Reconstituting and Evolving Robot Movements by PCA on Captured Human Motions

Syungkwon Ra, ChangHwan Kim, Sang-Rok Oh
Korea Institute of Science and Technology, Korea

Paolo Dario
Scuola Superiore Sant'Anna, Italy

† PCA : Principal Component Analysis

Introduction

- ***Can robots imitate the natural motions of human?***
  - By means of motion capture system
- Robot movement lies on high-dimensional space
  - Dimension Reduction by principal component analysis (PCA)
- Evidence of optimization (usually w.r.t. a physical criterion, e.g., metabolic energy) taking place with movement learning.
  - Optimization minimizing total torque
- Transition from “closed-loop” to “open-loop” control as learning takes place.
  - Evolving robot movements

Related Works

- Neuroscience
  - [Raibert, Horn 1978] [Hollerbach, Flash 1982]
    : Brain carry out inverse dynamics-based optimization
    : Look-up table for motions
- PCA
  - [Fod, Mataric, Jenkins 2002] : Movement classification
  - [Tatani, Nakamura 2003] : Dimensionality reduction
  - [Lim, Ra, Park 2005] : Recombination of principal components
- Evolutionary Robotics
  - [Zykov, Bongard, Lipson 2004] : Evolving dynamic gaits
  - [Aydemir, Iba 2006] : Behavior Acquisition

Two Practical Issues

- Captured Human Motions
  - Kinematic information
  - Accumulated experience of a human being
  - Dynamically consistent and optimized

- ISSUE 1 : How to utilize a limited number of captured human motions for a robot?
  - We cannot store all human motions that a robot is in need of

- ISSUE 2 : Are human motions also optimal to a robot?
  - Their dynamic properties are different

Our Approaches

- Movement primitives are represented as joint trajectories
- Statistical analysis and reconstitution
  - The basis functions are obtained from Principal Component Analysis of motion data
  - Robot motions are reconstituted by linear combination of the basis functions.
- Evolving human motions to robot motions
  - Evolutionary computation
  - PCA-based genetic operator

Issue 1

- How to utilize a limited number of captured human motions for a robot?
  - We cannot store all human motions that a robot is in need of

PCA : PRINCIPAL COMPONENT ANALYSIS

- Given vector time series data $\{x[0], x[1], \cdots, x[N]\}$, each $x[i] \in \Re^p$ is a sample of the random vector $x \in \Re^p$.

- Sample mean : $\bar{x} = \frac{1}{N+1}\sum_{i=0}^{N} x[i] \quad \in \Re^p$

- Sample covariance : $S = \frac{1}{N}\sum_{i=0}^{N}(x[i]-\bar{x})(x[i]-\bar{x})^T \quad \in \Re^{p\times p}$

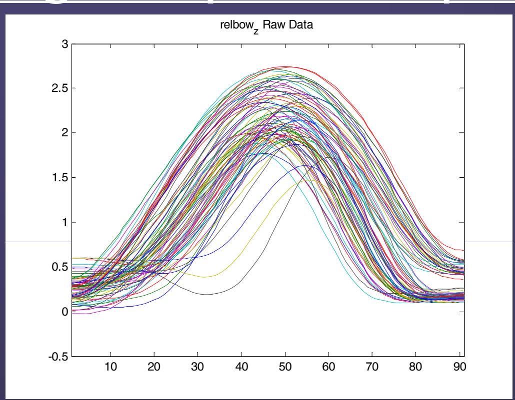
- Let [illegible] represent eigenvalue-eigenvector pairs for S , where $e_i^T e_j = \delta_{ij}$ (Kronecker delta)

  $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p > 0$

- The eigenvectors $\{e_1, e_2, \cdots ,e_p\}$ are the principal components.

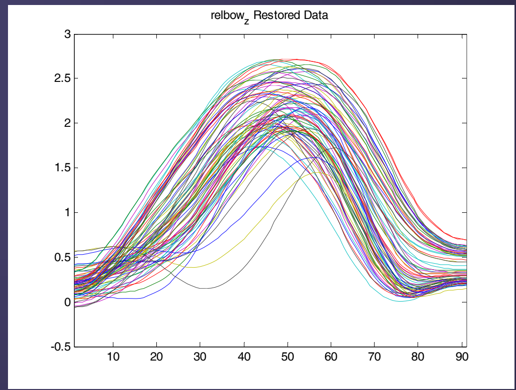
PCA on Captured Human Motions

original captured samples



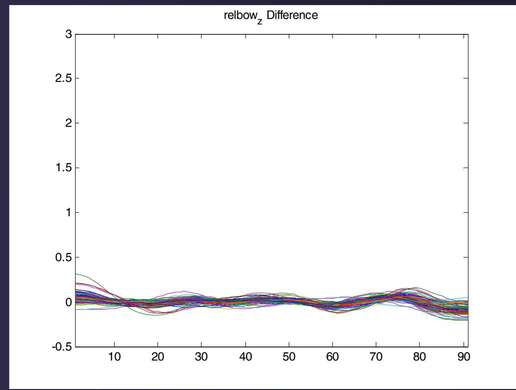
-

restored samples



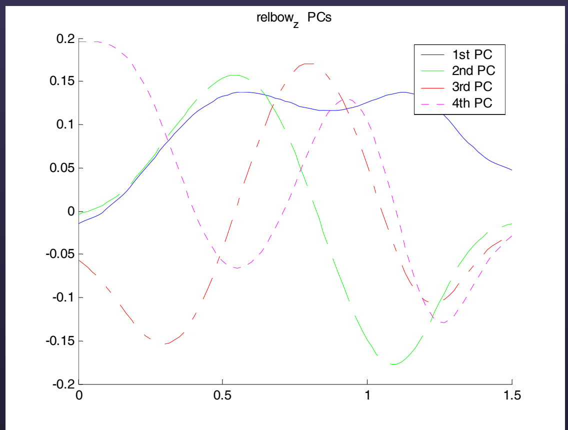
=

differences



PCA ↓

↗ Linear Combination



selected 4 P.Cs

P.C.	RATIO
1	74.36 %
2	23.50 %
3	1.65 %
4	0.28 %
Sum	99.79 %

related contribution

restored samples are represented as

$$q(t) = \bar{q}(t) + \sum_{i=1}^{4} \text{[illegible]}$$

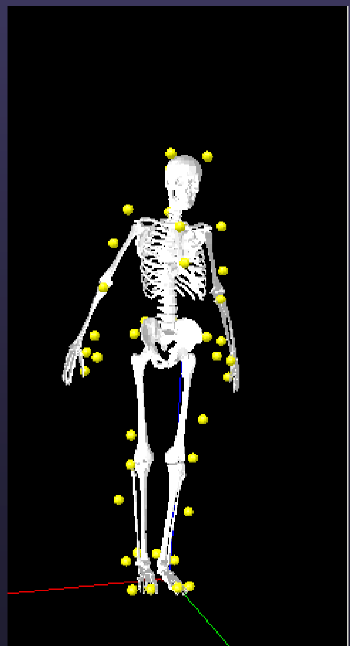
$\bar{q}(t)$: sample mean

$p_i(t)$: i-th principal component

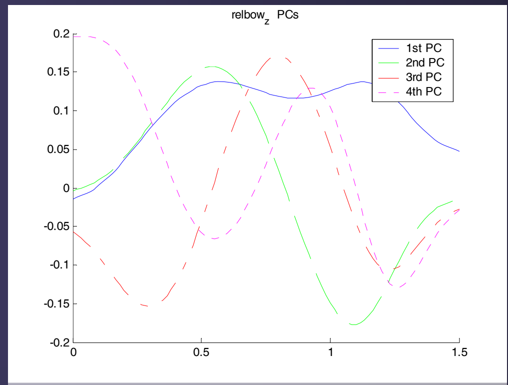
λ_i : i-th eigenvalue

Overall Procedure

Inverse Kinematics

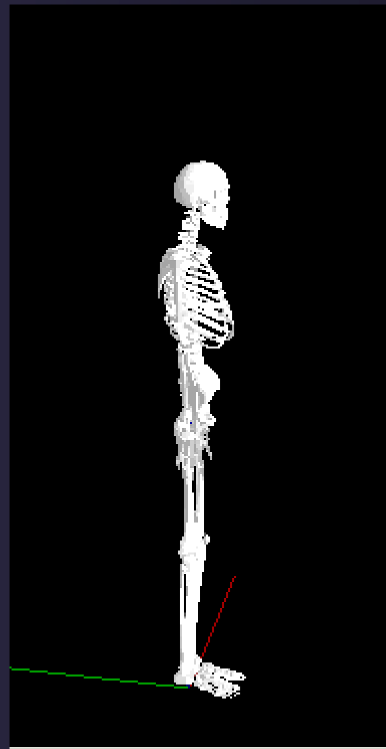


PCA



Motion Reconstitution

Motion condition



Motion Reconstitution via Kinematic Interpolation

- Linear combination of 3 dominant principal components

$$q(t) = \bar{q}(t) + \sum_{i=1}^{3} x_i p_i(t)$$

$q(t)$: robot joint trajectory

$\bar{q}(t)$: mean trajectory

$p_i(t)$: i-th principal

x_i: scalar weight

- Motion condition

[illegible]

Motion Reconstitution via Dynamics-based Optimization

- Linear combination of 4 dominant principal components

$$q(t) = \bar{q}(t) + \sum_{i=1}^{4} x_i q_i(t)$$

- Motion condition

[illegible]

- Dynamics-based Optimization : minimum torque

$$\min \; [illegible] = \frac{1}{2}\int [illegible]^2 \, [illegible]$$

subject to the dynamic equation of motion

[illegible]

- Optimization Variables are scalar weight x_i

Comparison : Optimization using B-spline

- Parameterize joint trajectories using B-spline

$$q(t) = \sum_{i=0}^{n} P_i N_{i,k}(t) \qquad N_{i,k}(t) = \frac{(t-t_i)N_{i,k-1}(t)}{t_{i+k}-t_i} + \frac{(t_{i+k+1}-t)N_{i+1,k-1}(t)}{t_{i+k+1}-t_{i+1}}$$

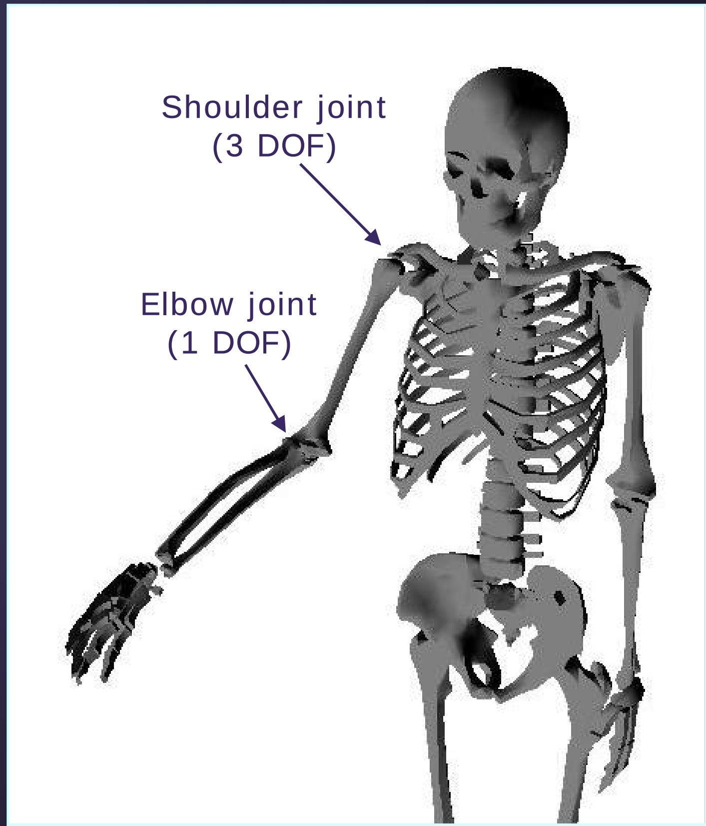
$$N_{i,0}(t) = \begin{cases} 1 & t_i \le t < t_{i+1} \\ 0 & \text{else} \end{cases}$$

- Optimization variables are control points.

- We benchmark the optimization results obtained using the PCA basis functions with those obtained by parameterizing joint trajectories using general B-splines

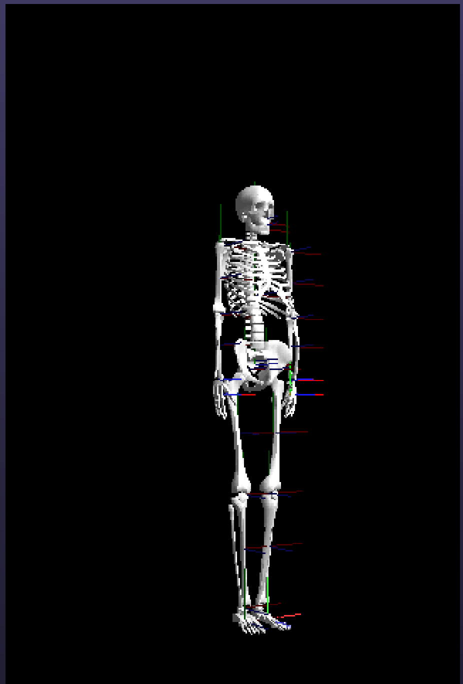
Case Study (I)

- Arm model
  - 3R(shoulder)-1R(elbow) structure
- Motion task
  - Raising and Reaching of hand
- Motion capture data
  - 50 trials of various hand lifting motions by subject.

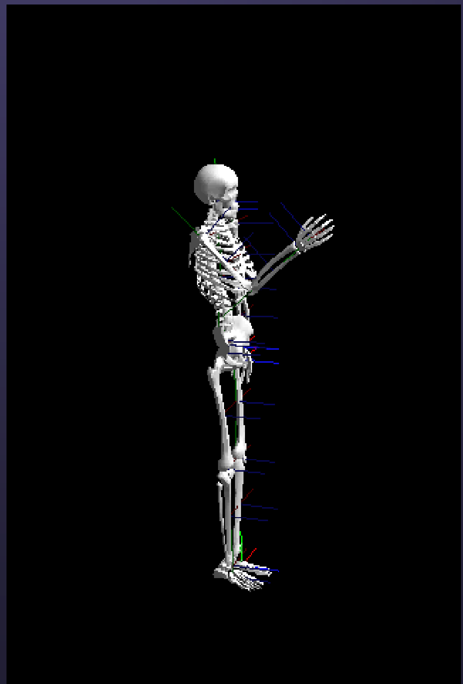


Kinematic Interpolation

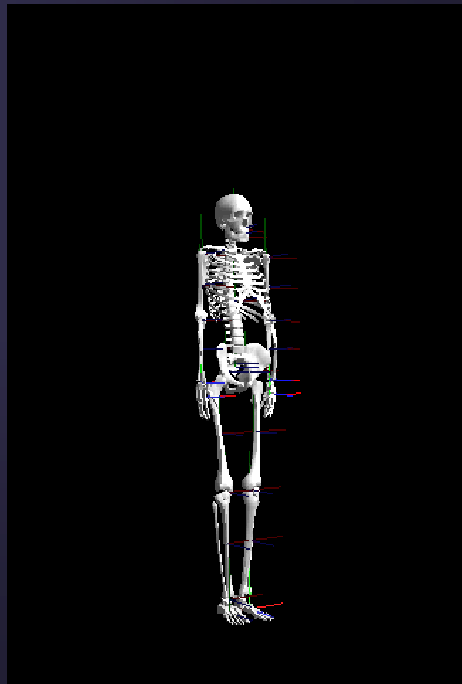
- Motions clearly resemble human motions, but may not necessarily be the most efficient from an energy consumption perspective.



Raising 1



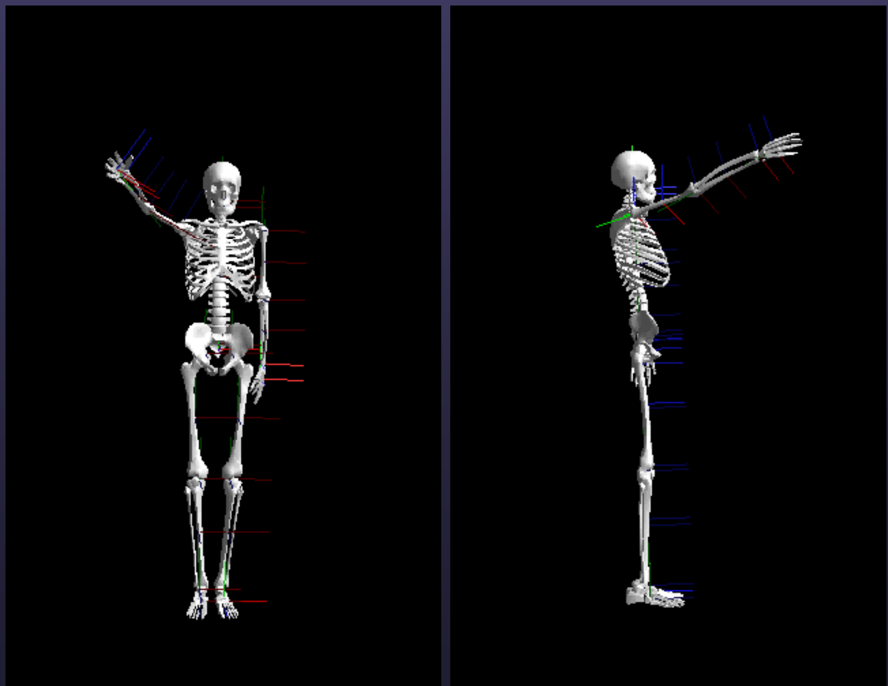
Raising 2



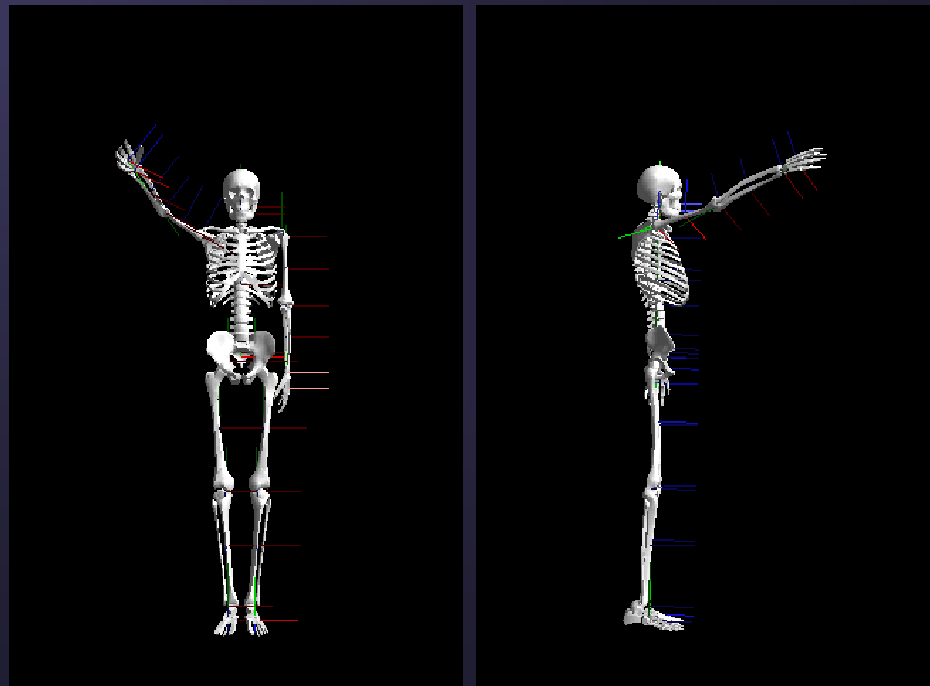
Reaching

Dynamics-based Optimization

- The motions generated from B-splines appear less natural than those obtained from PCA-based primitives.
- Since the principal components are extracted from human arm movement data, it is entirely reasonable to expect that such motions will resemble human arm motions.



B-spline



PCA

Comparison : PCA vs. B-spline

- B-spline

Number of optimization variables	12 (3x4dof)	16 (4x4dof)	20 (5x4dof)	24 (6x4dof)
Number of control points	7x4dof	8x4dof	9x4dof	10x4dof
Number of iterations	17	24	40	38
Objective function value	321.5	287.8	273.2	244.3
Computation time (sec.)	227.0	397.0	795.4	905.9

- PCA

Number of optimization variables	12 (3x4dof)	16 (4x4dof)	20 (5x4dof)	24 (6x4dof)
Number of principal components	2x4dof	3x4dof	4x4dof	5x4dof
Number of iterations	8	21	43	59
Objective function value	428.6	248.1	229.5	212.1
Computation time (sec.)	25.1	78.9	195.6	328.3

† MATLAB on Pentium 4 PC

Issue 2

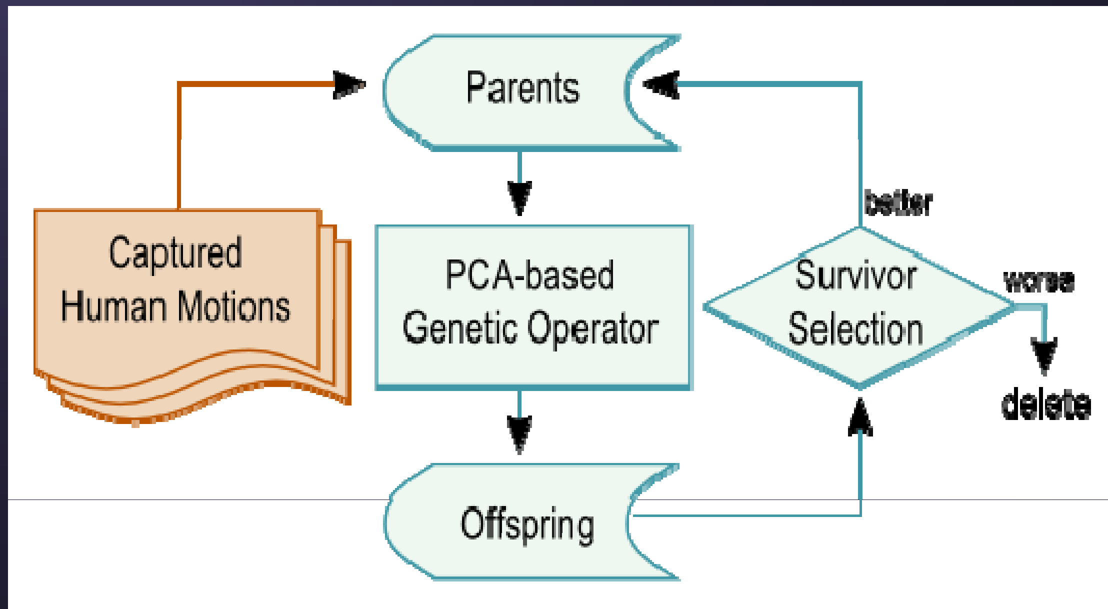
- Are human motions also optimal to a robot?
  - Their dynamic properties are different

Evolving Movement Primitives

AIM : "Refashion human motion patterns into trained ones for a specific robot"

- Each movement primitive m_i has its own condition c_i
- Assumption
  - Similar conditions, Similar movements
  - One optimal primitives can contribute to making neighborhood primitives be optimal
- Individuals
  - Movement primitives
- Genotype
  - Joint trajectories
- Fitness Function
  - Required torque

$$\frac{1}{2}\int \|\tau(t)\|^2 dt$$



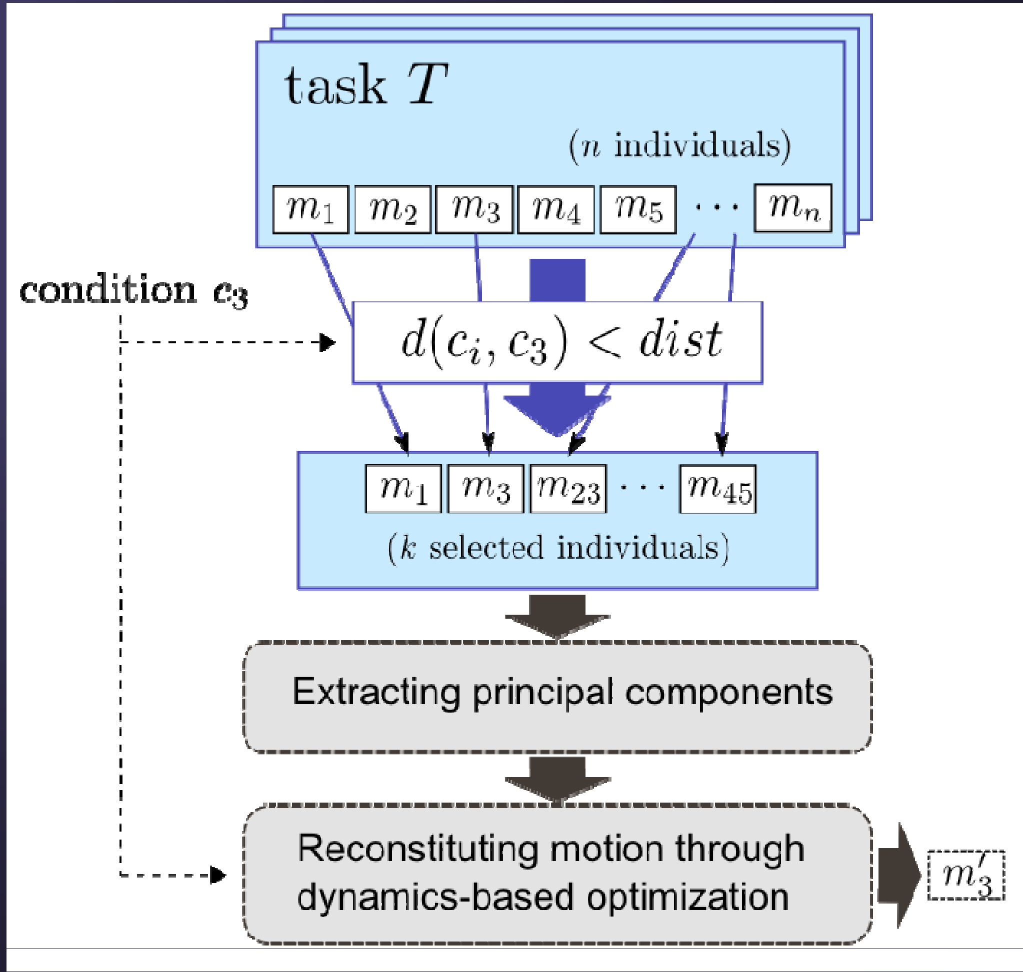
evolving procedure

PCA-based Genetic Operator

- Recombination operator
- Similarity
  - Distance metric

$$d(c_i, c_3)$$

- PCA
- Motion Reconstitution via dynamics-based optimization
- Repeating from (m_1, c_1) to (m_n, c_n)



Why Evolutionary Computation (EC)?

AIM : Refashion human motion patterns into trained ones for a specific robot

1) EC falls in the category of generate-and-test algorithms

→ Repeated trials while human learns a new sport or skill

2) EC uses not a one solution candidate but a whole collection simultaneously

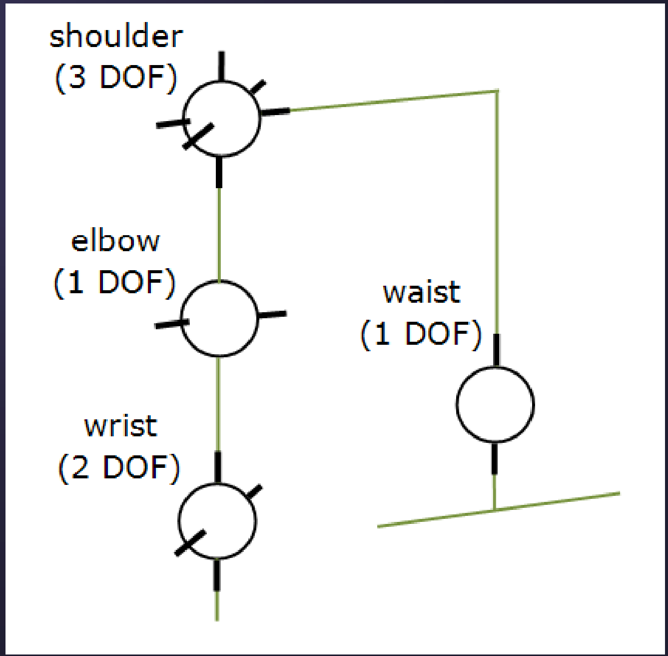
→ A set of movement primitives = collection of candidate solution

3) EC provides approximated solutions for high non-linearity problem

4) EC as global optimizer and genetic operator as local optimizer

Case Study (II) : Catching a ball

- KIST humanoid robot ‘MAHRU’
- Waist-arm model
  - 1R(waist)-3R(shoulder)-1R(elbow)-2R(wrist) structure



Case Study (II) : Catching a ball

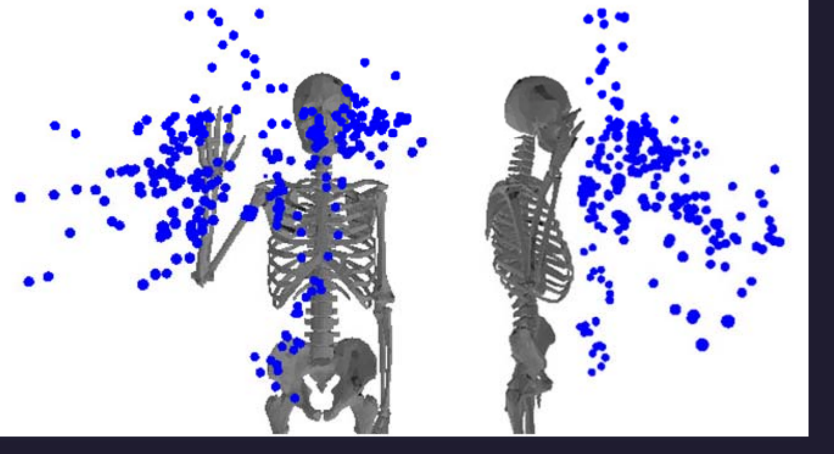
- Primitive motion
  - Catching a ball
- Motion capture data
  - 140 trials of various catching points by subject
- Distance metric

[illegible]

(R, p): rotation, posi[illegible]
at catching [illegible]
w, w: scaling weight[illegible]

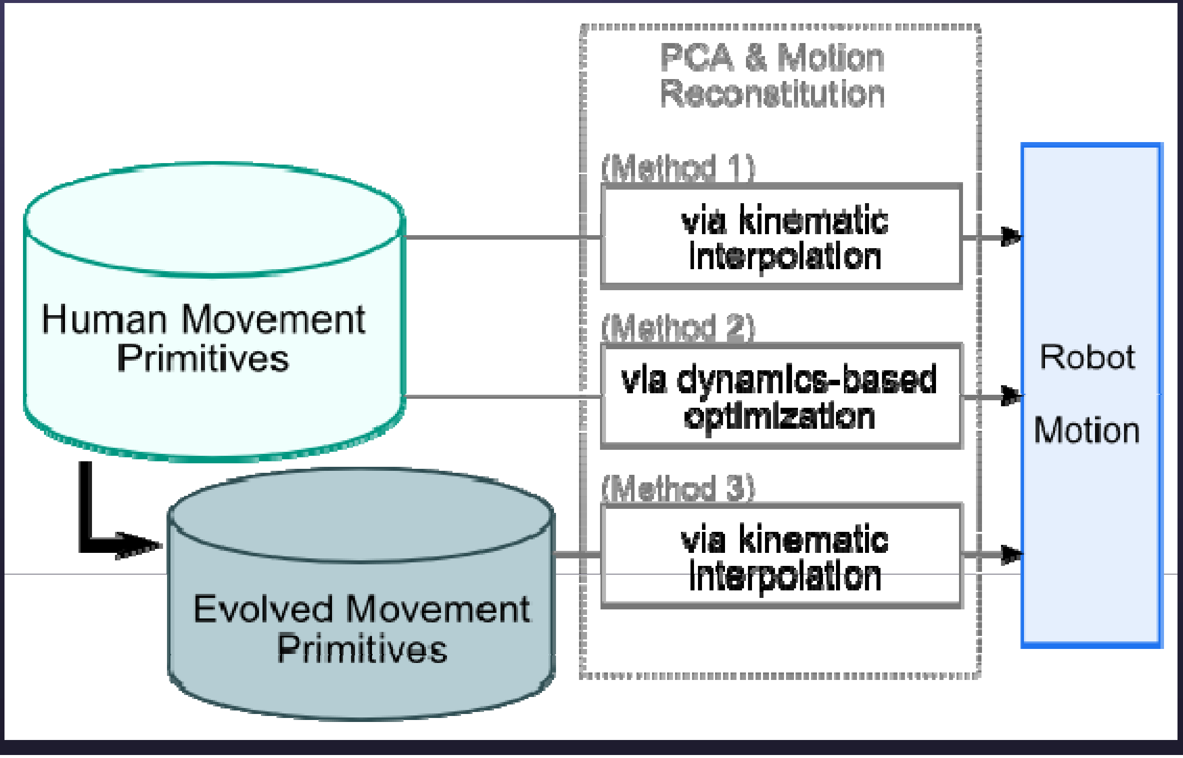
Standing (left), catching (center, right)

Catching points (blue dot)

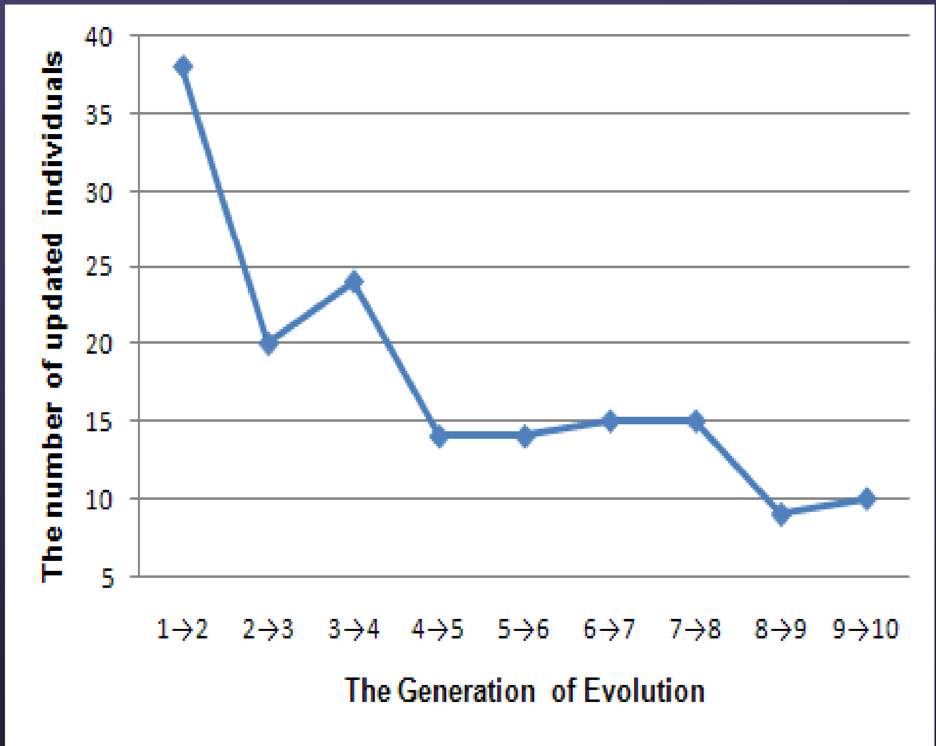


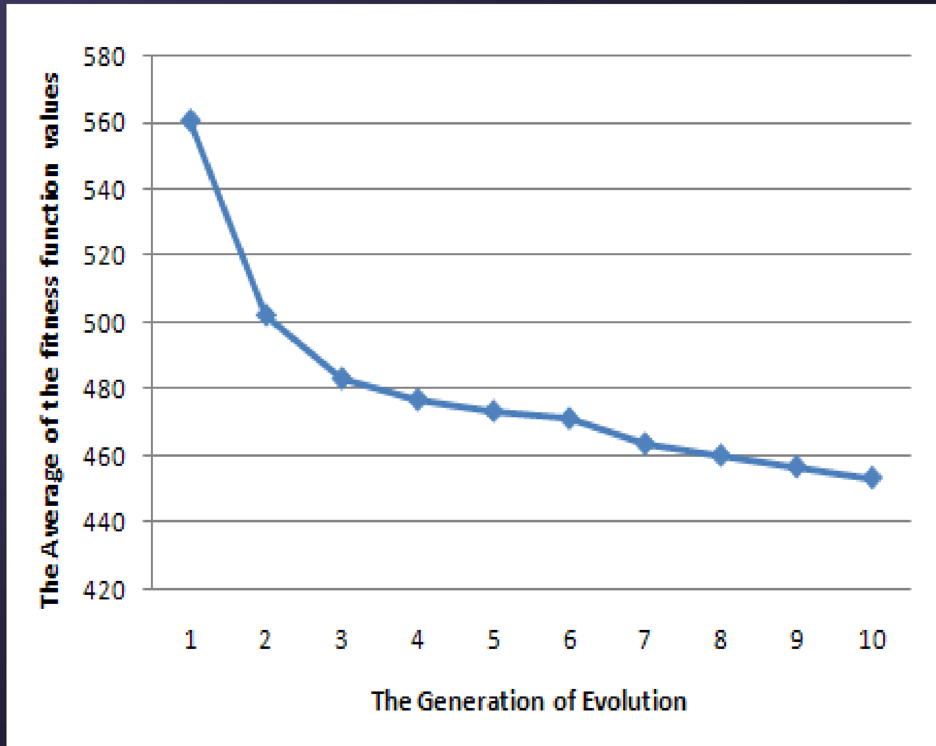
Three Methods

- Human movement primitives
  - kinematic interpolation → METHOD 1
  - dynamics-based optimization → METHOD 2
- Evolved Movement Primitives
  - kinematic interpolation → METHOD 3



Evolution Result





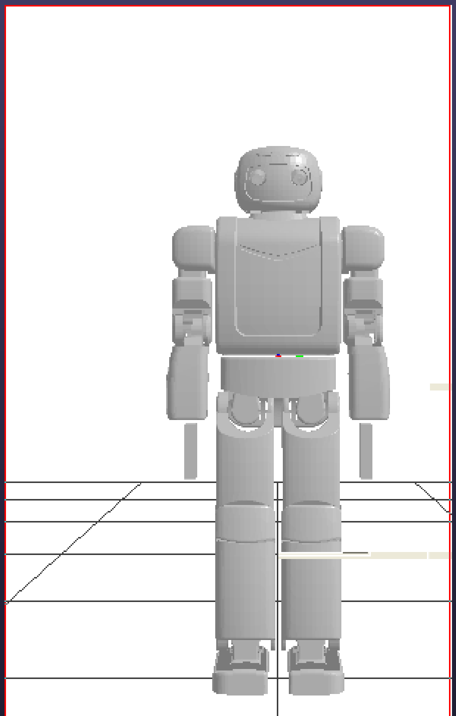
the number of updated individuals

average of fitness function value

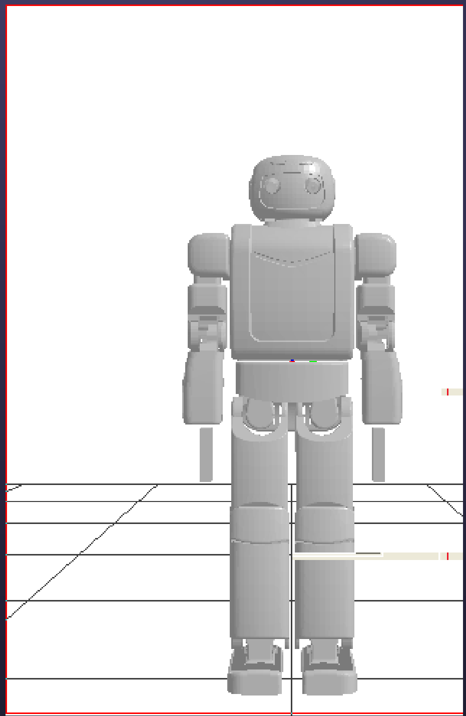
Kinematic Interpolation on Human movement (method 1) v.s Kinematic Interpolation on Evolved movement (method 3)

	Method 1	Method 3
Motion Generation Time	0.092 sec	0.101 sec
Torque-Consumption	370.0	275.3

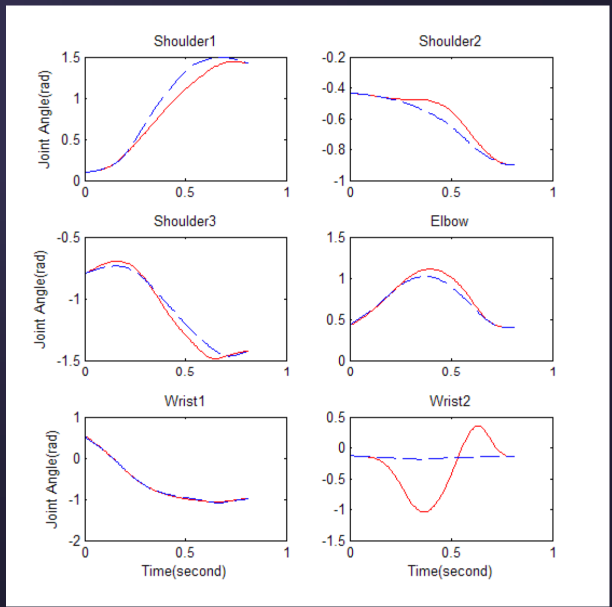
† C++ on Pentium 4 PC



Method 1



Method 3

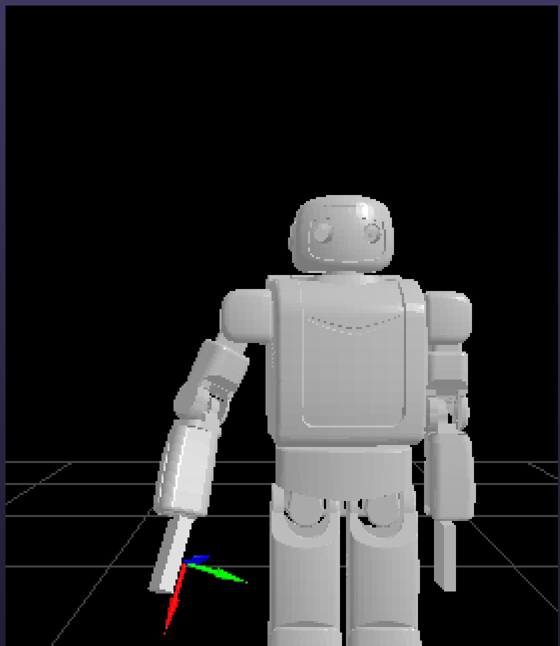


Joint trajectory
(blue : method 1, red : method 3)

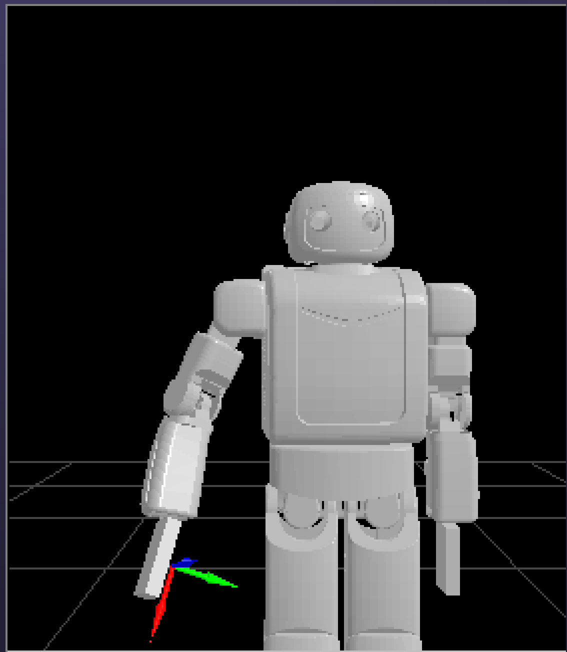
Dynamics-based Optimization on Human movement (method 2) v.s Kinematic Interpolation on Evolved movement (method 3)

	Method 2	Method 3
Motion Generation Time	11.32 sec	0.127 sec
Torque-Consumption	348.7	385.1

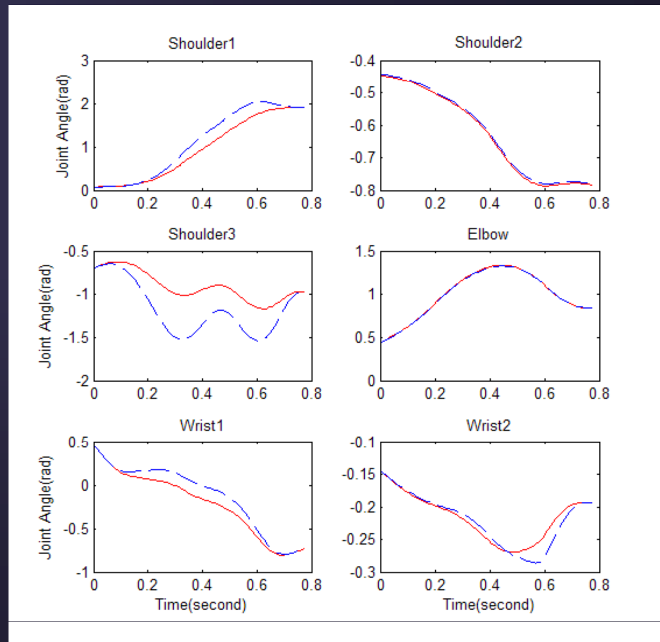
† C++ on Pentium 4 PC



Method 2



Method 3



Joint trajectory
(blue : method 2, red : method 3)

Method 1 vs. 2 vs. 3

- Generate arbitrary 10 catching motions
- Compare the average performances

	Method 1	Method 2	Method 3
Motion Generation Time	0.109 sec	13.21 sec	0.115 sec
Torque-Consumption	498.7	372.6	428.4

† C++ on Pentium 4 PC

- Result
  - Evolved movement primitives are sufficiently optimal
  - Generating motions from evolved movement primitives has light computational burden

Conclusion

- A framework for representing movement primitives based on PCA of motion data.
- Efficient storage, near real-time generation of optimal motions that resemble motion data.
- inverse dynamics-based optimization during repeated practices and storage and retrieval of a great variety of optimal movement patterns.
- Ongoing work : more sophisticated and full-body motions