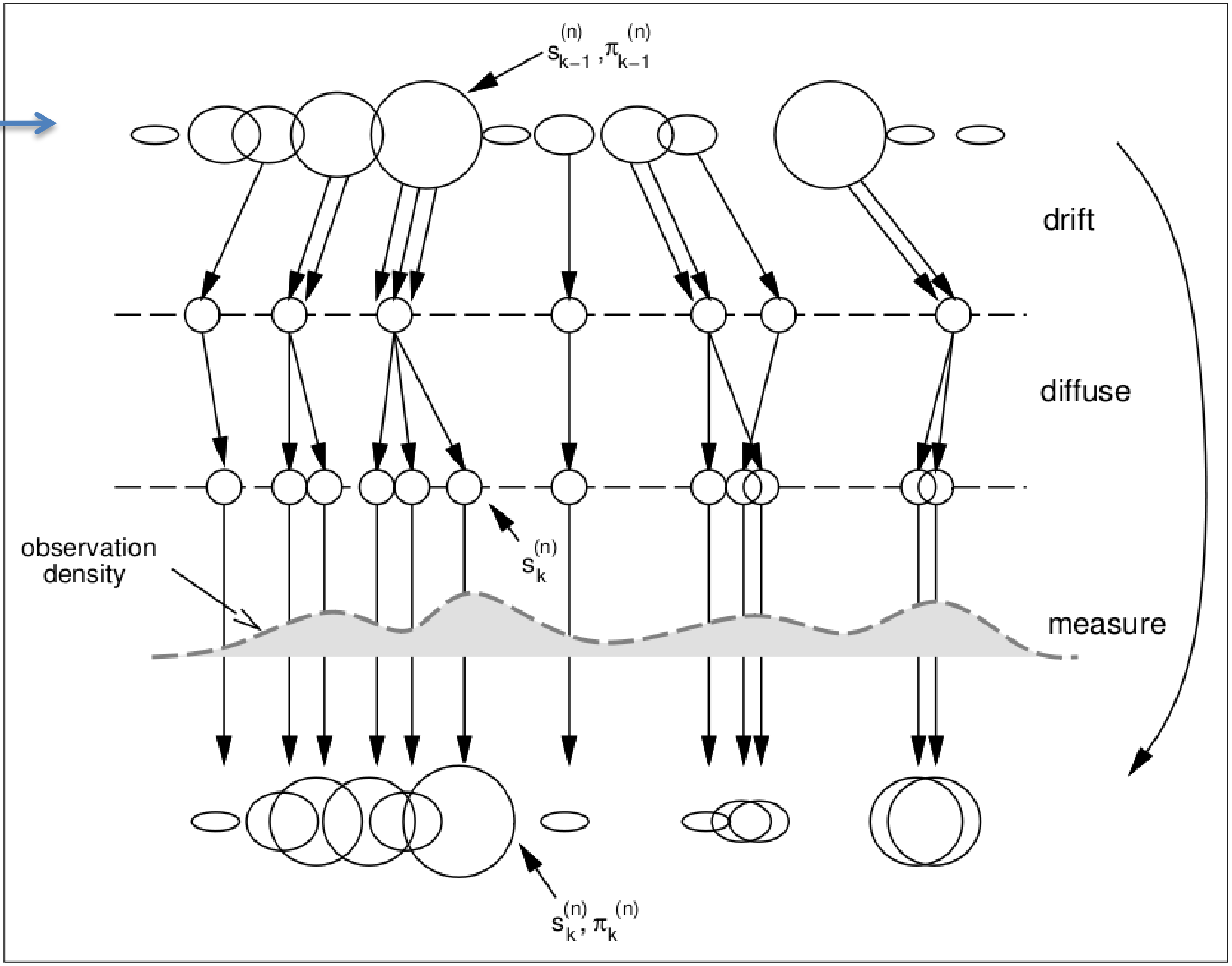
Particle Filter

$p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ →

$p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ →

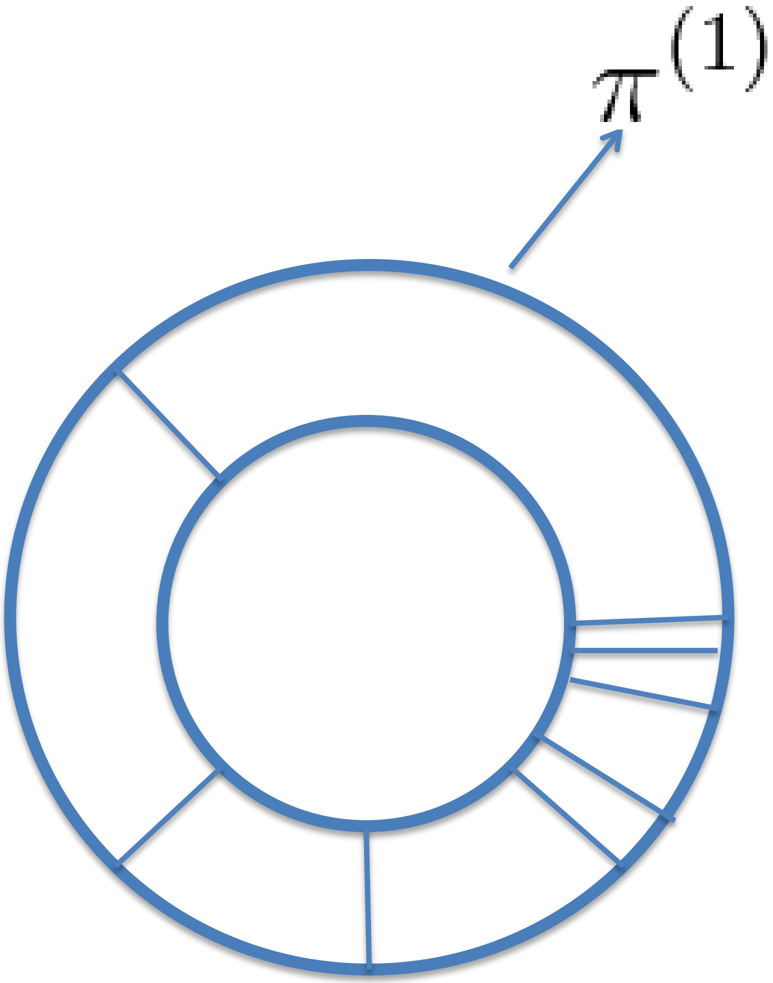
$p(\mathbf{y}_t|\mathbf{x}_t)$ →

$p(\mathbf{x}_t|\mathbf{y}_{1:t})$ →



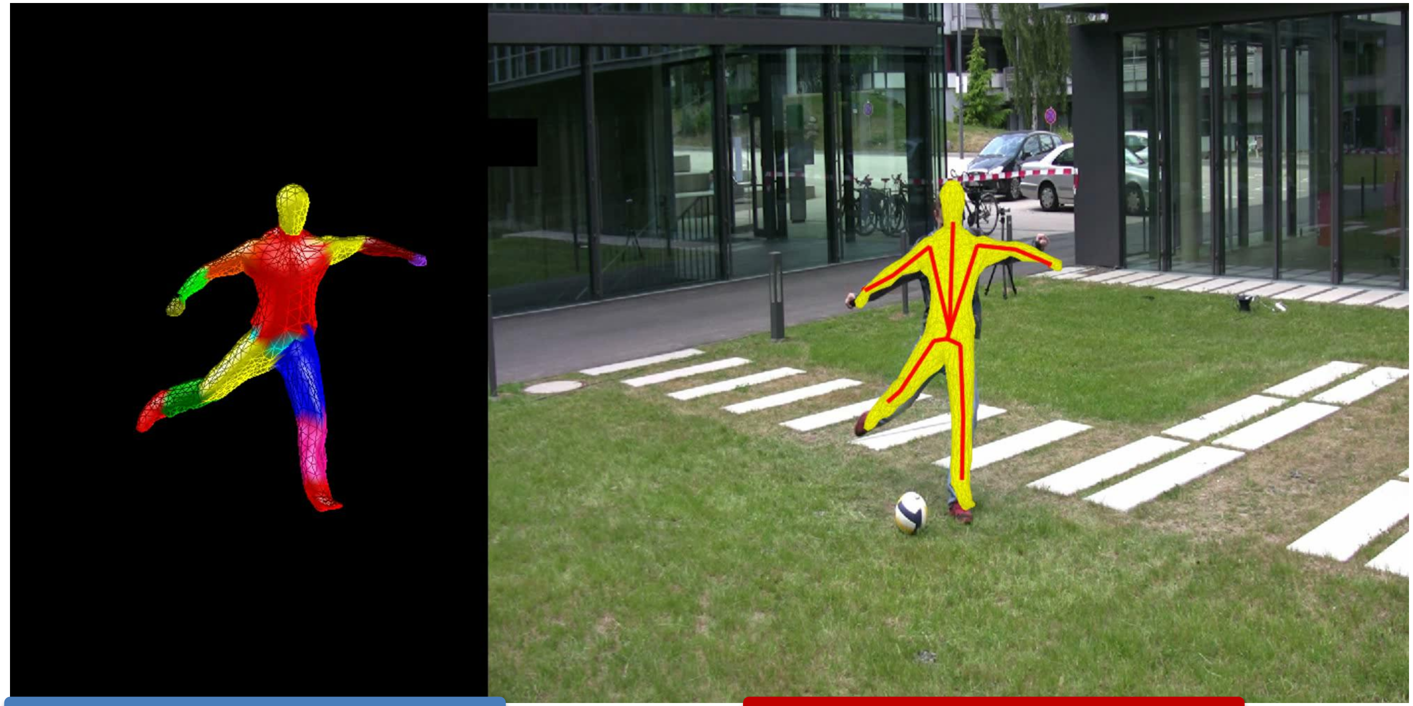
Condensation, Isaard and Blake 1996

$$\mathscr{P}_t^+ := \{\pi_t^{(i)}, \mathbf{x}_t^{(i)}\}_{i=1}^N$$



Resample with probability equal to the weights

Sampling



$p(\mathbf{x}_t | \mathbf{y}_{1:t-1})$

sample

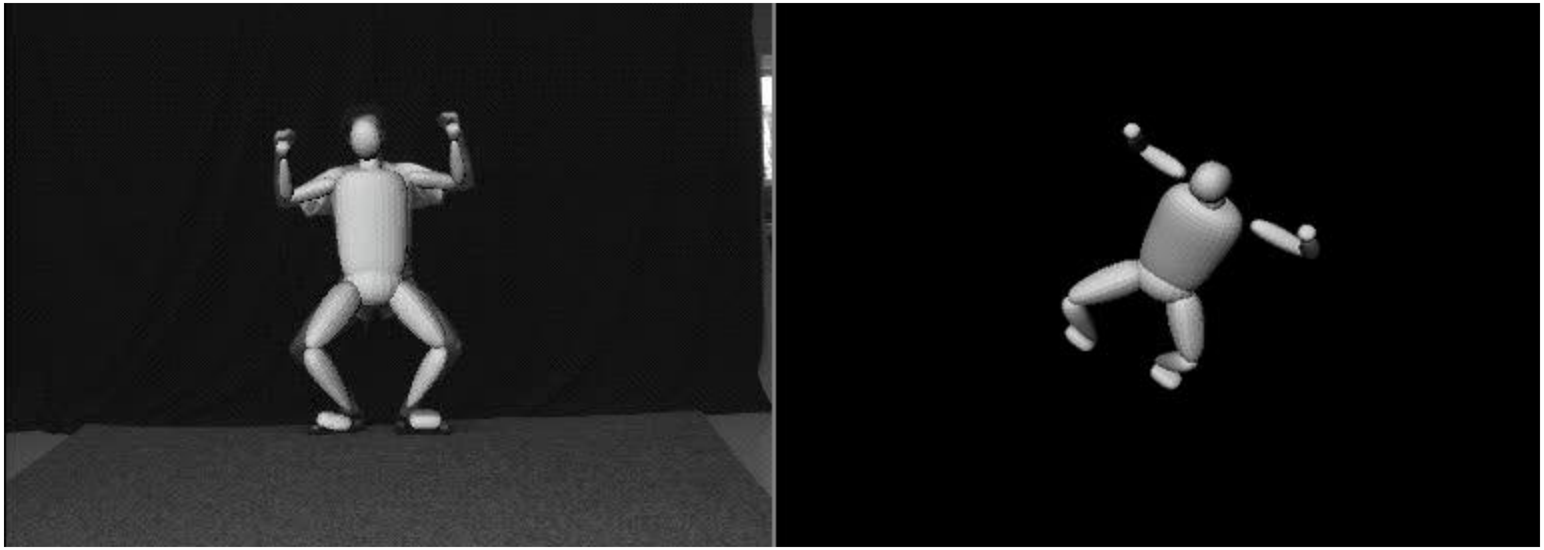
$p(\mathbf{y}_t | \mathbf{x}_t)$

weight

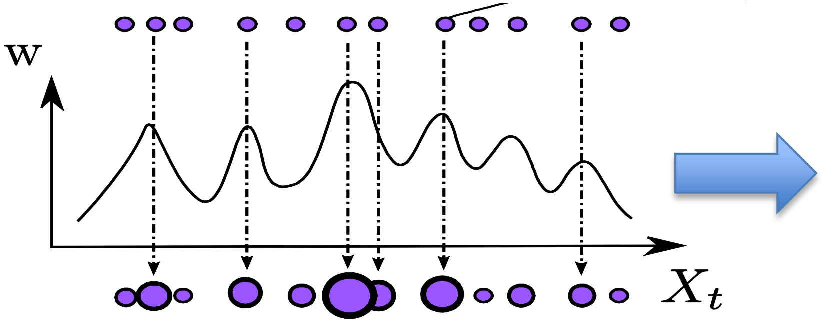
weight → $w(\mathbf{y}_t, \mathbf{x}_t = \mathbf{x}_t^{(i)}) = \exp\left(-e\left(\mathbf{x}_t^{(i)}\right)\right)$

Problems

Observation likelihood is highly multimodal !



Video from Sminchisescu and Triggs

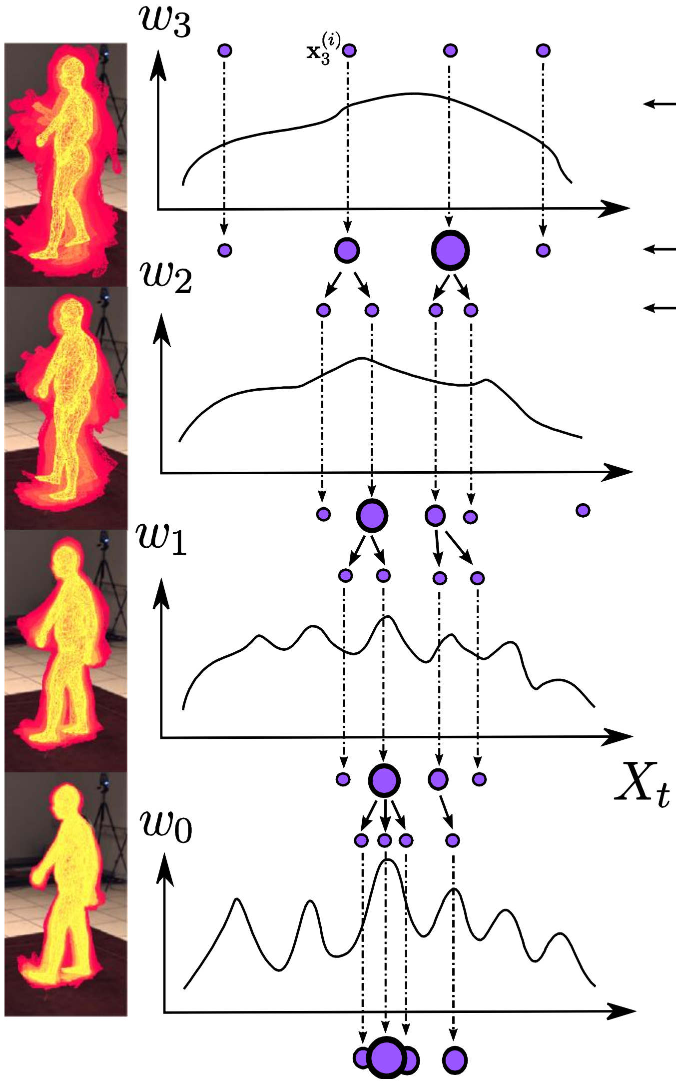


1) Multiple optima
2) Huge search space

Annealed Particle Filter

Iteratively evaluate smooth versions of $p(\mathbf{y}_t|\mathbf{x}_t)$

- ✓ Particles reduced by a factor >10
- ✓ Less prone to local optima
- ✗ Not as robust as Bayesian
- ✗ Still computationaly expensive



Deutscher et.al.
Gall et.al.

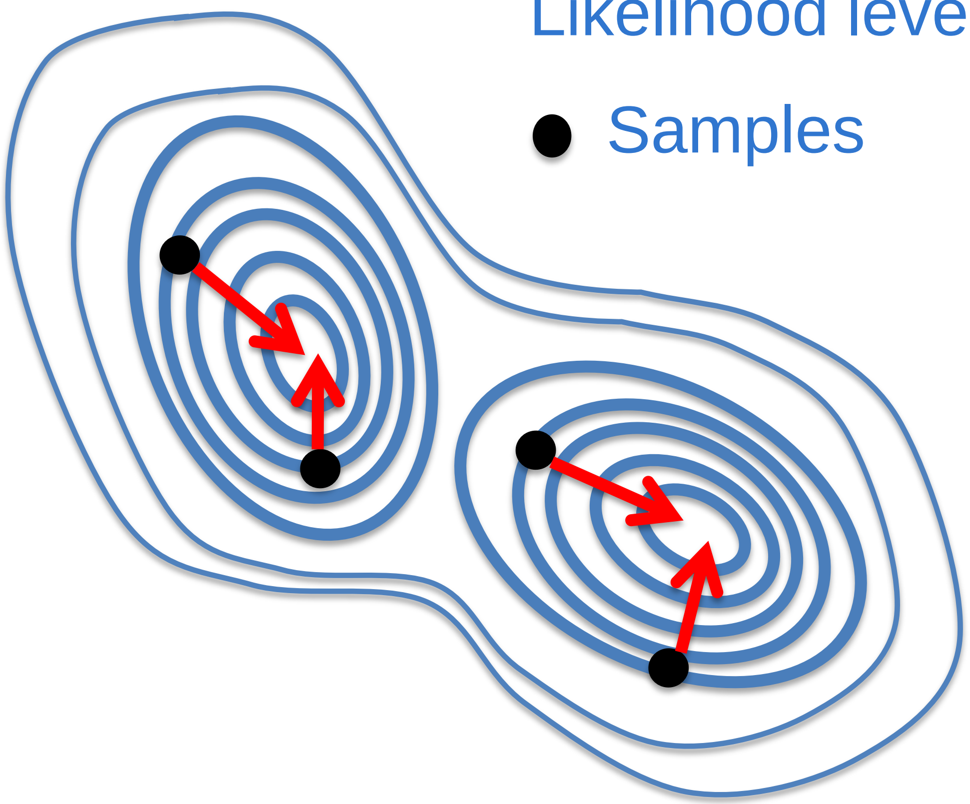
Hybrid MCMC

Localy optimize every sample of MCMC

Cho and Fleet

Likelihood levelsets

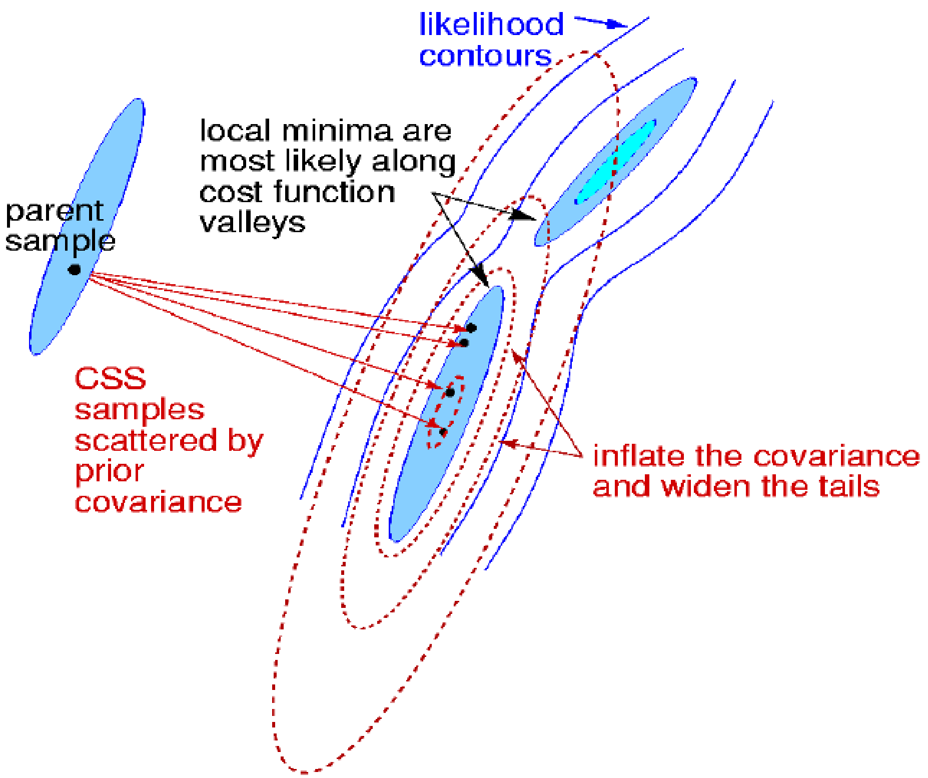
● Samples



Covariance Scaled Sampling

Scatter particles along cost function valley

Sminchisescu and Triggs



- ✓ Explore high dimensional space more efficiently
- ✓ Dedicates some particles to explore globaly

Discussion

Generative modeling:

- Need to model Kinematics
- Need to model Shape
- Need to model Observation Likelihood
- Texture
- Ilumination
- ufff lots of work so …

IS THIS THE END OF GENERATIVE ?

End of Generative ?

Well, depends on the application…

✗ In totaly uncontrolled scenarios will never work!

✓ But the **accuracy** is still higher and they **generalize** to complex motions better than discriminative approaches

✓ Useful as refinement stage coupled with discriminative initialization