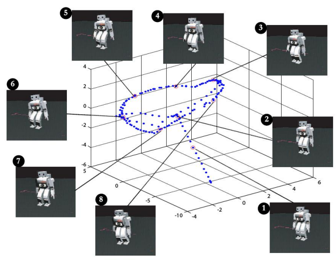
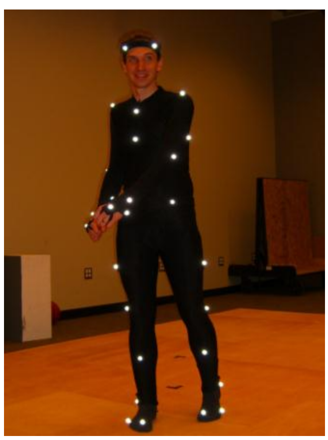
CSE 481C

Imitation Learning in Humanoid Robots

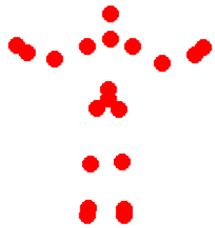
Motion capture, inverse kinematics, and dimensionality reduction



Robotic Imitation of Human Actions



The inverse kinematics problem



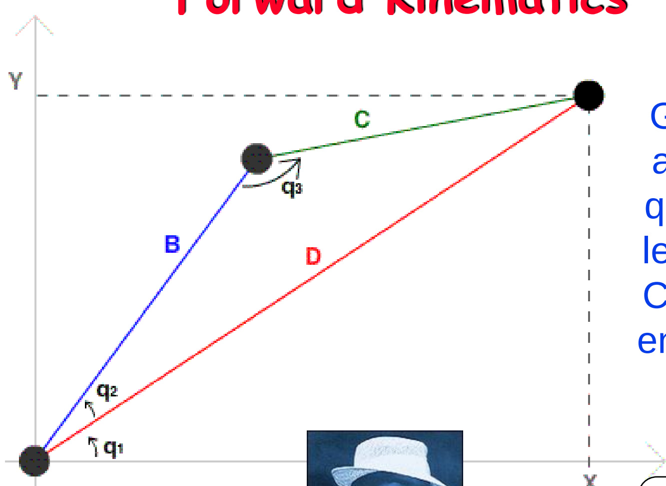
Joint angles

Human-robot skeleton mapping

Robot joint angles

3

Robot arm example: Forward Kinematics

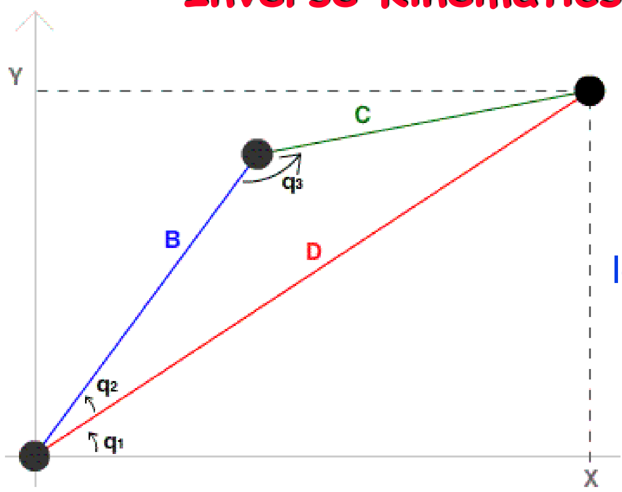


Given joint angles q1, q2, q3, and lengths B & C, compute end position (X,Y).

Dat's trivi y'all!

4

Robot arm example: Inverse Kinematics

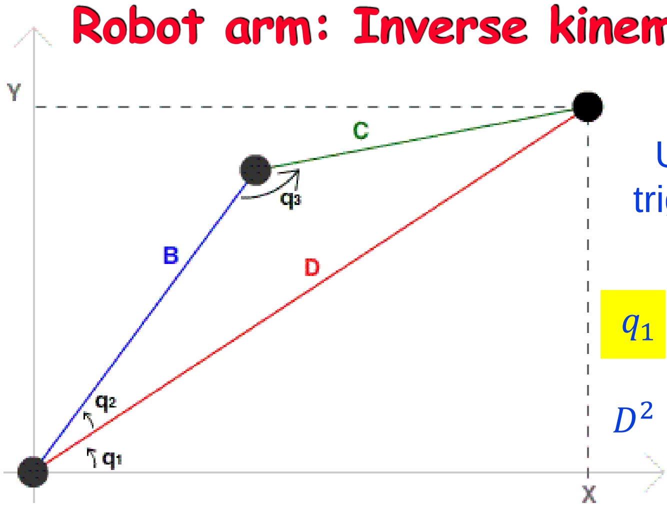


Assume you know (X,Y) and the lengths B & C

How do you compute joint angles q1, q2, q3?

5

Robot arm: Inverse kinematics



Use basic trigonometry

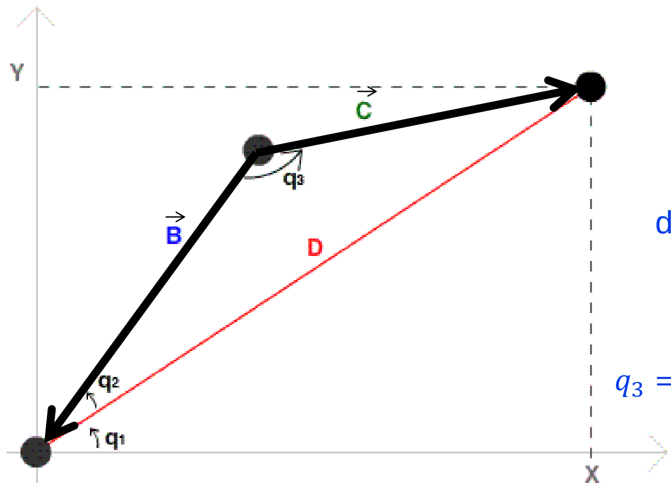
$$q_1 = \tan^{-1}\frac{Y}{X}$$

$$D^2 = X^2 + Y^2$$

Law of cosines

$$D^2 = B^2 + C^2 - 2BC\cos q_3$$
$$C^2 = B^2 + D^2 - 2BD\cos q_2$$

Robot arm: Direct method



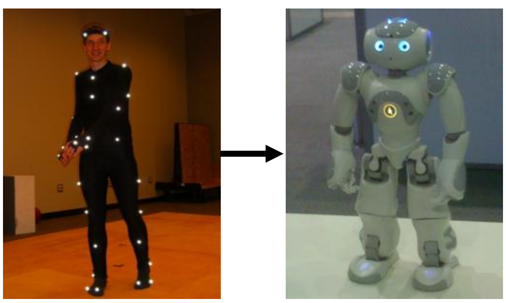
If you know all joint positions, can estimate joint angles directly using the "bone" vectors for B and C

$$q_3 = cos^{-1} \frac{\vec{B} \cdot \vec{C}}{\|\vec{B}\|\|\vec{C}\|}$$

See http://www.seethroughskin.com/blog/?p=1186 for an example

7

Problem 2: High dimensionality



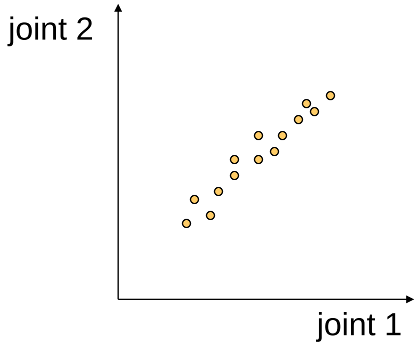
25-dimensional joint angle vector

Makes learning and optimization intractable

But…many angles are correlated

Can we reduce the representation to a more tractable dimensionality?

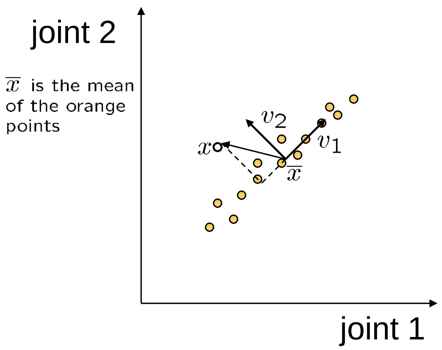
A Simple 2D Example



- What does the plot above suggest?
- The two joints are highly correlated for this dataset of robotic motions

(Adapted from Steve Seitz, Linda Shapiro) 9

Linear subspaces



Suppose we fit a line v_1
Let v_2 be orthogonal to v_1

Convert an input **x** into $\mathbf{v_1}$, $\mathbf{v_2}$ coordinates

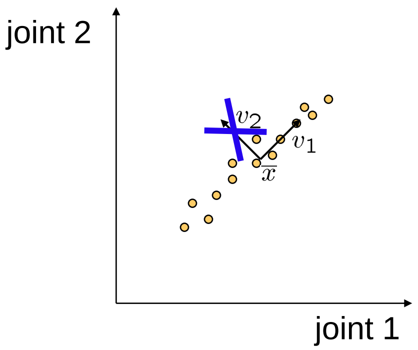
$$\mathbf{x} \rightarrow ((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2})$$

What does the $\mathbf{v_1}$ coordinate measure?
- position along v_1 axis
- use it to specify which point it is

What does the $\mathbf{v_2}$ coordinate measure?
- distance to line (position along v_2 axis)
- near 0 for these pts

10

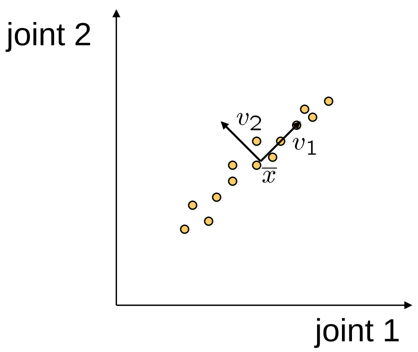
Dimensionality reduction: 2D to 1D



- We can represent the points with *only* their $\mathbf{v_1}$ coordinates
  – since $\mathbf{v_2}$ coordinates are all essentially 0
– Reduce dimensionality of data from 2D to 1D
- This makes it cheaper to perform computations on the points
- Bigger deal for higher dimensional inputs (like robot joint space!)

11

How do we find v_1, v_2, ... ?



Consider the variation along some direction **v** for all of the *N* points:

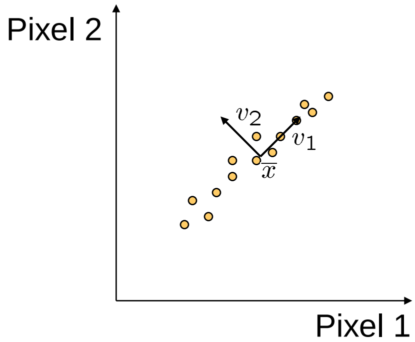
$$var(\mathbf{v}) = 1/N \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^T \mathbf{v}\|^2$$

Find unit vector **v** maximizing *var:* $\mathbf{v}_1 = \arg\max_{\mathbf{v}} \{var(\mathbf{v})\}$

$\mathbf{v_2}$ is then the unit vector orthogonal to $\mathbf{v}_1$

12

How do we find v_1, v_2, … ?



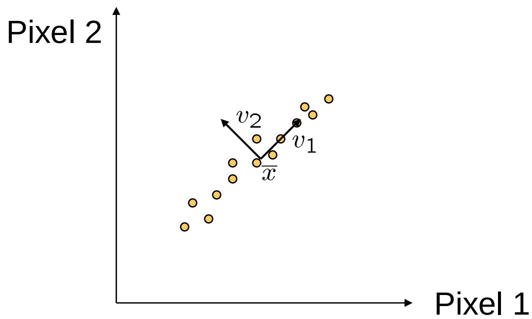
$$
\begin{aligned}
var(\mathrm{v}) &= \sum_{\mathrm{x}} \|(\mathrm{x}-\overline{\mathrm{x}})^{\mathrm{T}}\ \mathrm{v}\|^2 /N \\
&= \sum_{\mathrm{x}} \mathrm{v}^{\mathrm{T}}(\mathrm{x}-\overline{\mathrm{x}})(\mathrm{x}-\overline{\mathrm{x}})^{\mathrm{T}}\mathrm{v}\ /N \\
&= \mathrm{v}^{\mathrm{T}}\left[\sum_{\mathrm{x}}(\mathrm{x}-\overline{\mathrm{x}})(\mathrm{x}-\overline{\mathrm{x}})^{\mathrm{T}}\right]\mathrm{v}\ /N \\
&= \mathrm{v}^{\mathrm{T}}\mathbf{A}\mathrm{v} \quad \text{where } \mathbf{A} = \sum_{\mathrm{x}}(\mathrm{x}-\overline{\mathrm{x}})(\mathrm{x}-\overline{\mathrm{x}})^{\mathrm{T}}/N
\end{aligned}
$$

A = Covariance matrix of data points

We want to find a unit vector **v** that maximizes $\mathbf{v}^T\mathbf{A}\ \mathbf{v}$

13

Finding v_1 and v_2: The Math



$\mathbf{v_1}$ = argmax$_{\mathbf{v}}$ ($\mathbf{v}^T\mathbf{A}\ \mathbf{v}$) subject to $\mathbf{v}^T\mathbf{v} = 1$

Using Lagrange multiplier method,

$\mathbf{v_1}$ = argmax$_{\mathbf{v}}$ $[\mathbf{v}^T\mathbf{A}\ \mathbf{v} - \lambda(\mathbf{v}^T\mathbf{v} - 1)]$

Setting derivative wrt **v** to 0, we get:

$\mathbf{Av} = \lambda\mathbf{v}$ Thus, $\mathbf{v_1}$ is eigenvector of **A** with *largest* eigenvalue λ_1

$\mathbf{v_2}$ is eigenvector of **A** with *smaller* eigenvalue λ_2

14

Principal Component Analysis (PCA)

Suppose each of the N data points is L-dimensional

- Form L x L data covariance matrix A

$$A = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$$

- Compute eigenvectors of *A*
  - Eigenvectors of *A* define a new coordinate system that is a rotation of the original coordinate system
  - Eigenvector with largest eigenvalue captures the most variation among training vectors x
  - Eigenvector with smallest eigenvalue has least variation

15

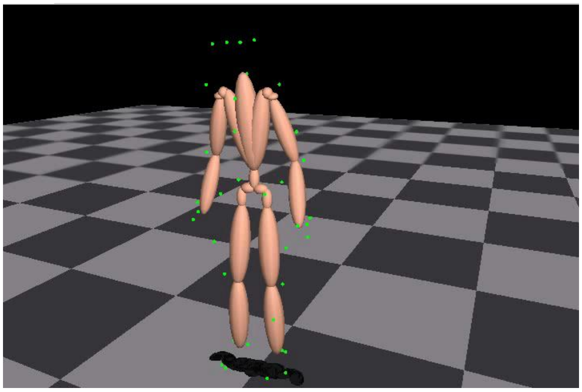
Dimensionality Reduction using PCA

We can reduce dimensionality by only using the top few eigenvectors with largest eigenvalues

- corresponds to choosing a “linear subspace” of the original data space
- represent points on a line, plane, “hyper-plane”
- these eigenvectors are known as *principal component vectors*
- procedure is known as *Principal Component Analysis (PCA)*

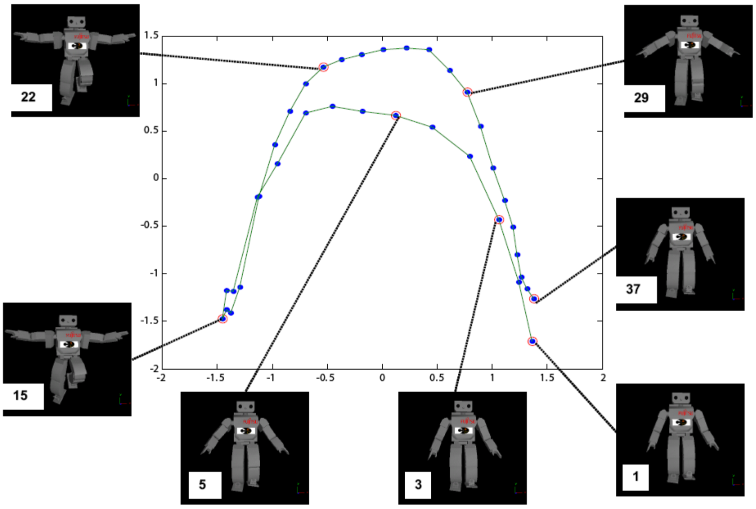
16

Applying PCA to Human Motion Capture Data

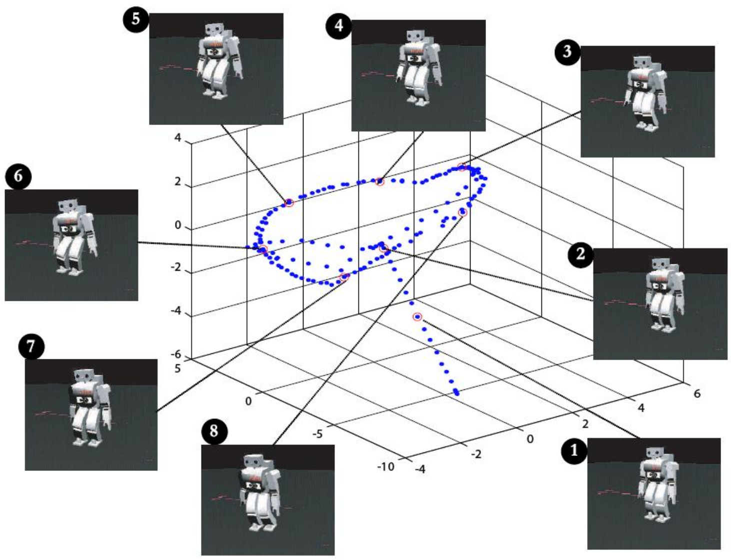


Human Motion Capture Sequence

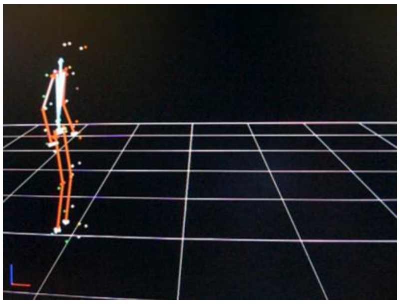
Reduced-Dimension Representation using Eigenposes



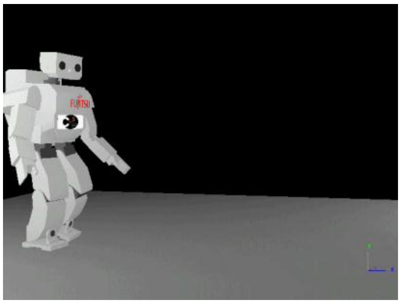
Eigenposes for Walking



Learning to Walk

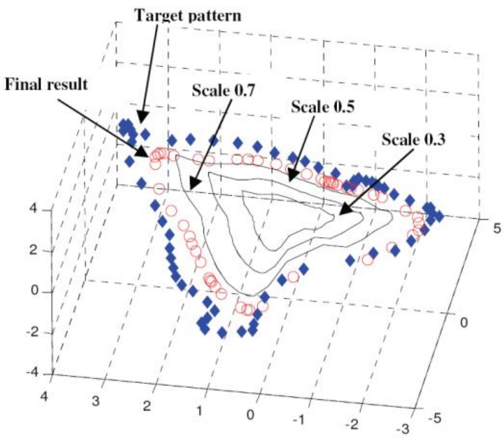


Human motion capture data

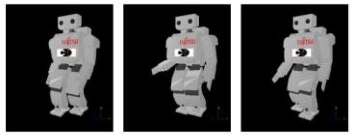


Unoptimized (kinematic) imitation

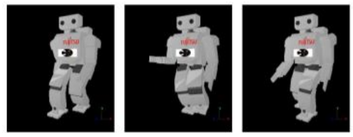
Learning to Walk



Motion scaling
(Take baby steps first)

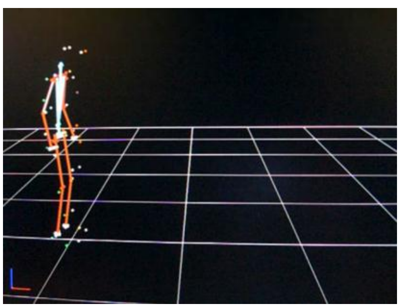


Final Result

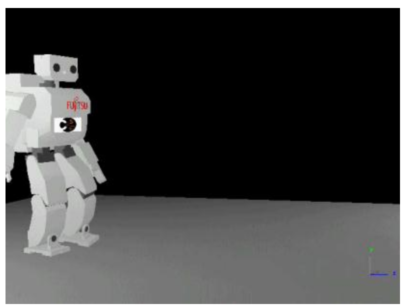


(Chalodhorn et al., IJCAI 2007)

Result: Learning to Walk



Human Motion Capture



Optimized Stable Walk

(Chalodhorn et al., IJCAI 2007)

Next class: Machine learning and probabilistic reasoning

Today's Goal:

Finish SDK installation and test Kinect, NAO simulator, and NAO robot (with Mike's help)

Begin Warm-Up Project #1

(details on the course website)