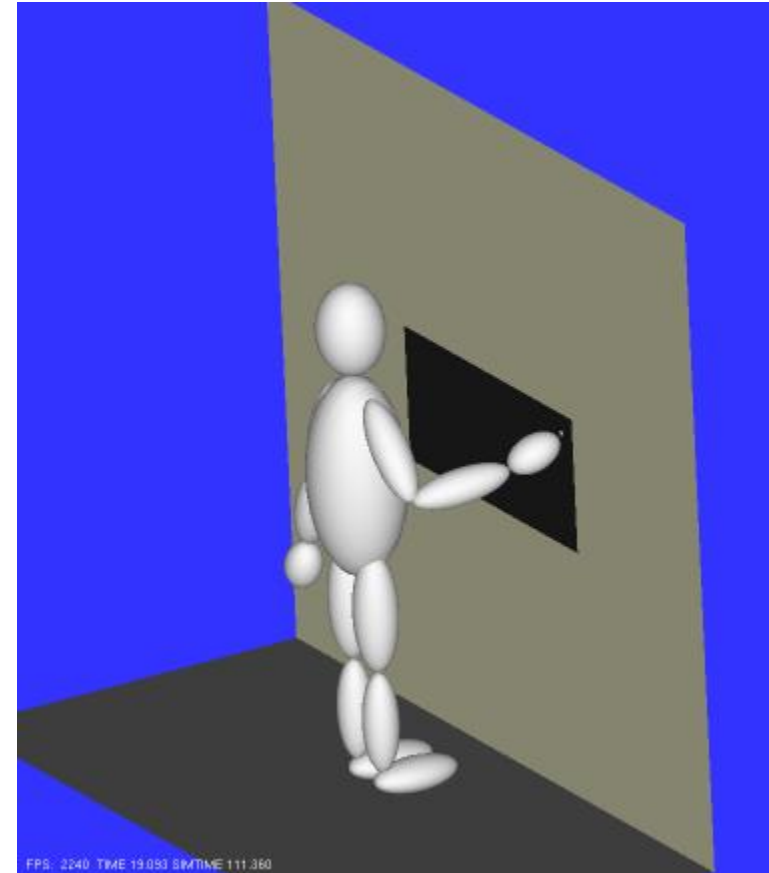


Inverse Kinematics & Motion Capturing

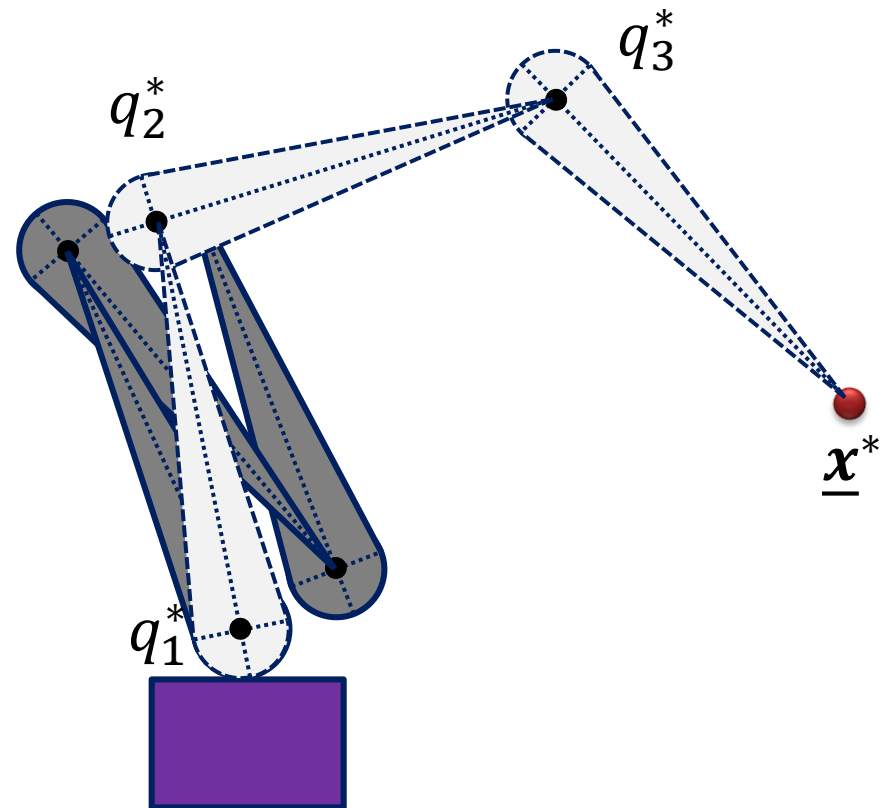
- ◆ Inverse Kinematics Problem
- ◆ Cyclic Coordinate Descent
- ◆ Unconstrained IK
- ◆ Constrained IK
- ◆ Motion Capturing
- ◆ Skeleton Fitting



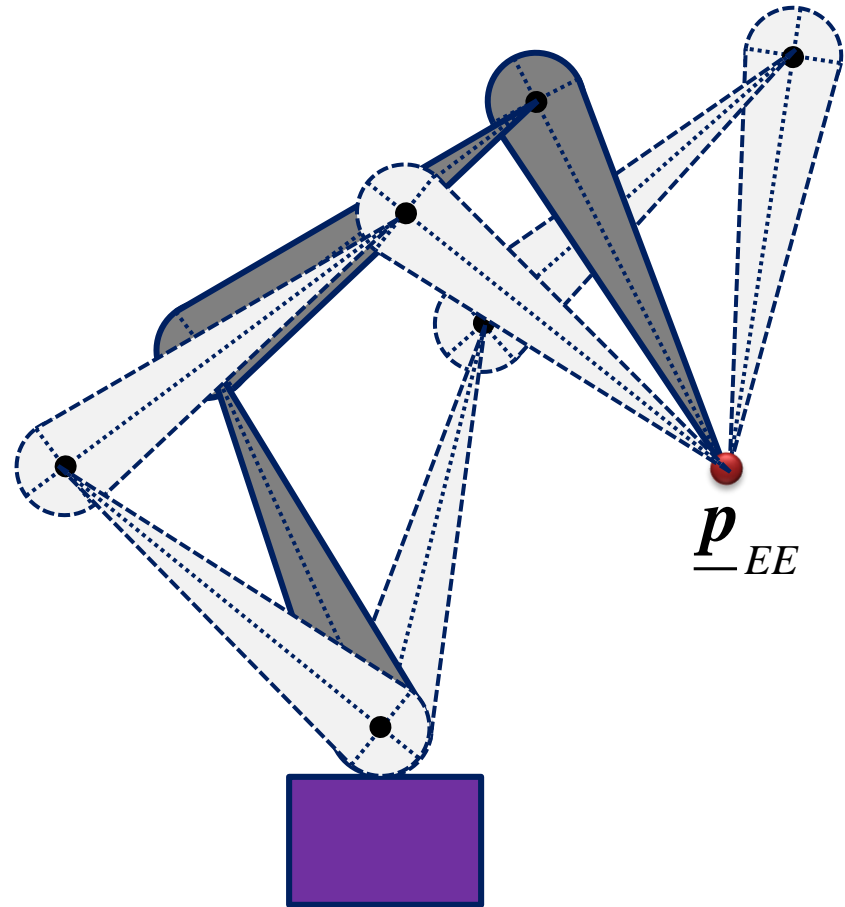


INVERSE KINEMATICS PROBLEM

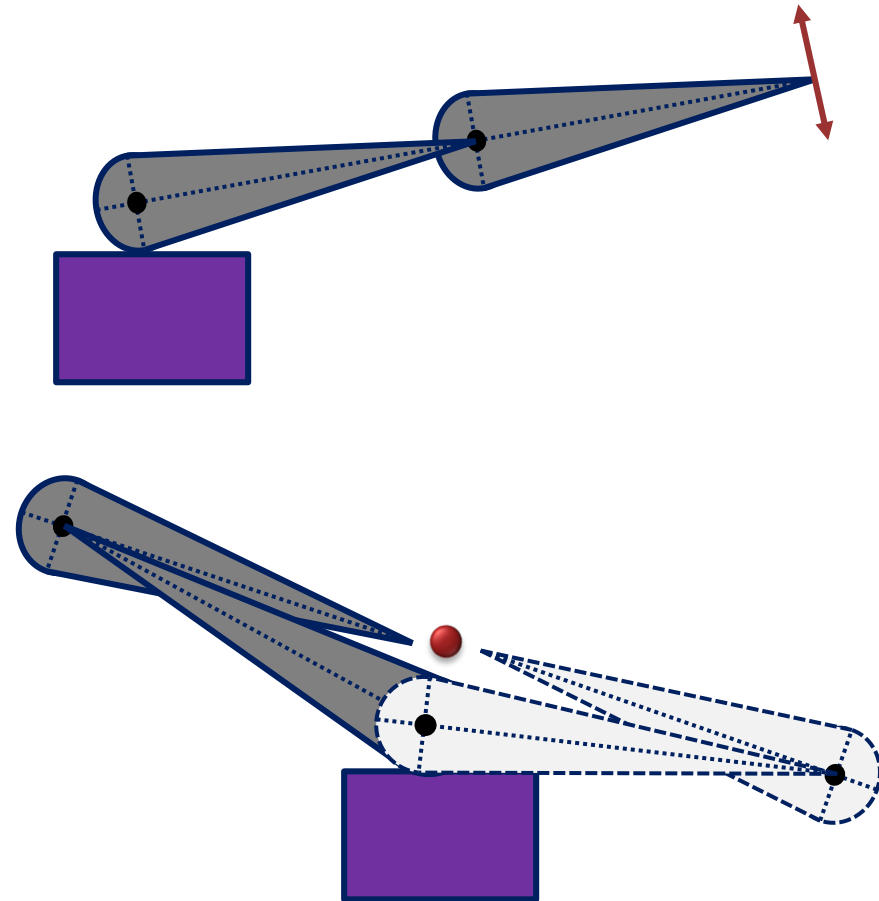
- Given a kinematic chain with forward kinematics $\underline{p}_{EE} = \underline{f}(q_j), \overrightarrow{\omega}_{EE} = \underline{F}(q_j)$ depending on **generalized coordinates** q_j and a target pose $[\underline{x}^*, \overrightarrow{\omega}^*]$, the inverse kinematics problem solves $\underline{p}^* = \underline{f}(q_j^*) \wedge \overrightarrow{\omega}^* = \underline{F}(q_j^*)$ for the q_j^*
- The parameters q_j are also called **state vector**.
- The parameters $[\underline{p}, \overrightarrow{\omega}]$ of the end effector pose are called the **dependent variables**



- ◆ The number of generalized coordinates give the **degrees of freedom (DOF)**
- ◆ In most cases the DOFs does not match the number of dependent variables and IK becomes ill-posed (as in example on the right) or unsolvable
- ◆ The inverse kinematics problem is often posed in form of a least squares energy minimization as detailed on slides 20 & 21.



- ◆ In so called **degenerate or singular configurations** the end effector loses one or several degrees of freedom
- ◆ Close to singular positions the distance in the state space can become extremely large compared to the distance in world space leading to oscillations during IK



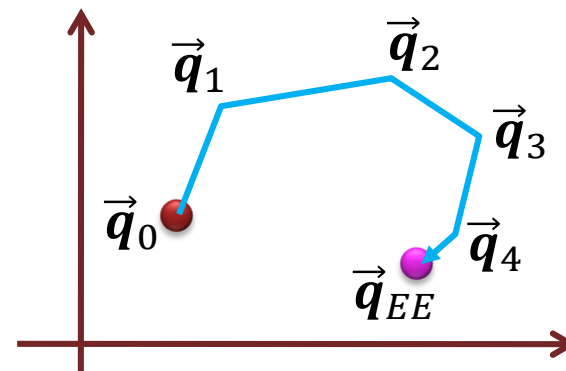
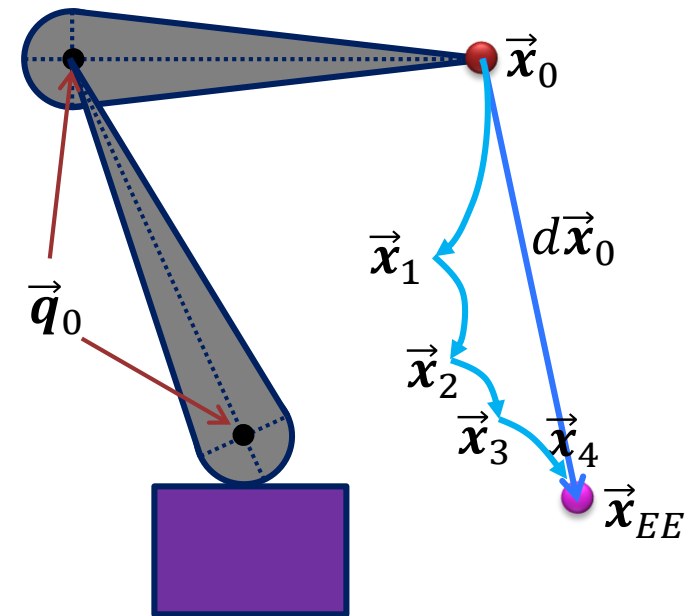


Constraints

- ◆ for realistic behavior it is necessary to consider different constraints:
- ◆ **collision constraints** to avoid self-collisions and collisions with environment. These can only locally be captured in formula.
- ◆ **joint angle constraints** given by the limitations of the joints and written in the form $l_j \leq q_j \leq u_j$
- ◆ **position constraints** to restrict the movement of the end effector. Typically point, line or plane constraints.
- ◆ **orientation constraints** to restrict the orientation of the end effector, for example in case of special demands for grasping.

Overall strategy

- Most solvers for non linear optimization problems are iterative and start with some initial guess \vec{q}_0 for the parameter vector. This can be
 - the default state
 - the state of the previous time step in an animation
 - the result of a previous optimization phase
- At the current guess some descent direction $d\vec{q}_k$ is found in parameter space
- A step size h_k is estimated from the energy function
- Steps $\vec{q}_{k+1} = \vec{q}_k + h_k d\vec{q}_k$ are taken until no further improvement in energy is possible.



- Each joint in a kinematic chain has a relative transformation ${}^{i-1}\tilde{\mathbf{T}}_i(q_{i1}, \dots, q_{in_i})$ with n_i parameters
- Gathering all $n = \sum_i n_i$ parameters in the parameter vector $\vec{\mathbf{q}}$ the chain transform is

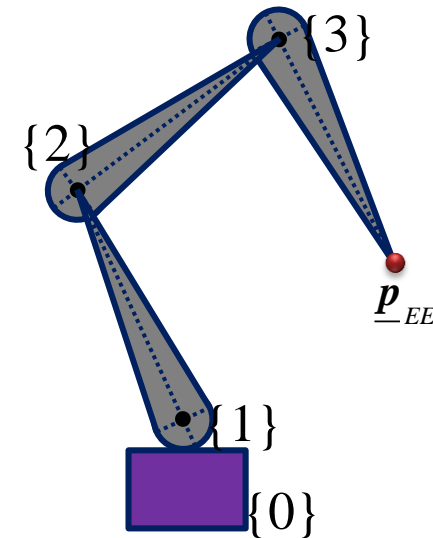
$${}^0\tilde{\mathbf{T}}_N(\vec{\mathbf{q}}) = {}^0\tilde{\mathbf{T}}_1 {}^1\tilde{\mathbf{T}}_2 \cdot \dots \cdot {}^{N-1}\tilde{\mathbf{T}}_N$$

- Transformations are in homogenous notation

$${}^0\tilde{\mathbf{T}}_N = \begin{pmatrix} {}^0\mathbf{R}_N & {}^0\vec{\mathbf{t}}_N \\ \vec{\mathbf{0}}^T & 1 \end{pmatrix}$$

and contain rotation ${}^0\mathbf{R}_N$ and translation ${}^0\vec{\mathbf{t}}_N$.

- The columns of ${}^0\mathbf{R}_N$ define the coordinate axes of the end effector frame with respect to world coordinates
- If the end effector is in the origin of frame N , ${}^0\vec{\mathbf{t}}_N$ is the location of the end effector in world coordinates, in general we have $\underline{\mathbf{p}}_{EE}^0 = {}^0\mathbf{R}_N \underline{\mathbf{p}}_{EE}^N + {}^0\vec{\mathbf{t}}_N$





Euler Angle Formulation

- Representing the orientation as Euler angles, the pose \vec{x} of the end effector is a 6-dimensional vector

$$\vec{x}_{EE} = (\underline{p}_{EE}^0, \vec{\omega}_{EE}^0) = (x, y, z, \phi, \theta, \psi)^T \in \mathbf{R}^6$$

- The forward kinematics is described as a function \vec{f} that maps the n -dimensional parameter vector \vec{q} to a 6D pose:

$$\vec{x} = \vec{f}(\vec{q})$$

- The inverse kinematic problem for a given end effector pose is to find the parameters that best match the pose:

$$\vec{q}^* = \min_{\vec{q}} \arg E(\vec{q}; \vec{x}_{EE}) \quad \text{with} \quad E(\vec{q}; \vec{x}_{EE}) = \frac{1}{2} \|\vec{f}(\vec{q}) - \vec{x}_{EE}\|^2$$

- This is a non linear least squares problem. One can introduce weights w_i for the **residua** $r_i(\vec{q}) = f_i(\vec{q}) - x_{EE,i}$:

$$E(\vec{q}; \vec{x}_{EE}) = \frac{1}{2} \sum_i w_i r_i^2(\vec{q})$$

Orientation Matrix Formulation

$$\begin{aligned} \mathbf{o}_x(\vec{q})^T \mathbf{o}_{x,EE} &= \cos \psi_1 \\ \mathbf{o}_y(\vec{q})^T \mathbf{o}_{y,EE} &= \cos \psi_2 \\ \mathbf{o}_z(\vec{q})^T \mathbf{o}_{z,EE} &= \cos \psi_3 \end{aligned}$$

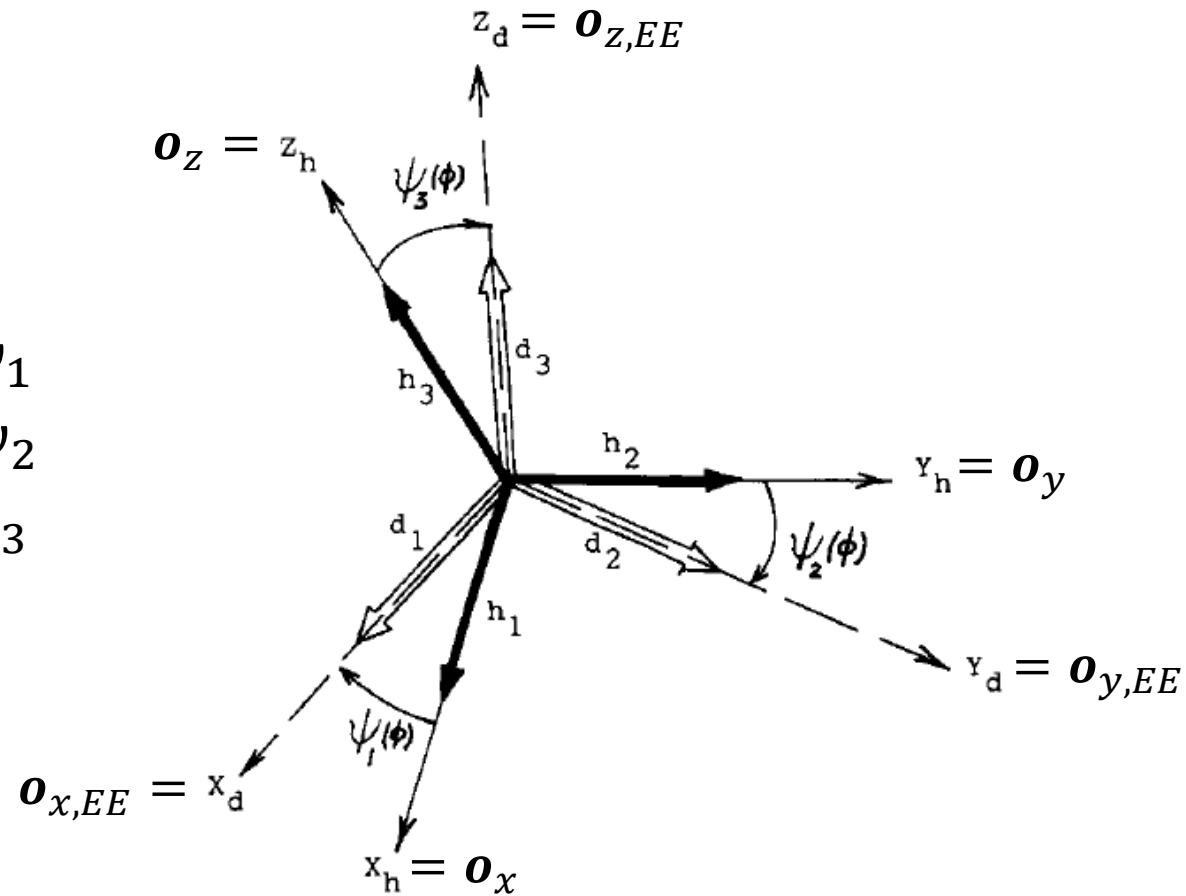


Fig. 5. Definition of the direction angles.



Orientation Matrix Formulation

- Using an orthonormal matrix $\mathbf{O} = (\mathbf{o}_x \mathbf{o}_y \mathbf{o}_z)$ to represent orientation, the pose $\vec{\mathbf{x}}$ of the end effector is 12D and defined as concatenation of position columns of \mathbf{O} :

$$\vec{\mathbf{x}}_{EE} = \left(\underline{\mathbf{p}}_{EE}^0 \quad \mathbf{o}_{x,EE}^T \quad \mathbf{o}_{y,EE}^T \quad \mathbf{o}_{z,EE}^T \right)^T \in \mathbb{R}^{12}$$

- The forward kinematics is described as a function $\vec{\mathbf{f}}$ that maps the n -dimensional parameter vector $\vec{\mathbf{q}}$ to a pose:

$$\vec{\mathbf{x}} = \vec{\mathbf{f}}(\vec{\mathbf{q}}), \quad \text{with } \vec{\mathbf{f}} = \left(\underline{\mathbf{p}}^T \quad \mathbf{o}_x^T \quad \mathbf{o}_y^T \quad \mathbf{o}_z^T \right)^T$$

- The inverse kinematic problem is again based on an energy function: $\vec{\mathbf{q}}^* = \text{minarg } E(\vec{\mathbf{q}}; \vec{\mathbf{x}}_{EE})$ with weights

$$E(\vec{\mathbf{q}}; \vec{\mathbf{x}}_{EE}) = \frac{1}{2} \left\| \underline{\mathbf{p}}(\vec{\mathbf{q}}) - \underline{\mathbf{p}}_{EE}^0 \right\|^2 + \frac{1}{2} \sum_{\alpha=x}^z w_{\alpha} \left(1 - \mathbf{o}_{\alpha}(\vec{\mathbf{q}})^T \mathbf{o}_{\alpha,EE} \right)^2$$

- The weights w_{α} can be used to blend out orientation constraints for individual axes



Quaternion Formulation

- With a normalized quaternion $\hat{q} = (s \ x \ y \ z)$ the pose is 7D:

$$\vec{x}_{EE} = \left(\underline{p}_{EE}^0{}^T \hat{q}_{EE}^0{}^T \right)^T \in \mathbf{R}^7, \quad \vec{f}(\vec{q}) = \left(\underline{p}^T \hat{q}^T \right)^T$$

- Care needs to be taken with the least squares energy

$$E(\vec{q}; \vec{x}_{EE}) = \frac{1}{2} \left\| \underline{p}(\vec{q}) - \underline{p}_{EE}^0 \right\|^2 + \frac{1}{2} w_q \left\| \hat{q}(\vec{q}) - \hat{q}_{EE}^0 \right\|^2$$

as \hat{q} and $-\hat{q}$ represent the same rotation.

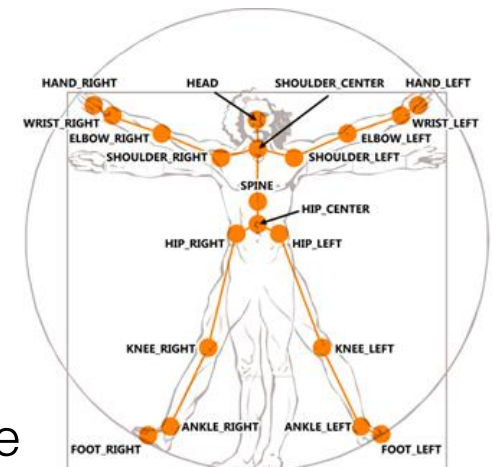
- Solution 1:

- during iterative optimization compute $\hat{q}(\vec{q}_i)$ for current \vec{q}_i .
- test if $\langle \hat{q}(\vec{q}_i), \hat{q}_{EE}^0 \rangle < 0$
- if yes replace \hat{q}_{EE}^0 with $-\hat{q}_{EE}^0$ in energy
- compute the next \vec{q}_{i+1} .

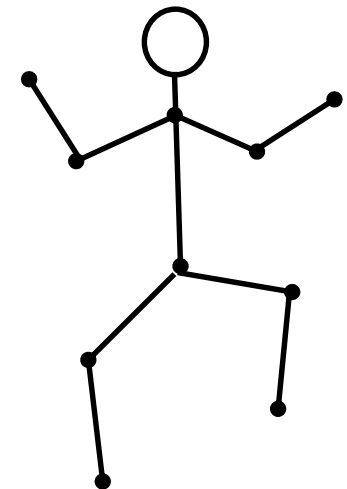
- Solution 2: use energy function that ignores sign:

$$E(\vec{q}; \vec{x}_{EE}) = \frac{1}{2} \left\| \underline{p}(\vec{q}) - \underline{p}_{EE}^0 \right\|^2 - w_q \langle \hat{q}(\vec{q}_i), \hat{q}_{EE}^0 \rangle^2$$

- multiple target locations are important in applications with skeletons like transforming point based motion capture data to skeleton parameters or when working with multiple end effectors
- a fixed base joint can only be used in the very special case of two constraints – one at a base node and one at an endeffector node
- otherwise we can alternately optimize
 - the rigid body transform of the root node by minimizing the squared sum of the end effector constraints with the Kabsch algorithm
 - the joint parameters by a joined IK problem minimizing an energy that sums over the squared endeffector-constraint distances where the endeffector locations are computed along kinematic chains from the fixed root node



- ◆ In an interactive editor one has to define kinematic chains based on user input. This can be done by adding **fixation points** (in the simplest case one at the root node)
- ◆ For a climbing figure we need several end effector constraints. This is discussed by Chris Hecker on his [inverse kinematics page](#)
- ◆ Dual quaternion IK was presented by Ben Kenwright in 2013: [Inverse Kinematics with Dual-Quaternions, Exponential-Maps, and Joint Limits](#)





CYCLIC COORDINATE DESCENT

Iterative Methods

- Iterative algorithms need an initial guess \mathbf{x}_0
 $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots \Rightarrow f_k = f(\mathbf{x}_k), \nabla f_k = \nabla f(\mathbf{x}_k), \nabla^2 f_k = \nabla^2 f(\mathbf{x}_k),$
- sequence chosen monoton with termination criterion:

$$f_0 \geq f_1 \geq f_2 \geq \dots \quad \|\nabla f_k\| < \varepsilon$$

Line Search Approach

choose search direction \mathbf{h}_k

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{h}_k$$

- choose step width α_k to minimize f along line:
 $\phi_k(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{h}_k),$
 $\alpha_k = \min_{\alpha > 0} \phi_k(\alpha)$

Trust Region Approach


- choose model m_k to approximate f inside trust region T around \mathbf{x}_k

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k,$$

$$\mathbf{h}_k = \min_{\mathbf{h} \in T} \arg m_k(\mathbf{x}_k + \mathbf{h})$$

- example:

$$m_k(\mathbf{x}_k + \mathbf{h}) = f_k + \nabla f_k^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \mathbf{B}_k \mathbf{h}$$

approximates Hessian




Coordinate Descent & Powell's Method

Coordinate Descent:

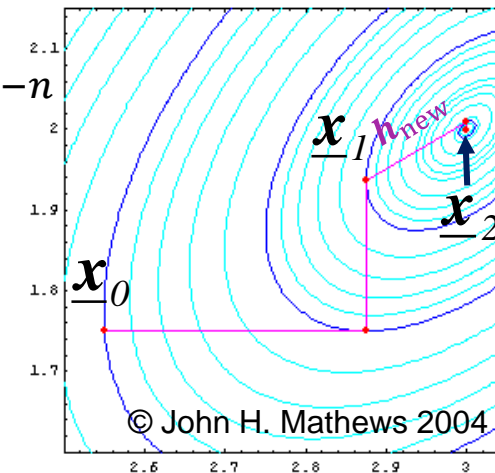
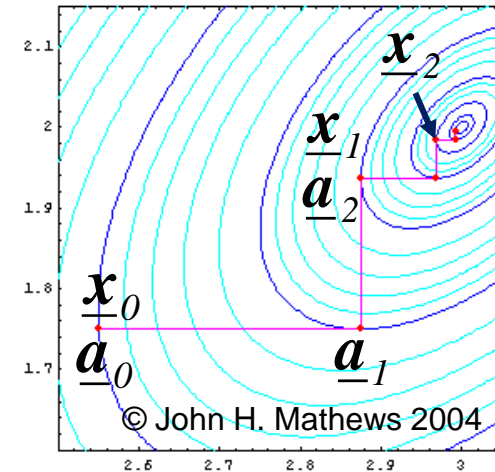
- Search directions are standard basis vectors \hat{e}_i
- line search along search directions
- Stop if no improvement in one complete cycle

Powell's method:

- start with standard basis as set of search directions and do line search
- after one cycle add new direction $\mathbf{h}_{\text{new}} = \mathbf{x}_i - \mathbf{x}_{i-n}$
- remove search direction with largest improvement (closest to new search direction \mathbf{h}_{new}) to avoid degenerated set of search directions

Both are derivative free methods!

- Another mentionable derivative-free alternative is the **Downhill-Simplex method** of Nelder and Mead





- In IK coordinate descent is often used to generate initial guess for secondary method with fast convergence rate close to optimum
- coordinate descent can be implemented efficiently as the line search problem can be solved analytically

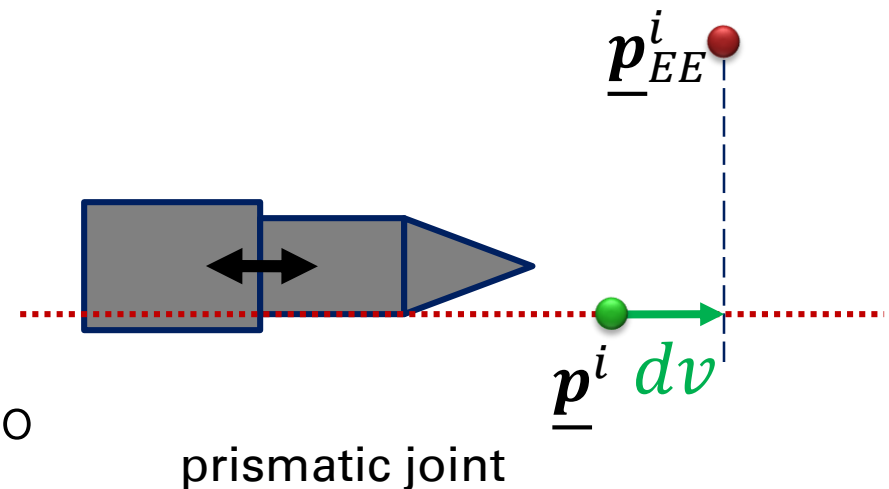
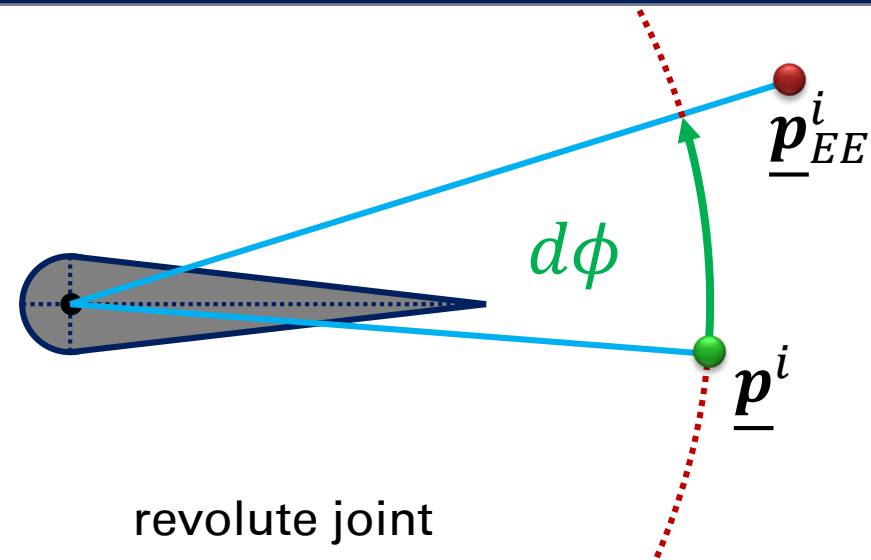
Cyclic Coordinate Descent Algorithm

- start with initial guess \vec{q}^0 for joint parameters
- for increasing $j = 1 \dots$ do
 - for each joint parameter i (typically from end effector to base)
 - reduce IK to parameter q_i^j
 - solve line search IK analytically to get q_i^{j+1}
- until convergence, i.e. $\|\vec{f}(\vec{q}^{j+1}) - \vec{f}(\vec{q}^j)\| < \varepsilon$

CCD – Line Search – Simplified



- First approach: only optimize end effector position $\underline{p} \rightarrow \underline{p}_{EE}$
- For given joint i transform positions into joint coordinate system $\Rightarrow \underline{p}^i, \underline{p}_{EE}^i$
- Update joint parameter q_i^j such that end effector comes as close to target position as possible
- **Revolute joint:** $q_i^j \equiv \phi$ rotation $d\phi$ makes position vector \underline{p}^i parallel to \underline{p}_{EE}^i
- **Prismatic joint:** $q_i^j \equiv v$ translation dv moves \underline{p}^i to the orthogonal projection of \underline{p}_{EE}^i onto z-direction.





- **Prismatic joints** do not affect orientation and optimization is done as in the case without orientation
- For a **revolute joint** i they reformulate energy minimization in local coordinates according to Orientation Matrix Formulation

$$E(d\phi) = \underbrace{\frac{1}{2} \left\| \underline{\mathbf{p}}^i(\phi_i + d\phi) - \underline{\mathbf{p}}_{EE}^i \right\|^2}_{E_1(d\phi)} + \underbrace{\frac{1}{2} \sum_{\alpha=x}^z w_{\alpha} \left(1 - \mathbf{o}_{\alpha}^i(\phi_i + d\phi)^T \mathbf{o}_{\alpha,EE}^i \right)^2}_{E_2(d\phi)}$$

into the maximization of a derived function $g = g_1 + g_2$

- **Position term:** $2E_1(d\phi) = \left\| \underline{\mathbf{p}}^i(\phi_i + d\phi) - \underline{\mathbf{p}}_{EE}^i \right\|^2$
 $= \left\| \underline{\mathbf{p}}^i(\phi_i + d\phi) \right\|^2 + \left\| \underline{\mathbf{p}}_{EE}^i \right\|^2 - 2 \left\langle \underline{\mathbf{p}}^i(\phi_i + d\phi), \underline{\mathbf{p}}_{EE}^i \right\rangle$

The first two terms on the right side are not affected by the joint rotation, such that we can choose $g_1(d\phi) := \left\langle \underline{\mathbf{p}}^i(\phi_i + d\phi), \underline{\mathbf{p}}_{EE}^i \right\rangle$ and it holds: $\min_{d\phi} E_1(d\phi) = \max_{d\phi} g_1(d\phi)$



◆ **Revolute Joint** transform $E = E_1 + E_2$ into $g = g_1 + g_2$

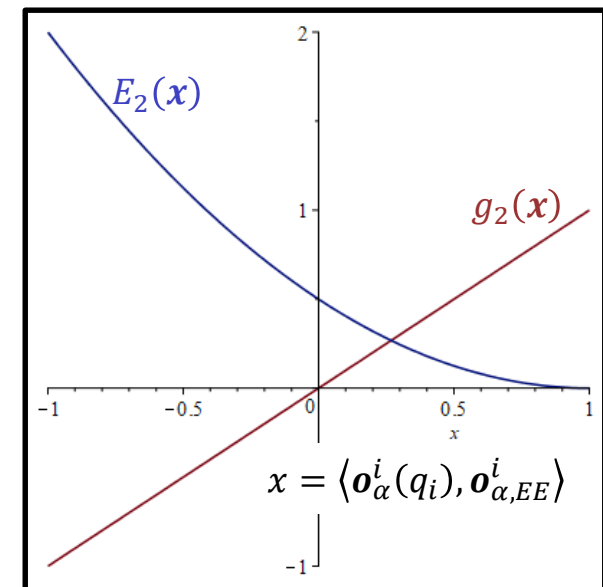
◆ **Position term:** $g_1(d\phi) = \langle \underline{p}^i(\phi_i + d\phi), \underline{p}_{EE}^i \rangle$

◆ **Orientation term:**

$$\begin{aligned} 2E_{2,\alpha}(d\phi) &= w_\alpha \left(1 - \mathbf{o}_\alpha^i(\phi_i + d\phi)^T \mathbf{o}_{\alpha,EE}^i \right)^2 \\ &= w_\alpha \left(1 - 2\langle \mathbf{o}_\alpha^i(\phi_i + d\phi), \mathbf{o}_{\alpha,EE}^i \rangle + \left(\mathbf{o}_\alpha^i(\phi_i + d\phi)^T \mathbf{o}_{\alpha,EE}^i \right)^2 \right) \end{aligned}$$

◆ Ignoring $\left(\mathbf{o}_\alpha^i(\phi_i + d\phi)^T \mathbf{o}_{\alpha,EE}^i \right)^2$ one can set $g_2(d\phi) := \langle \mathbf{o}_\alpha^i(\phi_i + d\phi), \mathbf{o}_{\alpha,EE}^i \rangle$

◆ **Careful:** Only in case the IK problem has a solution, it can be shown that $\min_{d\phi} E_2(d\phi) = \max_{d\phi} g_2(d\phi)$





- ◆ **Revolute Joint** transform $E = E_1 + E_2$ into $g = g_1 + g_2$

- ◆ applying joint rotation: $\mathbf{R}(d\phi) = \begin{pmatrix} \cos d\phi & -\sin d\phi & 0 \\ \sin d\phi & \cos d\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$

gives $\underline{\mathbf{p}}^i(\phi_i + d\phi) = \mathbf{R}(d\phi)\underline{\mathbf{p}}^i(\phi_i)$ and

$$g_1(d\phi) = \langle \underline{\mathbf{p}}^i(\phi_i + d\phi), \underline{\mathbf{p}}_{EE}^i \rangle$$

$$= a_p \cdot \cos d\phi + b_p \cdot \sin d\phi + c_p$$

$$\text{with } a_p = p_x^i p_{EE,x}^i + p_y^i p_{EE,y}^i \quad b_p = p_x^i p_{EE,y}^i - p_y^i p_{EE,x}^i \quad c_p = p_z^i p_{EE,z}^i$$

- ◆ similarly we get

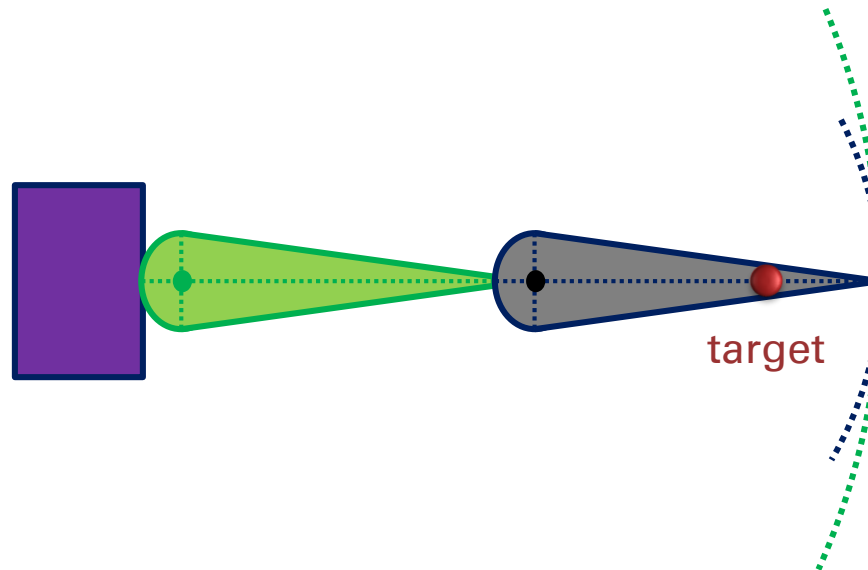
$$g_2(d\phi) = \langle \mathbf{o}_\alpha^i(\phi_i + d\phi), \mathbf{o}_{\alpha,EE}^i \rangle$$

$$= a_{o,\alpha} \cdot \cos d\phi + b_{o,\alpha} \cdot \sin d\phi + c_{o,\alpha}$$

- ◆ gathering all we get

$$g(d\phi) = g_1(d\phi) + g_2(d\phi) = a \cdot \cos d\phi + b \cdot \sin d\phi + c$$

- ◆ we get optimal $d\phi$ from $\partial_{d\phi} g(d\phi) = 0$ and $\partial_{d\phi}^2 g(d\phi) < 0$



- ◆ the CCD can get stuck in degenerate situations
- ◆ solution strategy:
 - ◆ per iteration add a random offset to each joint parameter
 - ◆ decrease (i.e. divide by 2) amplitude of random offset after each iteration



UNCONSTRAINED IK

Steepest Descent

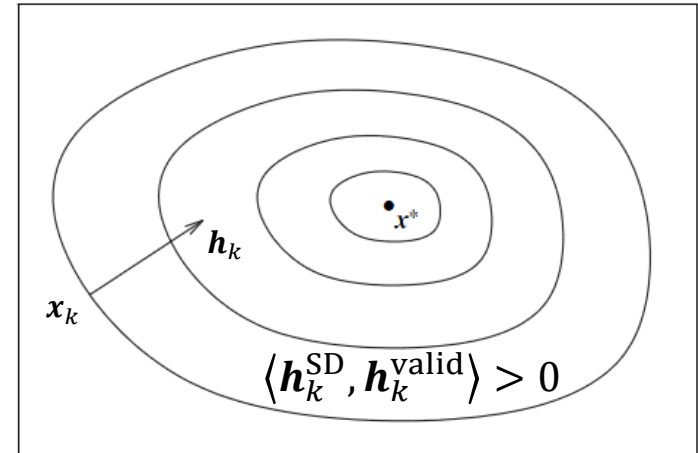
- ◆ In the **steepest descent** method the search **direction** is the negative gradient which points in direction of steepest descent of the function

$$\mathbf{h}_k^{\text{SD}} = -\nabla f_k$$

- ◆ In case of a linear least squares problem
 $f(\mathbf{p}) = \mathbf{r}^T \mathbf{W} \mathbf{r}$, with $\mathbf{r} = \mathbf{A} \mathbf{p} - \mathbf{b}$
gradient is $\nabla f_k = 2\mathbf{A}^T \mathbf{W} \mathbf{r}_k$
and the optimal step width can be computed exactly to

$$\alpha_k = -\frac{\mathbf{r}_k^T \mathbf{W} \mathbf{A} \nabla f_k}{\nabla f_k^T \mathbf{A}^T \mathbf{W} \mathbf{A} \nabla f_k}$$

- ◆ otherwise line search needed.



All directions that have a positive scalar product with negative gradient are **descent** and therefore **valid** search **directions**

Newton direction

- The **Newton direction** follows from the Taylor expansion up to second order
- The direction is computed from setting the gradient of the second order approximation to zero
- optionally one can solve linear system for \mathbf{h}_k^N
- **natural step length**: $\alpha_k = 1$
- \mathbf{h}_k^N is descent direction if Hessian is positive definite

$$\begin{aligned} m_k(\mathbf{x}_k + \mathbf{h}) &= f_k + \nabla f_k^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \nabla^2 f_k \mathbf{h} \\ \mathbf{0} &= \nabla_{\mathbf{h}} m_k(\mathbf{x}_k + \mathbf{h}_k^N) \\ &= \nabla f_k + \nabla^2 f_k \mathbf{h}_k^N \end{aligned}$$

$$\mathbf{h}_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k$$

$$\nabla^2 f_k \mathbf{h}_k^N = -\nabla f_k$$

- convergence rate near local minimum quadratic



Quasi Newton methods

- Quasi Newton methods track approximation of Hessian or its inverse

$$\mathbf{B}_k \approx \nabla^2 f_k \quad \text{or} \quad \mathbf{H}_k \approx \nabla^2 f_k^{-1}$$

- Taylor expansion of gradient gives **secant equation** as constraint

$$\underbrace{\nabla f_{k+1} - \nabla f_k}_{\mathbf{y}_k} \approx \nabla^2 f_k \cdot \underbrace{(\mathbf{x}_{k+1} - \mathbf{x}_k)}_{\mathbf{s}_k}$$

- Furthermore, \mathbf{B}_{k+1} should be symmetric, positive def. and minimize difference to \mathbf{B}_k in Frob. norm yielding

$$\Rightarrow \mathbf{B}_{k+1} \mathbf{s}_k = \mathbf{y}_k$$

(DFP)

- Davidon, Fletcher, Powell (DFP) update
- same idea for \mathbf{H}_k yields the better

$$\mathbf{B}_{k+1} = (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) \mathbf{B}_k (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) + \rho_k \mathbf{y}_k \mathbf{y}_k^T$$

$$\text{with } \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$$

- (BFGS) update
- Low memory implementation represents \mathbf{H}_k through few vectors \rightarrow L-BFGS

(BFGS)

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T$$





- ◆ n -dim. parameter space $P = R^n: q_{j=1\dots n}$ or $\vec{q} \in P$
- ◆ m -dim. target space $T = R^m: x_{i=1\dots m}$ or $\vec{x} \in T$
- ◆ forward kinematic function $\vec{f}: \vec{q} \mapsto f_i(\vec{q})$ or $\vec{f}(\vec{q}) \in T$
- ◆ gradient is **Jacobian matrix**: $J(\vec{q}) = \left(\frac{\partial f_i(\vec{q})}{\partial q_j} \right)_{ij} \in R^{m \times n}$
- ◆ Taylor series: $\vec{f}(\vec{q}_0 + d\vec{q}) = \vec{f}(\vec{q}_0) + J(\vec{q})d\vec{q} + \mathcal{O}(d\vec{q}^2)$
- ◆ Residua: $r_i(\vec{q}) = f_i(\vec{q}) - x_i$ or $\vec{r}(\vec{q}) = \vec{f}(\vec{q}) - \vec{x}$
- ◆ objective function: $E(\vec{q}) = \frac{1}{2} \sum_{i=1}^n w_i r_i^2$ or $E(\vec{q}) = \frac{1}{2} \vec{r}^T \mathbf{W} \vec{r}$
with $\mathbf{W} = \text{diag}(w_i)$
being the weight matrix

- ◆ In matrix representation:

$$\underline{\mathbf{p}}_{EE}^0 = {}^0\mathbf{R}_N \underline{\mathbf{p}}_{EE}^N + {}^0\vec{\mathbf{t}}_N$$

$${}^0\mathbf{T}_N(\vec{\mathbf{q}}) = {}^0\mathbf{T}_1 {}^1\mathbf{T}_2 \cdot \dots \cdot {}^{N-1}\mathbf{T}_N = \begin{pmatrix} {}^0\mathbf{R}_N & {}^0\vec{\mathbf{t}}_N \\ \vec{\mathbf{0}}^T & 1 \end{pmatrix}$$

- ◆ We exploit the fact that only ${}^{j-1}\mathbf{T}_j(q_j)$ depends on q_j .
Therefore we have

$$\frac{\partial {}^0\mathbf{T}_N(\vec{\mathbf{q}})}{\partial q_j} = {}^0\mathbf{T}_{j-1} \frac{\partial {}^{j-1}\mathbf{T}_j(q_j)}{\partial q_j} {}^j\mathbf{T}_N$$

- ◆ For a rotation around an axis we get in 3D:

$$\frac{\partial}{\partial \varphi} \text{Rot}_z(\varphi) = \frac{\partial}{\partial \varphi} \begin{pmatrix} \cos \varphi & -\sin \varphi & 0 \\ \sin \varphi & \cos \varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -\sin \varphi & -\cos \varphi & 0 \\ \cos \varphi & -\sin \varphi & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- ◆ This can be used in DH-notation:

$$\frac{\partial}{\partial \varphi} {}^{j-1}\mathbf{T}_j = \underbrace{\text{Rot}_x(\alpha_{j-1}) \cdot \text{Trans}_x(a_{j-1}) \cdot \text{Trans}_z(d_j)}_{{}^{i-1}\mathbf{C}_i} \cdot \frac{\partial}{\partial \varphi_j} \text{Rot}_z(\varphi_j)$$

- energy: $E(\vec{q}) = \frac{1}{2} \sum_{i=1}^n w_i r_i^2$, with $r_i(\vec{q}) = f_i(\vec{q}) - x_i$
- gradient: $\frac{\partial}{\partial q_j} E = \sum_i w_i r_i \frac{\partial f_i}{\partial q_j}$ or $\vec{\nabla}_{\vec{q}} E = \vec{r}^T \mathbf{W} \mathbf{J}$ (row vector)

Descent directions:

- steepest descent direction:

$$\mathbf{h}_k^{\text{SD}} = -\vec{\nabla}_{\vec{q}}^T E_k = -\mathbf{J}_k^T \mathbf{W} \vec{r}_k$$

- approximate Hessian $\tilde{\mathbf{H}}$ (assume \mathbf{J} in $\vec{\nabla}_{\vec{q}} E$ to be constant):

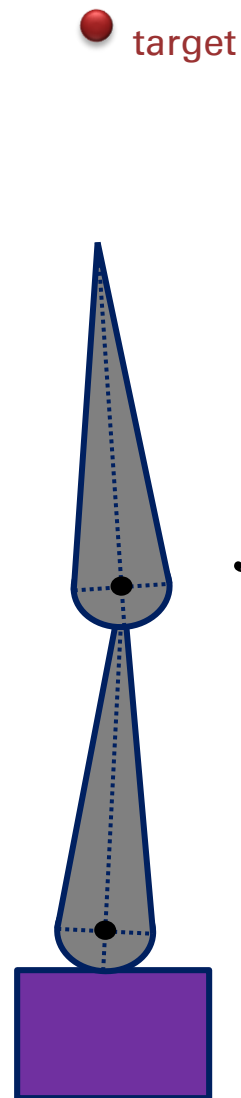
$$\mathbf{H}_k := \vec{\nabla}_{\vec{q}} \vec{\nabla}_{\vec{q}}^T E_k \approx \mathbf{J}_k^T \mathbf{W} \mathbf{J}_k =: \tilde{\mathbf{H}}_k$$

- approximate also Newton direction (the resulting WNLLS approach is then called Gauss-Newton method):

$$\tilde{\mathbf{H}}_k \mathbf{h}_k^{\text{GN}} = -\vec{\nabla}_{\vec{q}}^T E_k \Rightarrow \mathbf{J}_k^T \mathbf{W} \mathbf{J}_k \mathbf{h}_k^{\text{GN}} = -\mathbf{J}_k^T \mathbf{W} \vec{r}_k$$

- as \mathbf{J} can become singular at singular configurations, solve for Gauss-Newton direction with truncated SVD.

- One problem with the Gauss-Newton method is that the Pseudo inverse becomes unstable in degenerate positions
- These arise in IK for example when the robot arm is fully extended
- This problem does not arise in the steepest descent method where only the transposed of the Jacobian is used
- Idea: combine both methods optimally



$$\mathbf{J} = \begin{pmatrix} \frac{\partial \vec{f}}{\partial q_1} & \frac{\partial \vec{f}}{\partial q_2} \end{pmatrix} = \begin{pmatrix} -0.95 & -0.97 \\ 0.08 & -0.06 \end{pmatrix}$$

$$\mathbf{J}^{-1} = \begin{pmatrix} -0.45 & 7.2 \\ -0.59 & -7.1 \end{pmatrix}$$

$$d\vec{x} = \begin{pmatrix} 0 \\ 0.5 \end{pmatrix}$$

$$d\vec{q} = \begin{pmatrix} 3.6 \\ -3.5 \end{pmatrix}$$

- Idea: combine steepest descent $\mathbf{h}_k^{\text{SD}} = -\mathbf{J}_k^T \mathbf{W} \vec{\mathbf{r}}_k$ with Gauss-Newton approach $\mathbf{J}_k^T \mathbf{W} \mathbf{J}_k \mathbf{h}_k^{\text{GN}} = -\mathbf{J}_k^T \mathbf{W} \vec{\mathbf{r}}_k$ with a weighting factor λ :

$$(\mathbf{J}_k^T \mathbf{W} \mathbf{J}_k + \lambda \mathbf{I}) \mathbf{h}_k^{\text{LM}} = -\mathbf{J}_k^T \mathbf{W} \vec{\mathbf{r}}_k \quad [1]$$

- Large λ result in steepest descent update and small λ in Gauss-Newton update. λ is initialized to large values and decreased close to the optimum, which leads to fast convergence (quadratic in best case).
- The simplest approach with quadratic convergence close to solution is to set $\lambda = \|\vec{\mathbf{r}}_k\|$
- Levenberg Marquardt suggest to adapt λ -control to the directional curvature by the generalization

$$(\mathbf{J}_k^T \mathbf{W} \mathbf{J}_k + \lambda \text{diag}(\mathbf{J}_k^T \mathbf{W} \mathbf{J}_k)) \mathbf{h}_k^{\text{LM}} = -\mathbf{J}_k^T \mathbf{W} \mathbf{r}_k \quad [2]$$

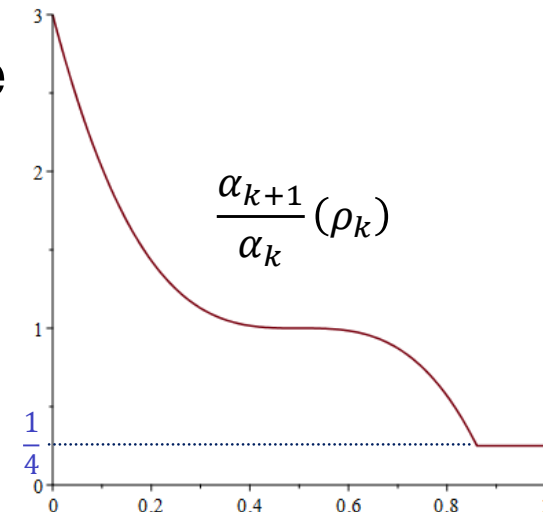
- ◆ An important criterion to control λ is the fraction between the real improvement and the improvement predicted by the quadratic approximation with \mathbf{J} and $\tilde{\mathbf{H}}$:

$$\rho_k(\vec{\mathbf{q}}) = \frac{E(\vec{\mathbf{q}}) - E(\vec{\mathbf{q}} + \mathbf{h}_k^{\text{LM}})}{2\mathbf{h}_k^{\text{LM}^T}(\lambda_k \mathbf{h}_k^{\text{LM}} - \mathbf{J}_k^T \mathbf{W} \mathbf{r}_k)} \quad [3]$$

- ◆ One can update a factor α_k in $\lambda_k = \sqrt{\alpha_k} \|\vec{\mathbf{r}}_k\|$ over the optimization resulting in the **self-adaptive LM** method:

- ◆ input from user: $\vec{\mathbf{q}}_0, \alpha_0$

1. compute $\|\vec{\mathbf{r}}_k\|$ and check for convergence
2. set $\lambda_k = \sqrt{\alpha_k} \|\vec{\mathbf{r}}_k\|$
3. solve [1] or [2] for \mathbf{h}_k^{LM} , $\vec{\mathbf{q}}_{k+1} = \vec{\mathbf{q}}_k + \mathbf{h}_k^{\text{LM}}$
4. compute ρ_k from [3]
5. set $\alpha_{k+1} = \alpha_k \cdot \max\left\{\frac{1}{4}, 1 - 2(2\rho_k - 1)^3\right\}$ and goto 1.





- ◆ Descent with steepest descend direction is also called **transposed Jacobian method** in IK literature. It needs to be combined with a line search method. As no SVD is necessary, a single iteration is fast. The method is therefore often used in interactive IK approaches, e.g. in character pose editors.
- ◆ Descent with the Newton direction computed from the approximate Hessian is called **Gauss-Newton method** or non linear least squares in math and **inverse Jacobian method** in IK. The natural step width of 1 simplifies line search. One can simply check for decrease in energy and half the step width until an energy decrease is achieved.
- ◆ The taxi cab method corresponds to iteratively optimize only one joint parameter at the time. This is called **cyclic coordinate descent method** in IK and often implemented as the optimal step width can be computed analytically.



- ◆ Levenberg-Marquardt and BFGS are better than inverse Jacobian and often used for IK. Furthermore, one can use the non linear conjugate gradient method discussed in CG1.
- ◆ Another strategy is to start with cyclic coordinate descent and continue with Levenberg-Marquardt or BFGS.





CONSTRAINED IK

- Most joints have constraints on their parameters which can be defined with a **lower and upper bound**:

$$l_j \leq q_j \leq u_j$$

- such constraints are often called **simple constraints** and the optimization problem is called **bound constrained**.
- In all descent approaches – including the cyclic coordinate descent – these constraints can be easily incorporated by the **gradient projection method**:

- First one defines the projection operation Π on the parameter space P :

$$\Pi(\vec{q}, \vec{l}, \vec{u})_j = \text{median}\{l_j, q_j, u_j\}$$

- And uses this to project descent directions to the feasible region:

$$\vec{h} = \Pi(\vec{q} + \vec{h}, \vec{l}, \vec{u}) - \vec{q}$$

- The rest of the algorithms stays the same
- Bound constrained versions exists for L-BFGS-B, (C++)



- ◆ Positional constraints like that a foot should stay on the floor are non linear in the parameters.

- ◆ Such general constraints can be written in the form of equalities or inequalities:

$$c_l(\vec{q}) = 0 \text{ or } h_o(\vec{q}) \geq 0$$

- ◆ approaches to incorporate equality constraints:
 - ◆ **Lagrangian multiplier** method incorporates constraints into objective function and adds multipliers as additional parameters
 - ◆ **constraint forces** can be derived from constraints and also be used to reinforce constraints in case of numerical deviations or initialization that violates constraints
- ◆ approaches to incorporate inequality constraints
 - ◆ Linear complementary problems (LCP)

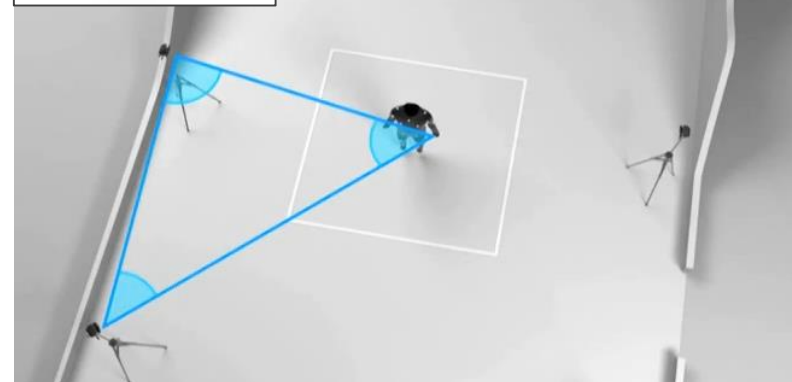


MOTION CAPTURING

Standard Motion Capturing



- ◆ Standard MoCap Approach
 - ◆ add markers at joints
 - ◆ illuminate markers from all directions
 - ◆ acquire views from several synchronized cameras
 - ◆ detect markers in each acquired view
 - ◆ match markers between views
 - ◆ reconstruct 3D positions
 - ◆ track points over time
 - ◆ per frame match points to skeleton and fit skeleton
- ◆ extension: use light emitters of different frequency as markers with id (no matching necessary)



<http://www.youtube.com/watch?v=DaOAZA0xSMY>





Keith Grochow, Steven L. Martin, Aaron Hertzmann, Zoran Popovic,
Siggraph 2004

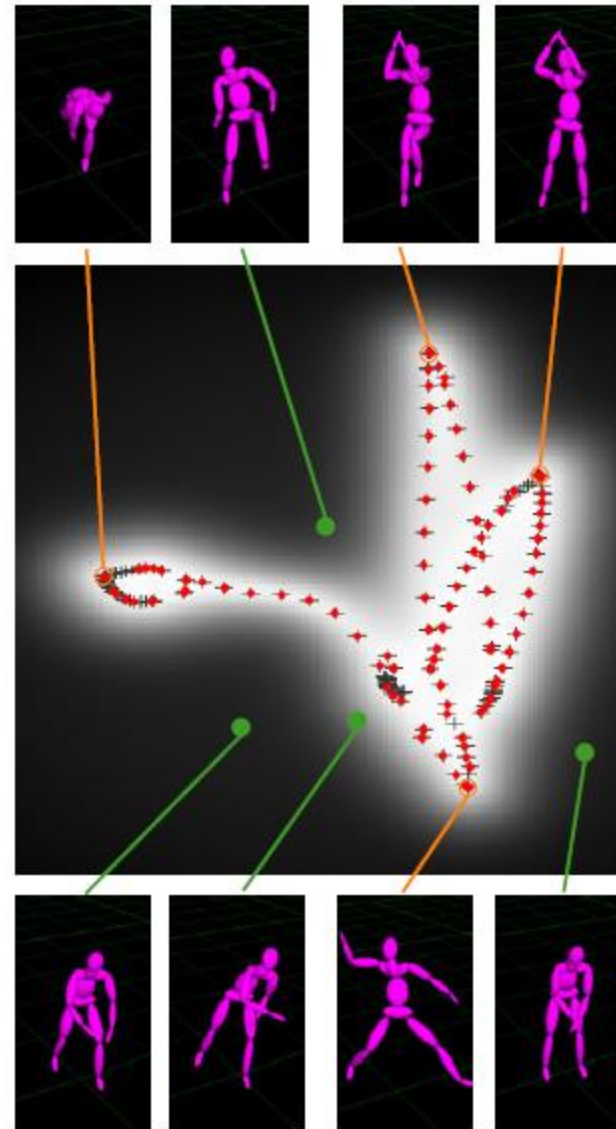
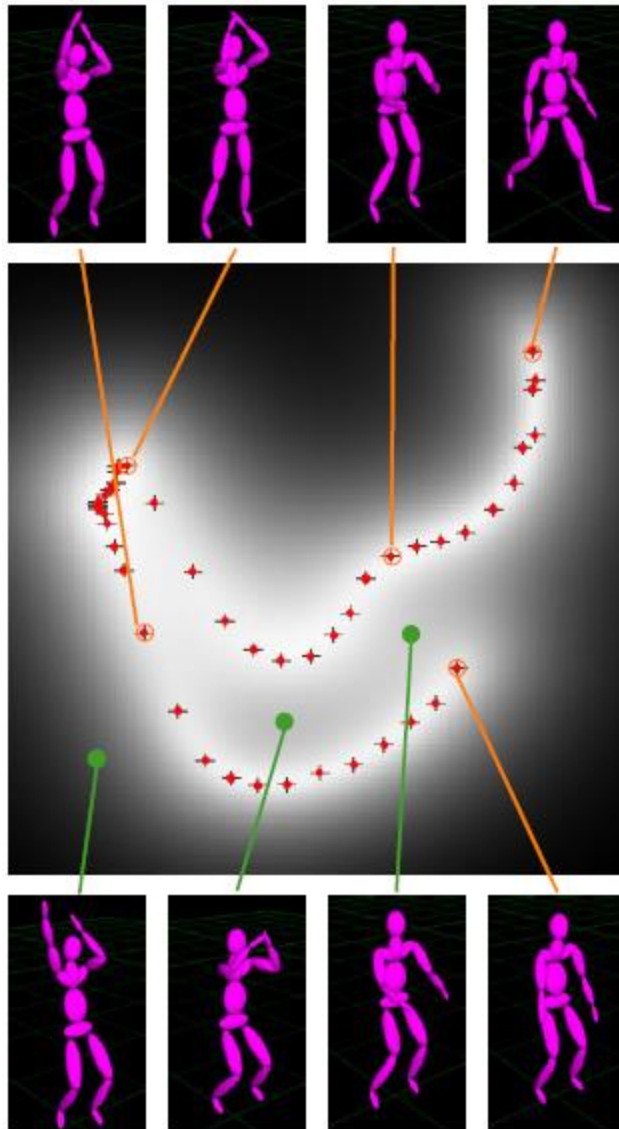
Idea

- ◆ use motion capture data to disambiguate IK

Overview

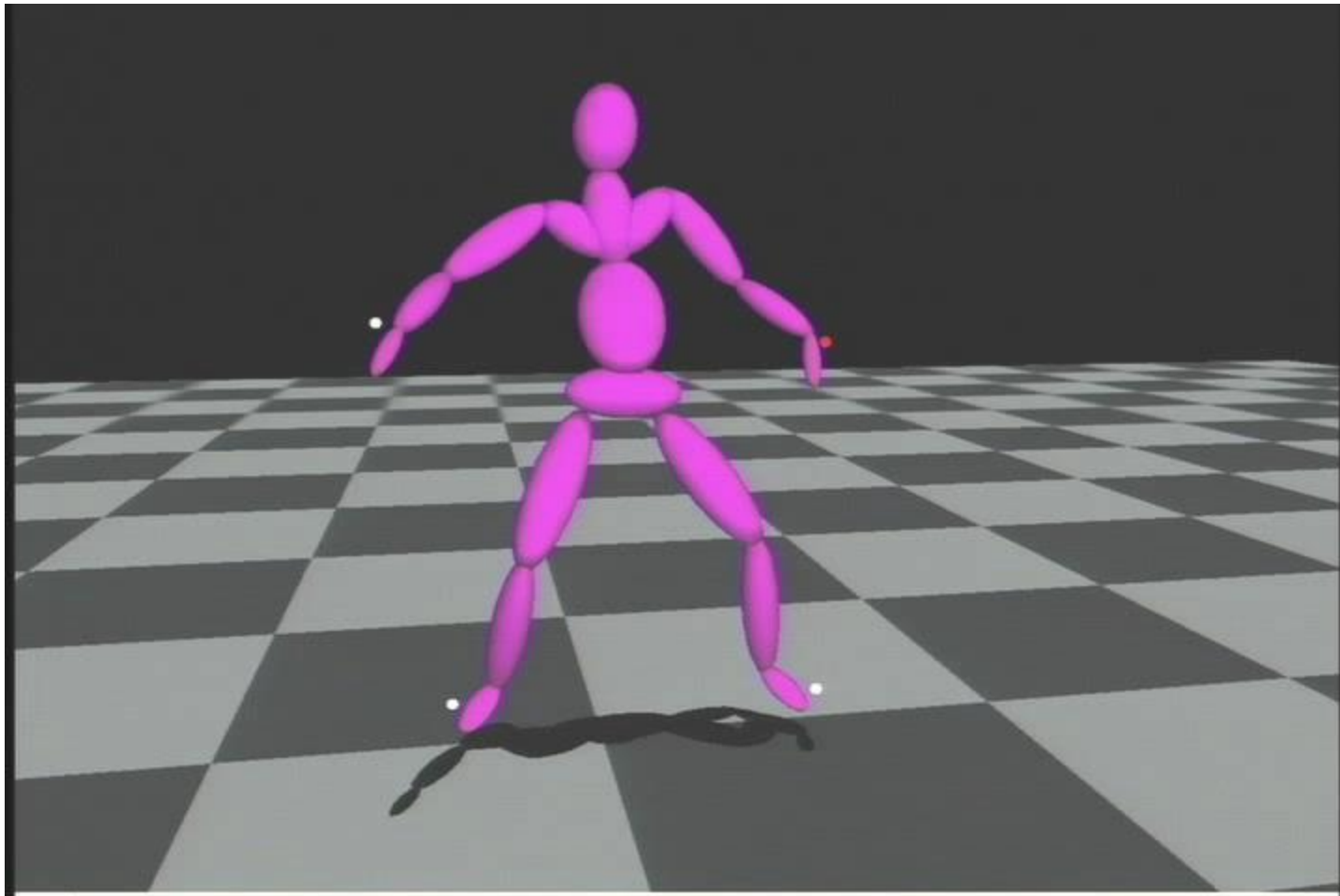
- ◆ capture different motion sequences
- ◆ map each pose to a feature space
- ◆ learn distribution of poses in reduced feature space
- ◆ solve IK problem by maximizing pose probability

Style-Based Inverse Kinematics



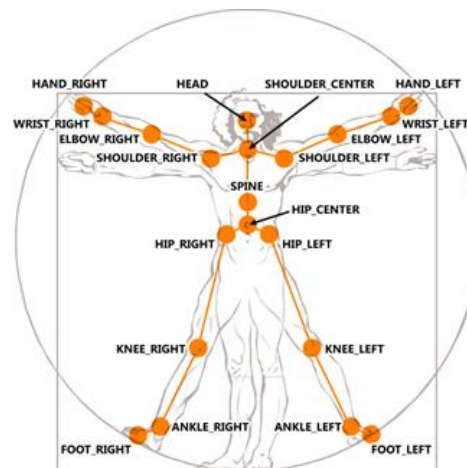
Example for
learned pose
spaces of
jump shot (left)
and baseball
pitch (right)
sequences
and their pose
probability
distributions.

Red points are
training poses,
orange connec-
tions mark some
training poses and
green connections
show extrapolated
poses



www.youtube.com/watch?v=X5Z7ZJ39zAA

- uses depth camera with a machine learning approach
- learning
 - 2D parameterized template human mesh (vitruvian manifold)
 - generate a huge number of depth images from different poses of vitruvian manifold (render depth & texture coords.)
 - learn local image features with decision tree to map each depth image pixel to texture coordinates
- recognition
 - use decision tree to estimate per pixel texcoords
 - fit pose to depth and texcoord image

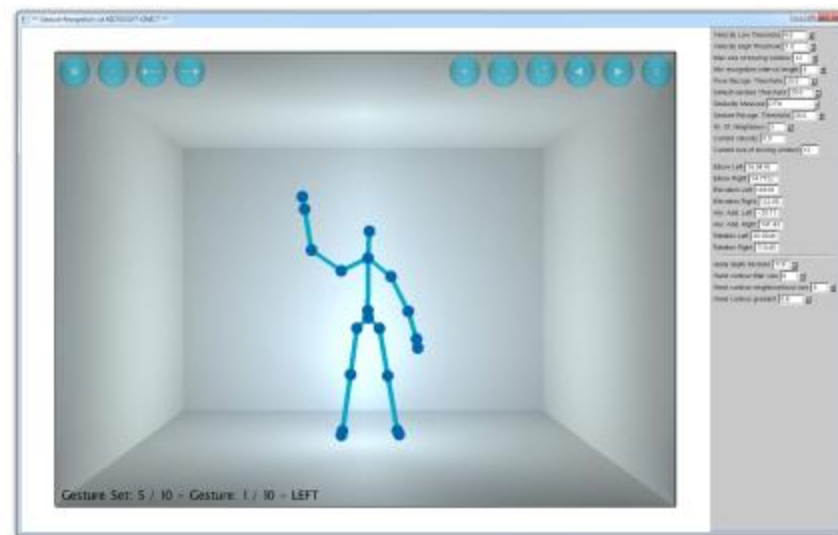


kinect skeleton



vitruvian manifold

Taylor, Shotton, Sharp, Fitzgibbon
CVPR 2012



© Ludwig Schmutzler



- ◆ Helge Rhodin, James Tompkin, Kwang In Kim, Kiran Varanasi, Hans-Peter Seidel, Christian Theobalt, Eurographics 2014 ([http](http://www.cgv.rwth-aachen.de/~rhodin/papers/2014_eurographics.pdf))

Idea

- ◆ use Kinect skeleton tracking to steer character with different skeleton

Overview

- ◆ Given sparse pose mapping from source to target, learn pose mapping without rigging and skinning
- ◆ Allows interactive control of virtual character

Interactive Motion Mapping for Real-time Character Control

EUROGRAPHICS 2014

Helge Rhodin¹, James Tompkin^{1,2}, Kwang In Kim^{1,3}, Kiran Varanasi^{1,4}
Hans-Peter Seidel¹, Christian Theobalt¹

¹Max-Planck-Institute for Informatics

²Intel Visual Computing Institute

³Lancaster University

⁴Technicolor Research



www.youtube.com/watch?v=SG5D12tBBAk



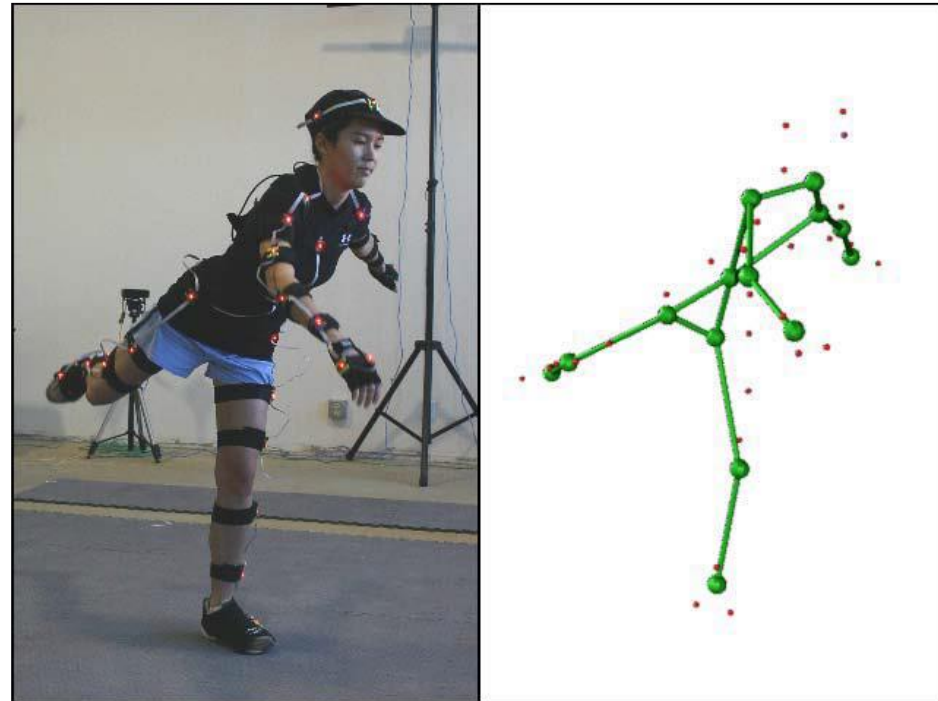


SKELETON FITTING

Problem Statement



- ◆ Input: set of 3D marker points tracked over time where markers can be occluded and re-appearing markers are not identified but receive a new label
- ◆ Output: skeleton topology, bone lengths, joint locations in rotation centers (markers are placed on surface)



Reference: Adam G. Kirk James F. O'Brien David A. Forsyth,
Skeletal Parameter Estimation from Optical Motion Capture Data, CVPR 2015

- ◆ filter erroneous marker information before vanishing and after their re-appearance
- ◆ build matrix of marker pair distances and measure pair distance variance over time
- ◆ segment markers into rigid components by spectral clustering
- ◆ build all segment pairs and fit joint locations to minimize variance of distance to both segment markers over time
- ◆ extract topology by minimum spanning tree
- ◆ match marker time slabs
- ◆ use IK to extract joint parameters

Skeletal Parameter Estimation from Optical Motion Capture Data

**Adam G. Kirk
James F. O'Brien
David A. Forsyth**

University of California - Berkeley

<http://graphics.berkeley.edu/papers/Kirk-SPE-2005-06>



- (1) Wang, L-CT, and Chih-Cheng Chen. "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators." *Robotics and Automation, IEEE Transactions on* 7.4 (1991): 489-499.
- (2) Welman, Chris. *Inverse kinematics and geometric constraints for articulated figure manipulation*. Diss. Simon Fraser University, 1993.
- (3) Meredith, Michael, and Steve Maddock. *Real-time inverse kinematics: The return of the Jacobian*. Technical Report No. CS-04-06, Department of Computer Science, University of Sheffield, 2004.
- (4) Hess, Peter, et al. "Style-Based Inverse Kinematics." (2005).
- (5) Shan, Shidong. *A Levenberg-Marquardt method for large-scale bound-constrained nonlinear least-squares*. Diss. The University of British Columbia (Vancouver), 2008.
- (6) Taylor, Jonathan, et al. "The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.
- (7) Kenwright, Ben. "Inverse Kinematics with Dual-Quaternions, Exponential-Maps, and Joint Limits." *International Journal On Advances in Intelligent Systems* 6.1 and 2 (2013): 53-65.
- (8) Rhodin, Helge, et al. "Interactive motion mapping for real-time character control." *Computer Graphics Forum*. Vol. 33. No. 2. 2014.
- (9) Adam G. Kirk James F. O'Brien David A. Forsyth, *Skeletal Parameter Estimation from Optical Motion Capture Data*, CVPR 2015
- (10) http://chrishecker.com/Inverse_kinematics
- (11) <https://github.com/PatWie/CppNumericalSolvers> (L-BFGS-B Solver in C++)