Carnegie Mellon

School of Computer Science

Efficient Parallel Learning of Linear Dynamical Systems on SMPs

Lei Li

Computer Science Department

School of Computer Science

Carnegie Mellon University

leili@cs.cmu.edu

Motion stitching via effort minimization
with James McCann, Nancy Pollard and
Christos Faloutsos
[Eurographics 2008]

Parallel learning of linear dynamical systems
with Wenjie Fu, Fan Guo, Todd Mowry and
Christos Faloutsos
[KDD 2008]

Background

- Motion Capture

- Markers on human body, optical cameras to capture the marker positions, and translated into body local coordinates.
- Application:
  – Movie/game/medical industry

Outline

- Background
- Motivation: effortless motion stitching
- Parallel learning with Cut-And-Stitch
- Experiments and Results
- Conclusion

Motivation

- Given two human motion sequences, how to stitch them together in a natural way(= looks natural in human’s eyes)?

e.g. walking to running

- Given a human motion sequence, how to find the best natural stitchable motion in motion capture database?

Intuition

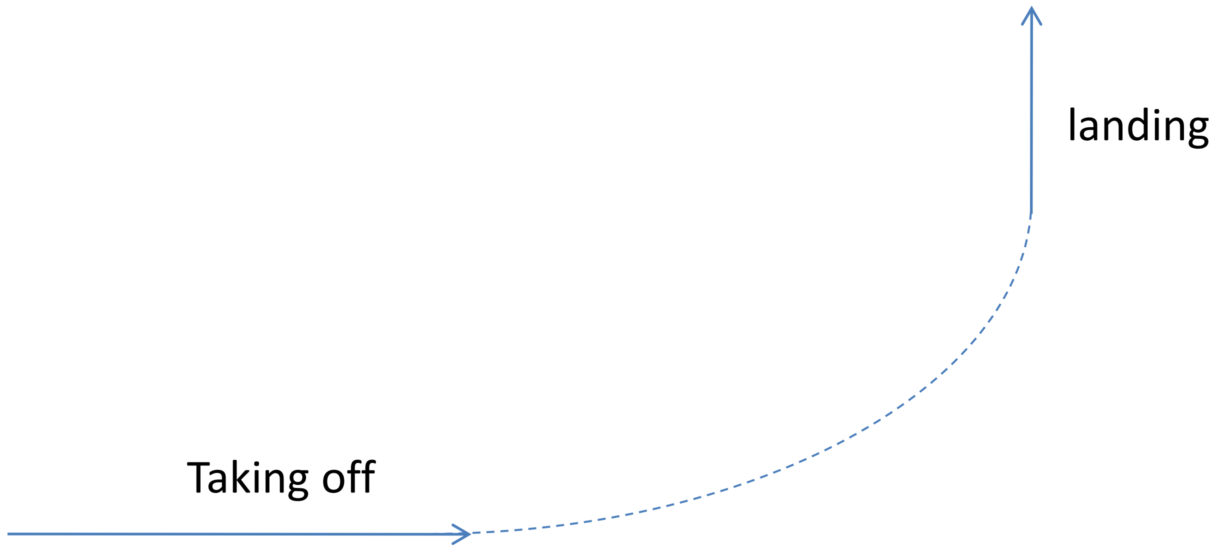
- Intuition:
  - Laziness is a virtue. Natural motion use minimum energy
- Laziness-score (L-score) = energy used during stitching
- Objective:
  - Minimize laziness-score

Example

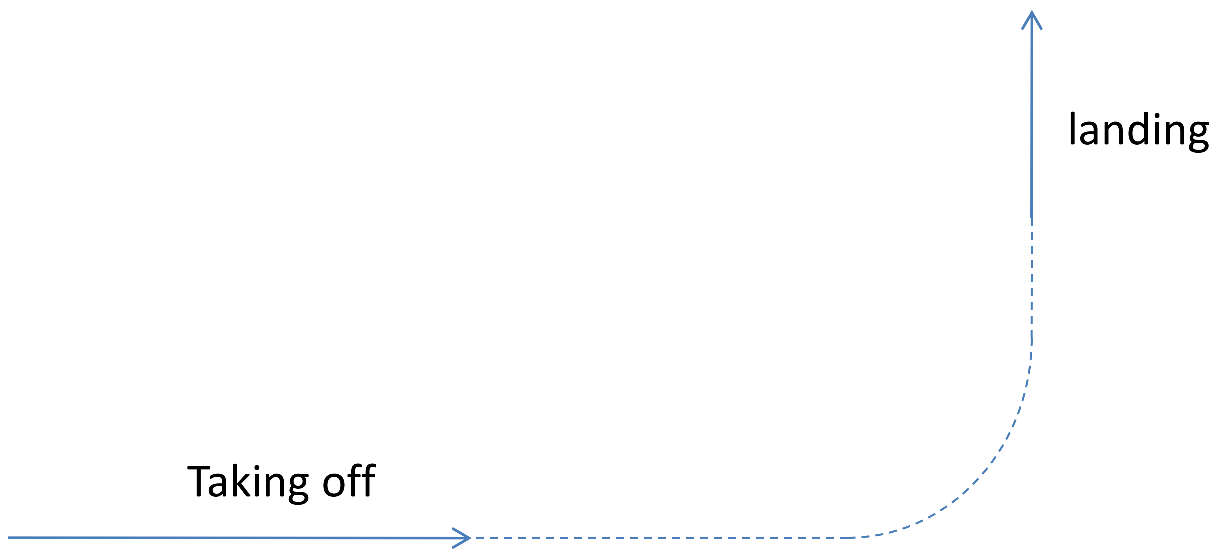
landing

Taking off

Example, Natural stitching



But, how about this way?

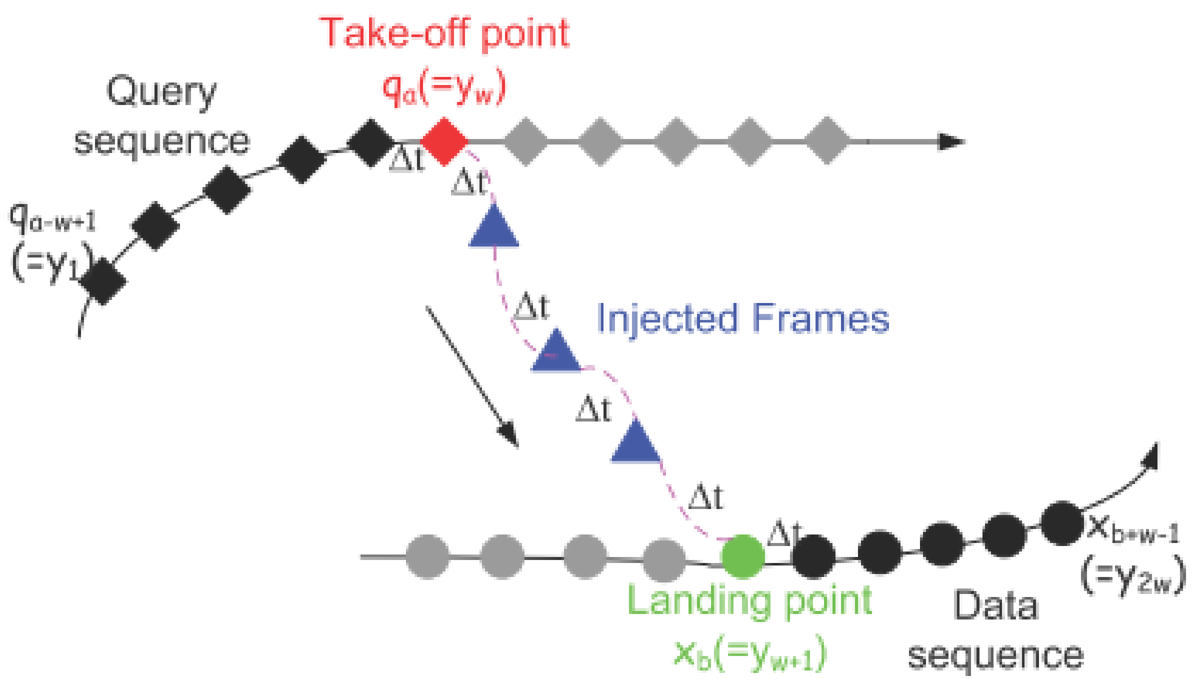


Observations

- Naturalness depends on smoothness
- Naturalness also depends on motion speed

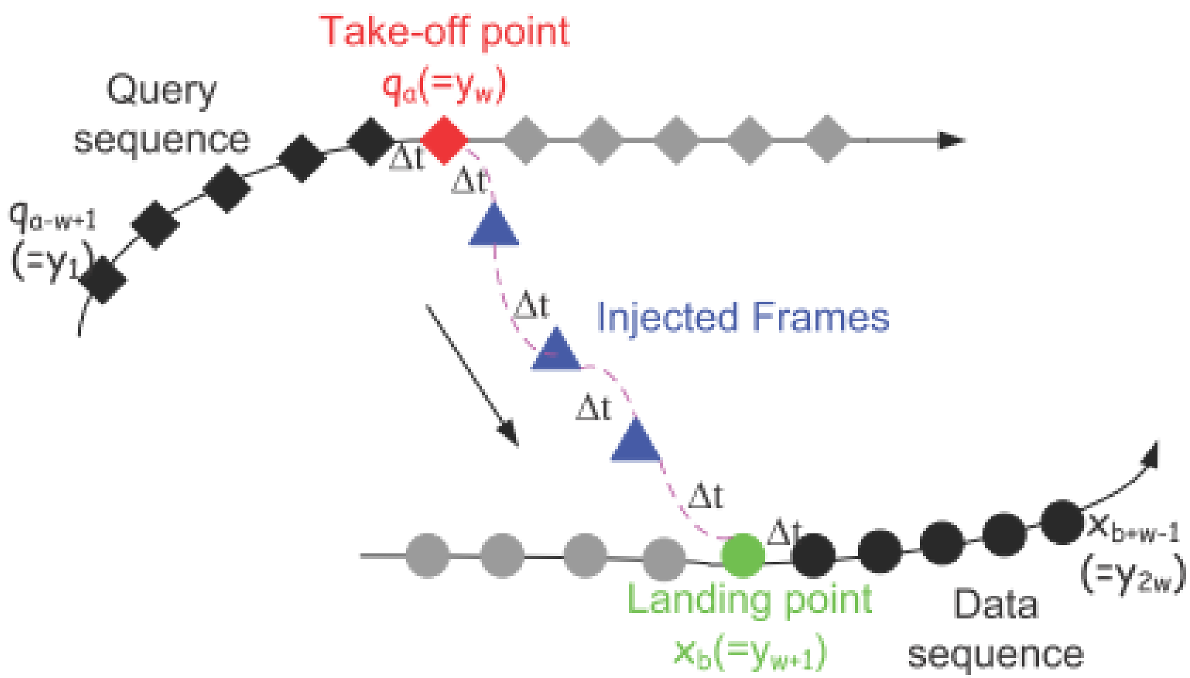
Proposed Method

- Estimate stitching path using Linear Dynamical Systems



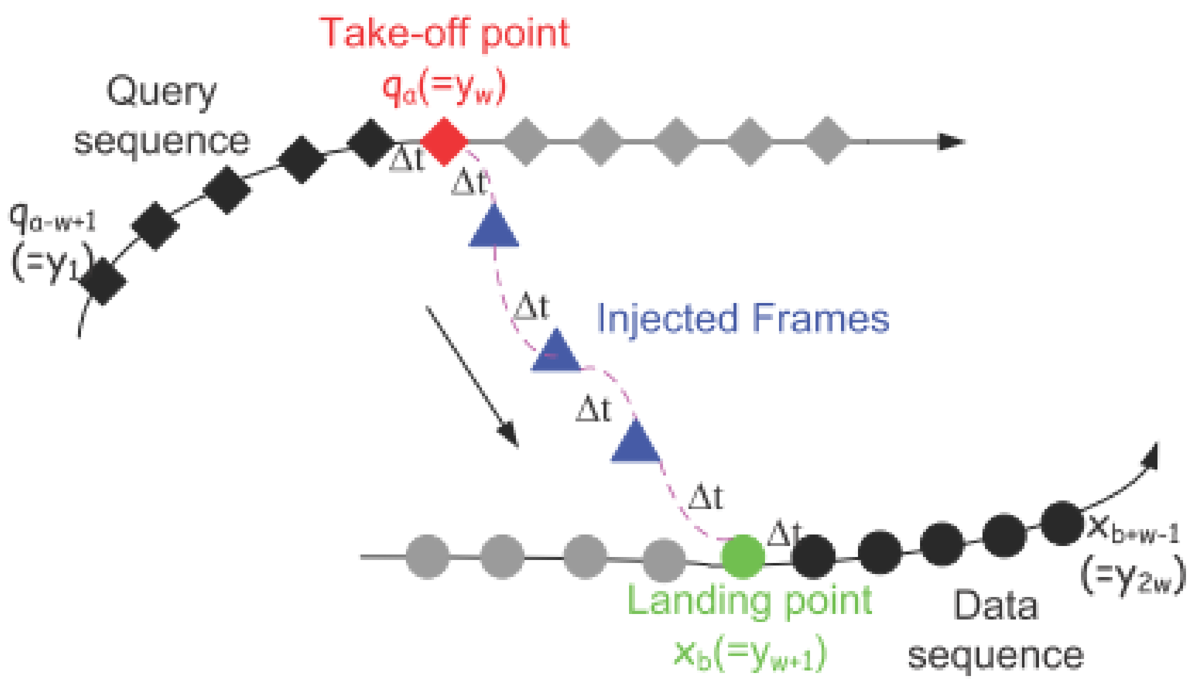
Proposed Method (cont')

- Estimate the velocity and acceleration during the stitching, compute energy (defined as L-score)

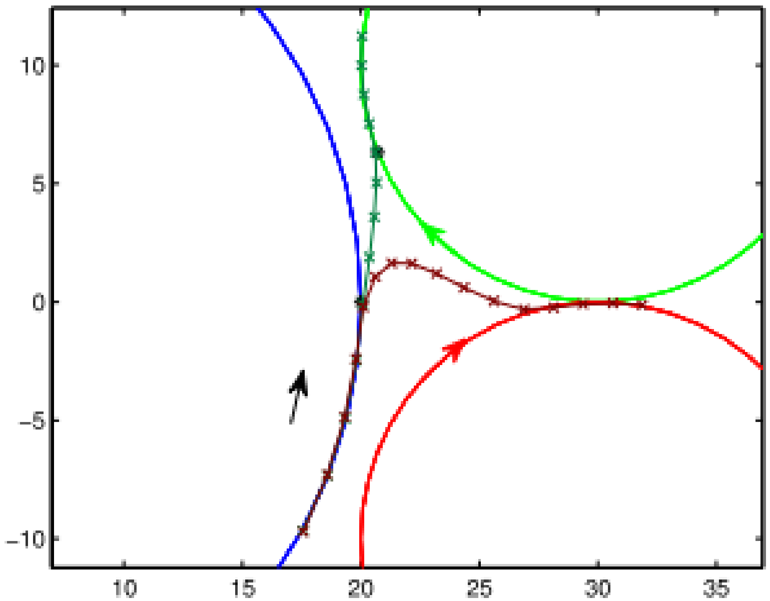


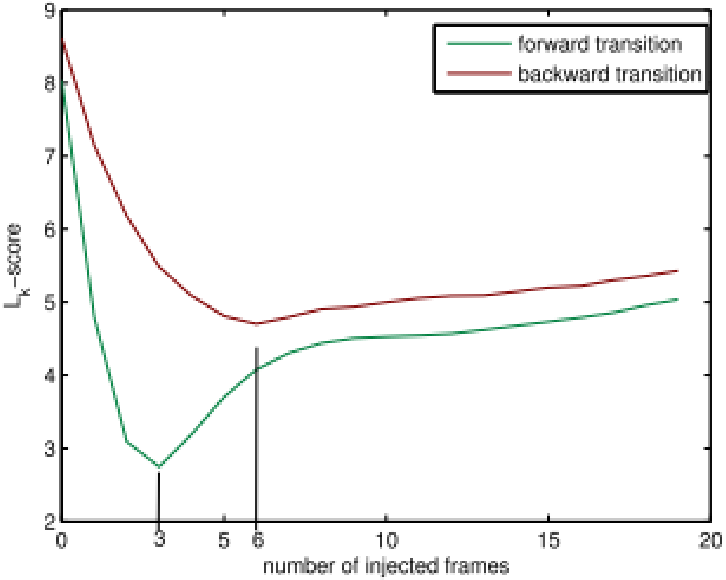
Proposed Method (cont')

- Minimize L-score with respect to any stitching hops. (defined as elastic L-score)



Example stitching





- [Link to video]()

Outline

- Background
- Motivation: effortless motion stitching
- Parallel learning with Cut-And-Stitch
- Experiments and Results
- Conclusion

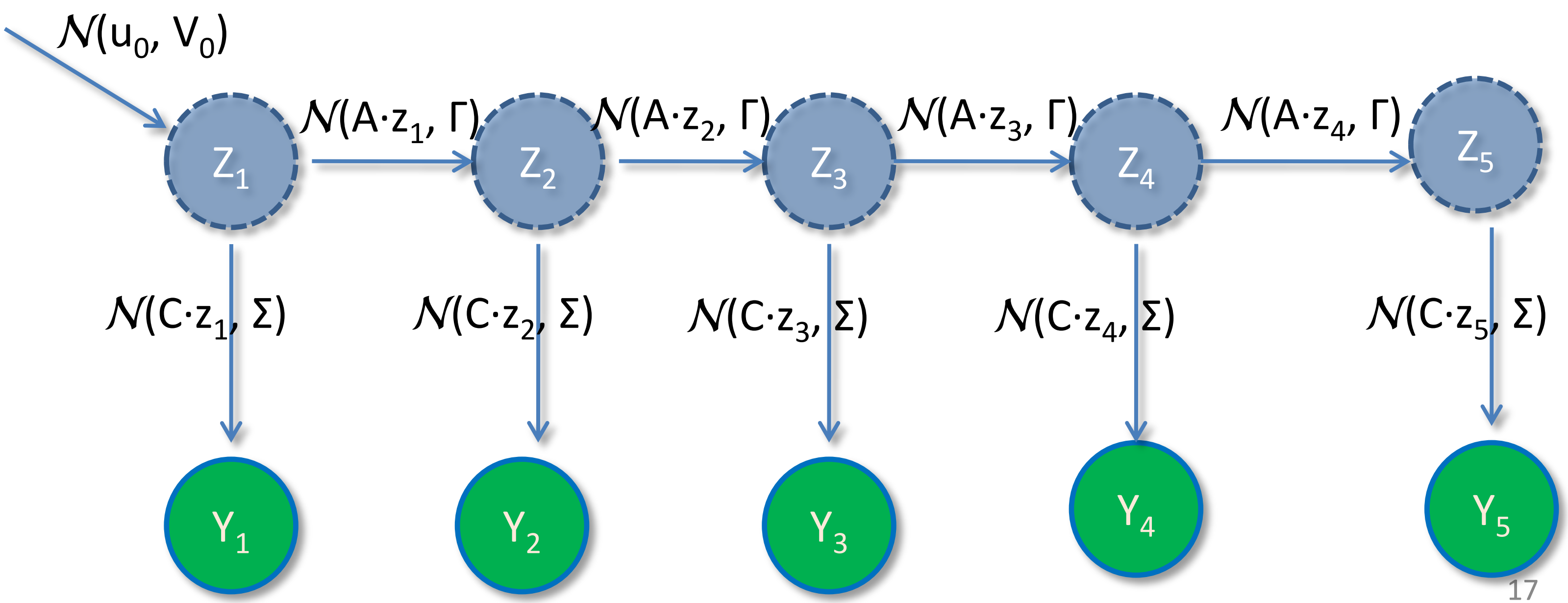
Parallel Learning for LDS

- Challenge:
  - Learning Linear Dynamical System is slow for long sequences
- Traditional Method:
  - *Maximum Likelihood* Estimation via *Expectation-Maximization*(EM) algorithm
- Objective:
  - **Parallelize** the learning algorithm
- Assumption:
  - *shared memory architecture*

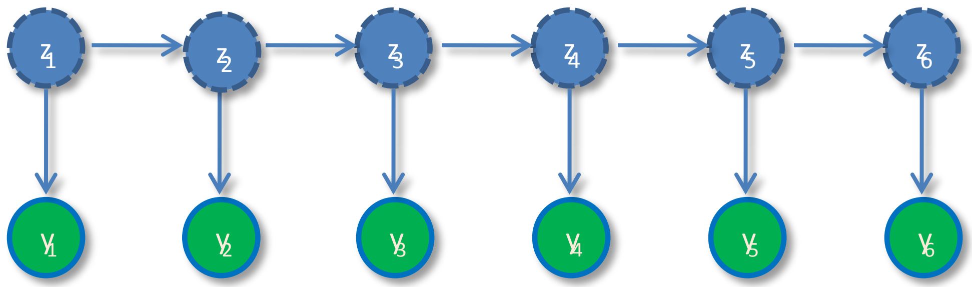
Linear Dynamical System

aka. Kalman Filter

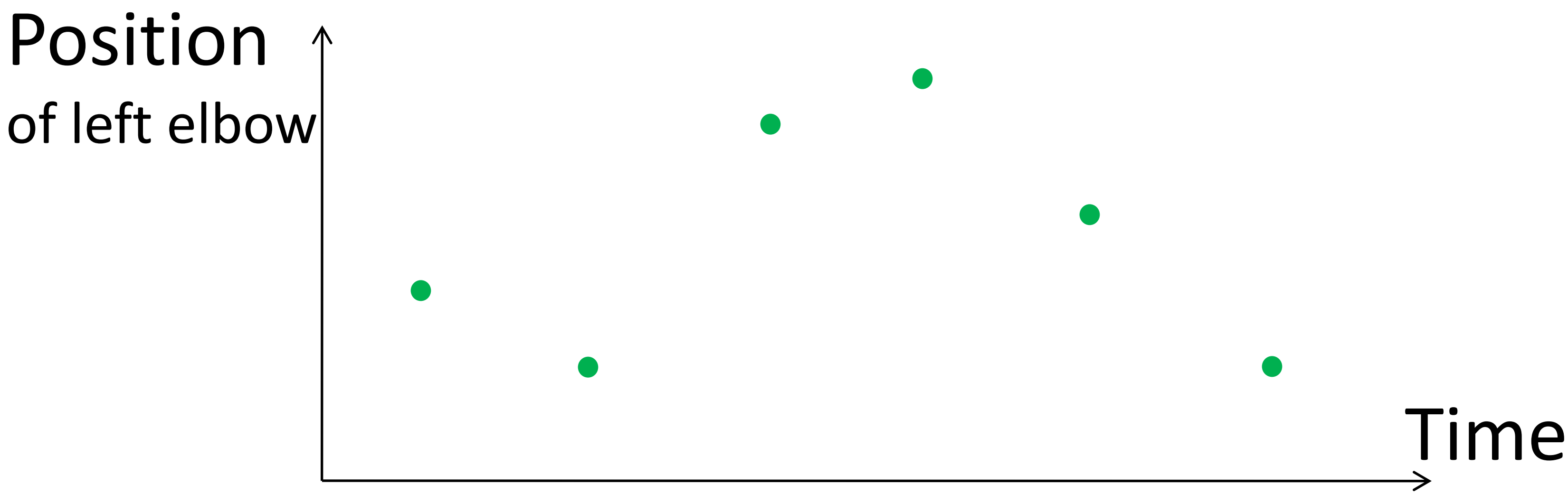
- Parameters: $\theta=(u_0, V_0, A, \Gamma, C, \Sigma)$
- Observation: $y_1 \ldots y_n$
- Hidden variables: $z_1 \ldots z_n$



Example

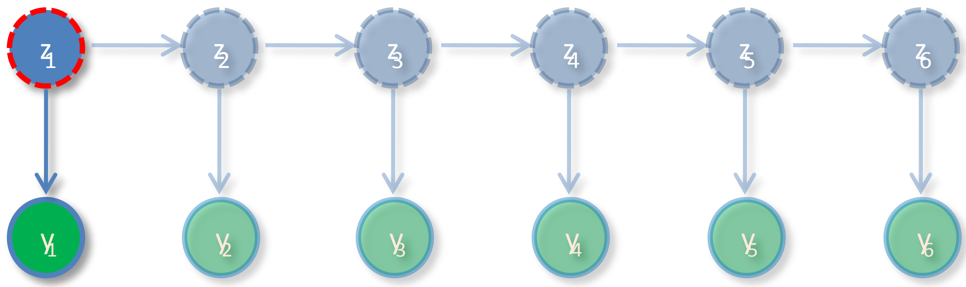


given positions, estimate dynamics (i.e. params)

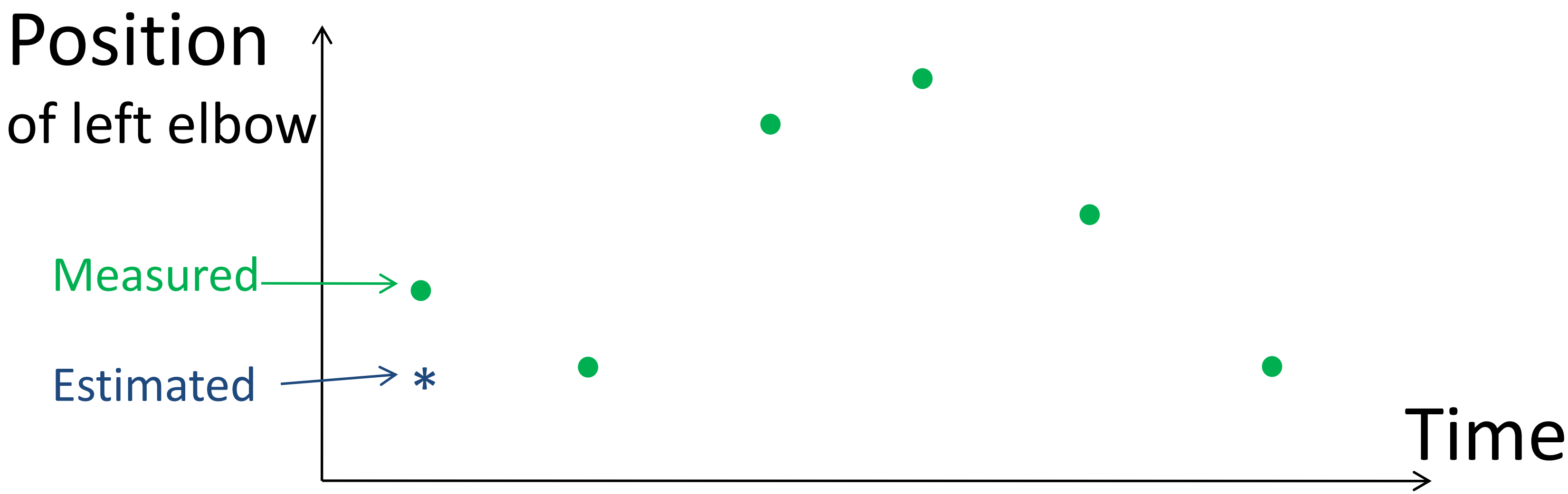


Traditional:
How to learn LDS?

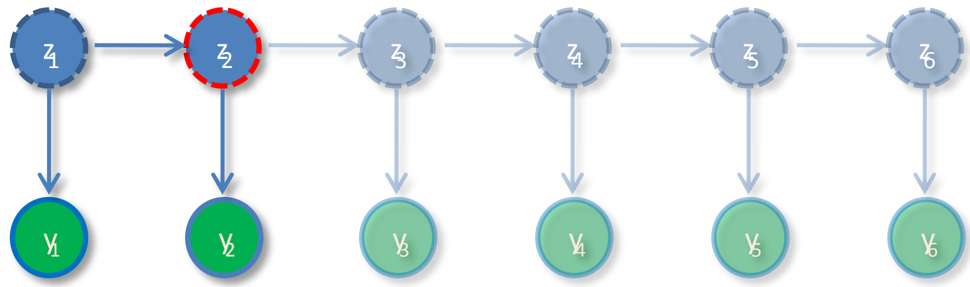
Sequential Learning (EM)



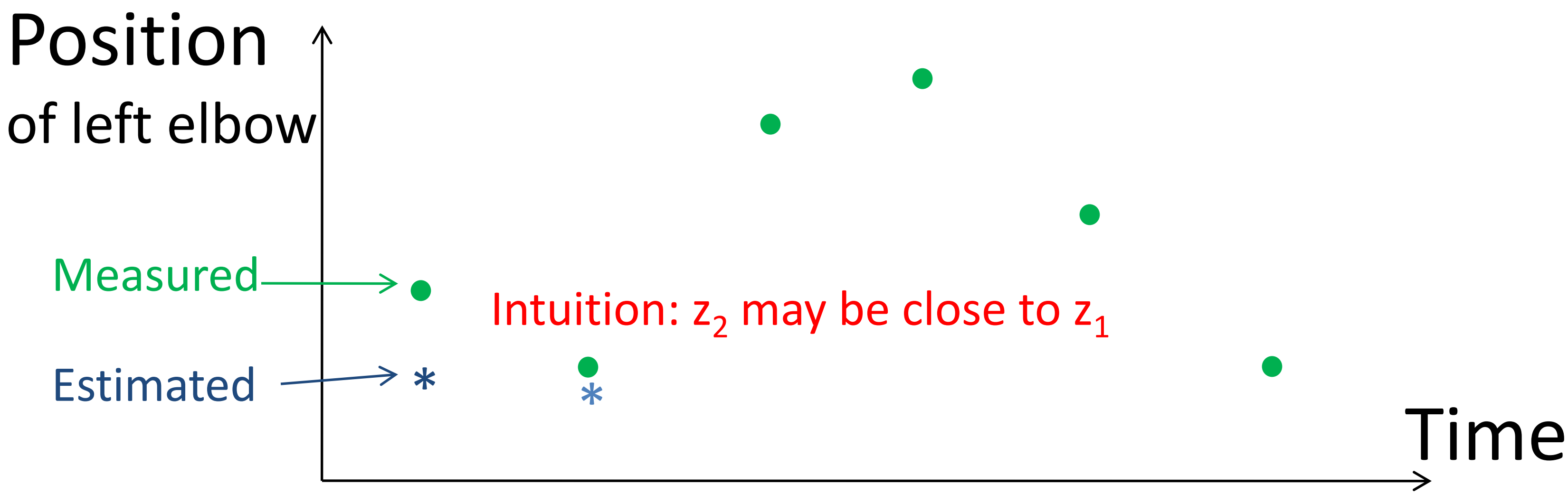
Compute $P(z_1 \mid y_1)$



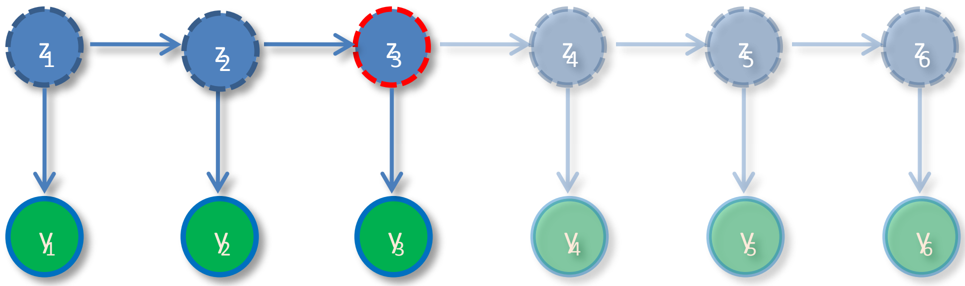
Sequential Learning (EM)



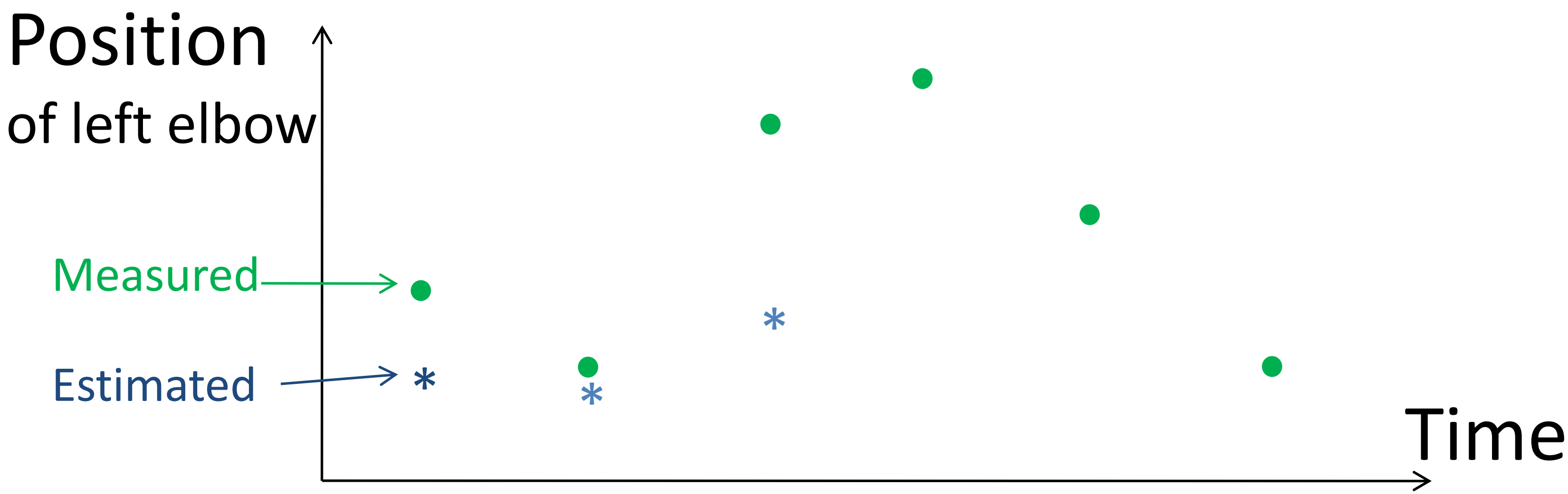
From $P(z_1 \mid y_1) \rightarrow$ Compute $P(z_2 \mid y_1, y_2)$



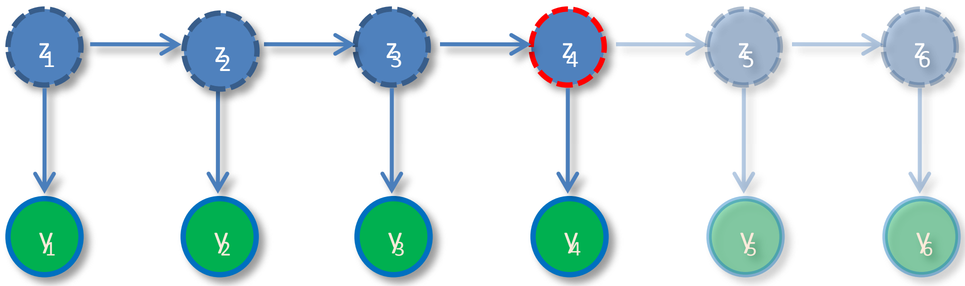
Sequential Learning (EM)



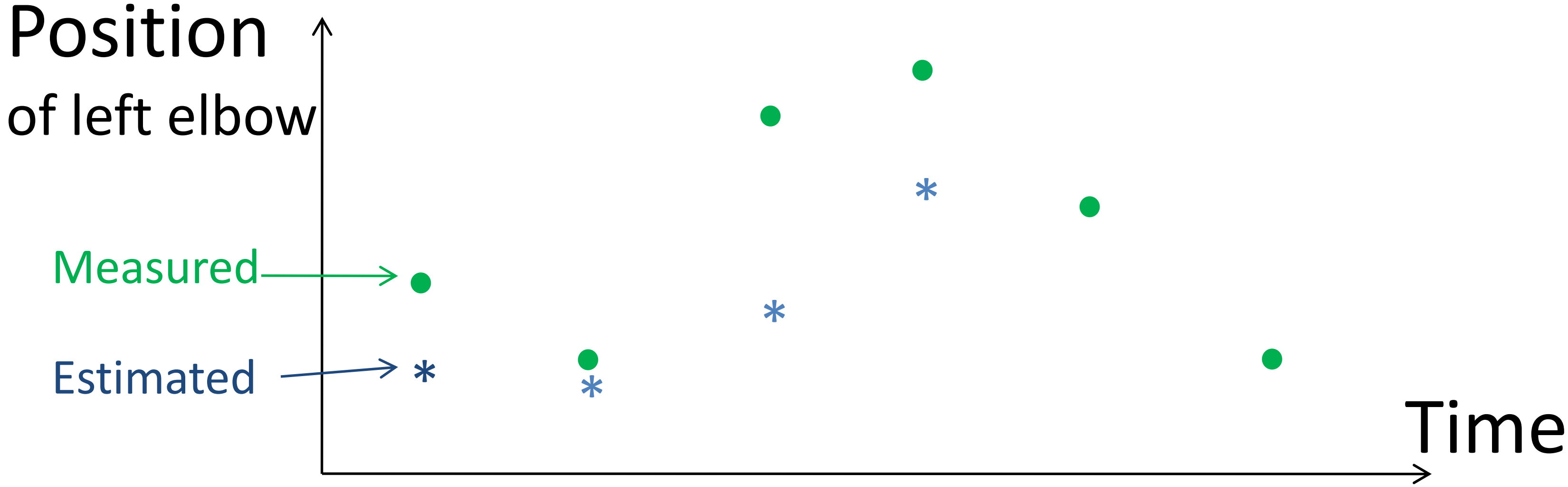
From $P(z_2 | y_1, y_2)$ → Compute $P(z_3 | y_1, y_2, y_3)$



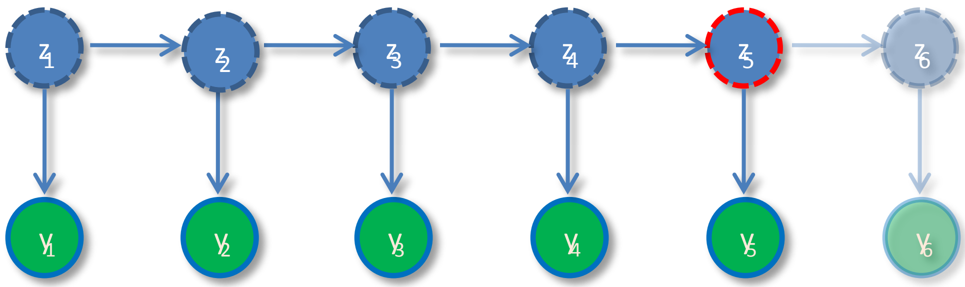
Sequential Learning (EM)



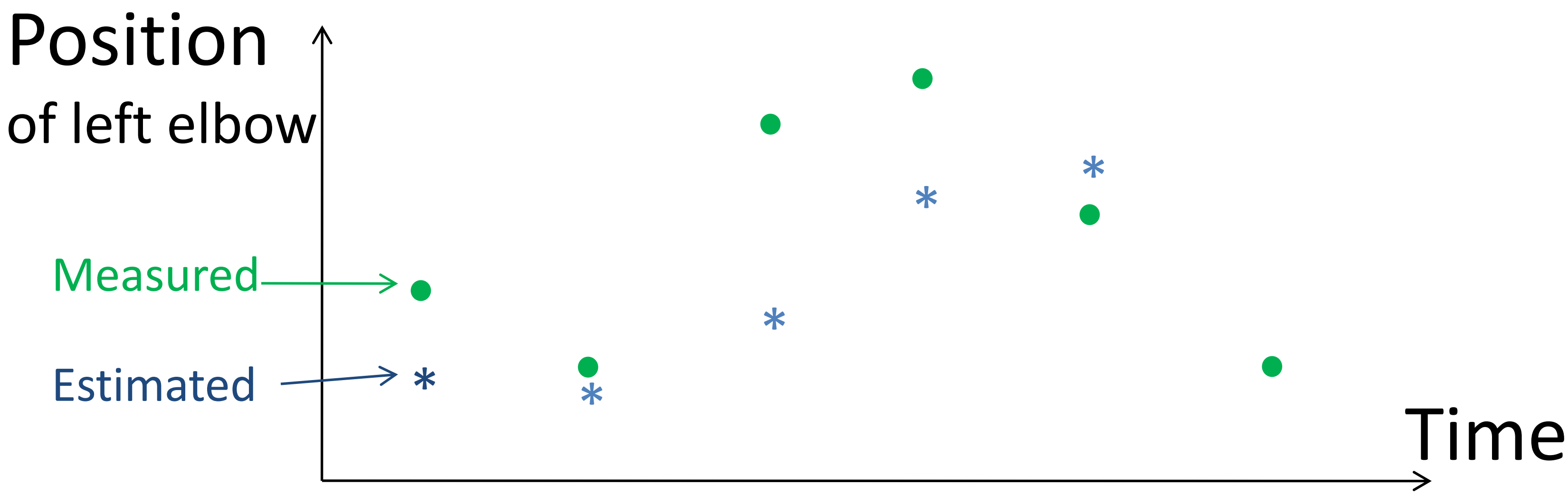
From $P(z_3 | y_1, y_2, y_3) \rightarrow$ Compute $P(z_4 | y_1, y_2, y_3, y_4)$



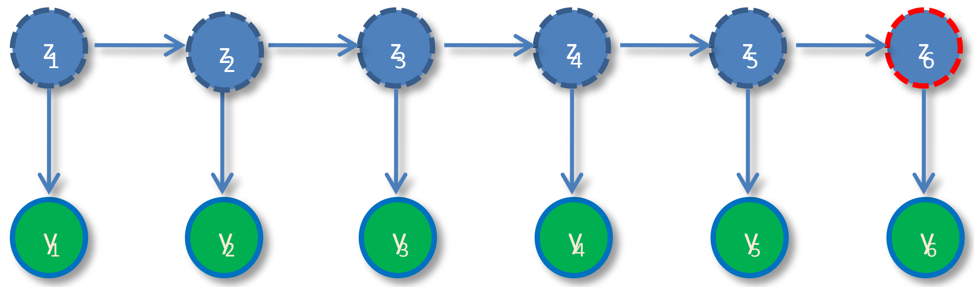
Sequential Learning (EM)



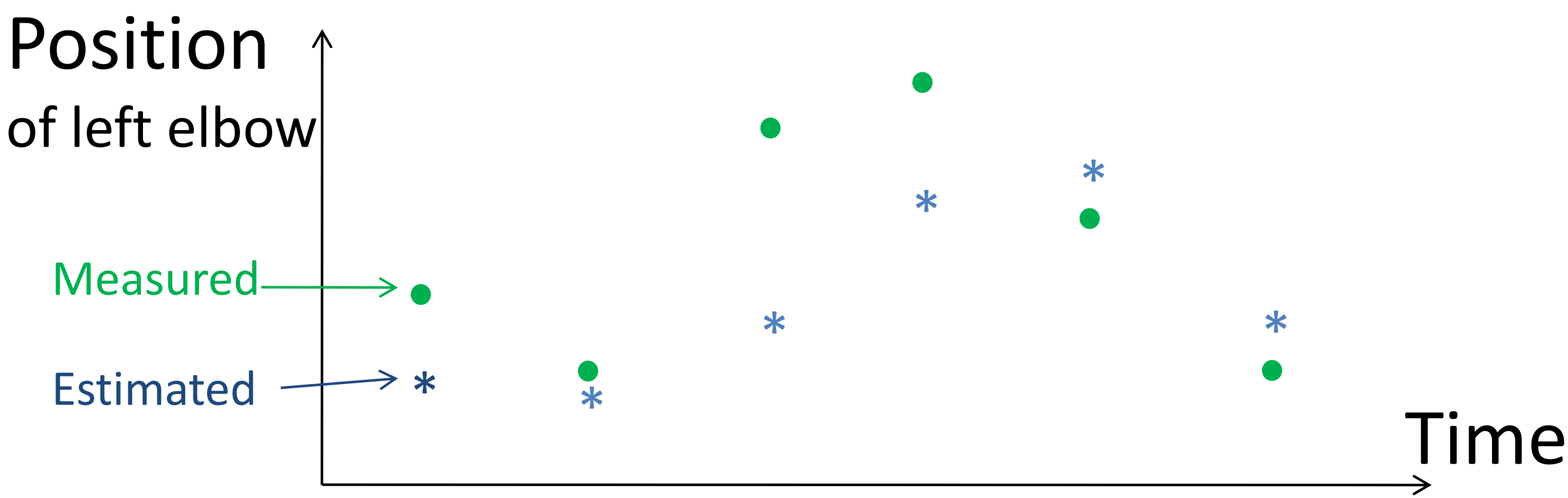
From $P(z_4 | y_1, y_2, y_3, y_4) \rightarrow$ Compute $P(z_5 | y_1, y_2, y_3, y_4, y_5)$



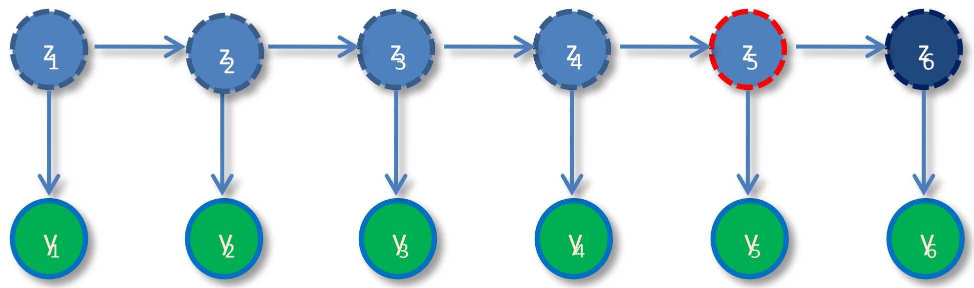
Sequential Learning (EM)



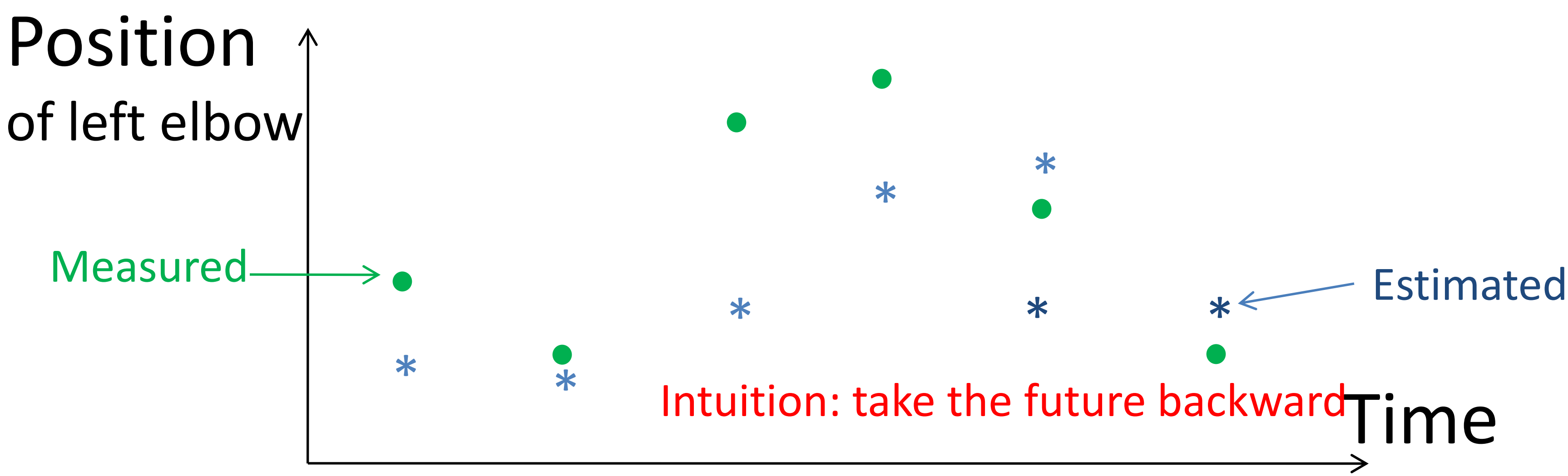
From $P(z_5 | y_1, y_2, y_3, y_4, y_5)$ → Compute $P(z_6 | y_1, y_2, y_3, y_4, y_5, y_6)$



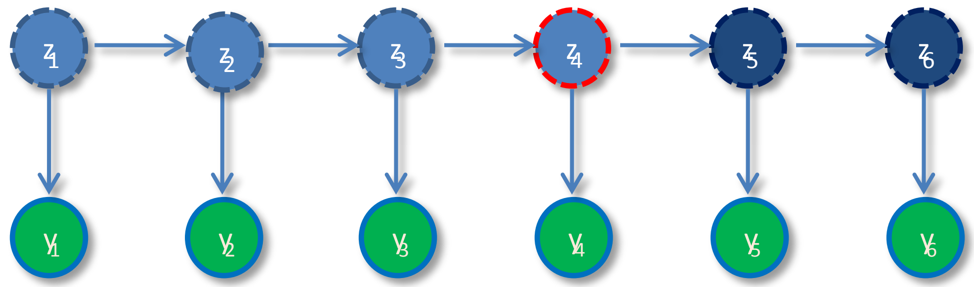
Sequential Learning (EM)



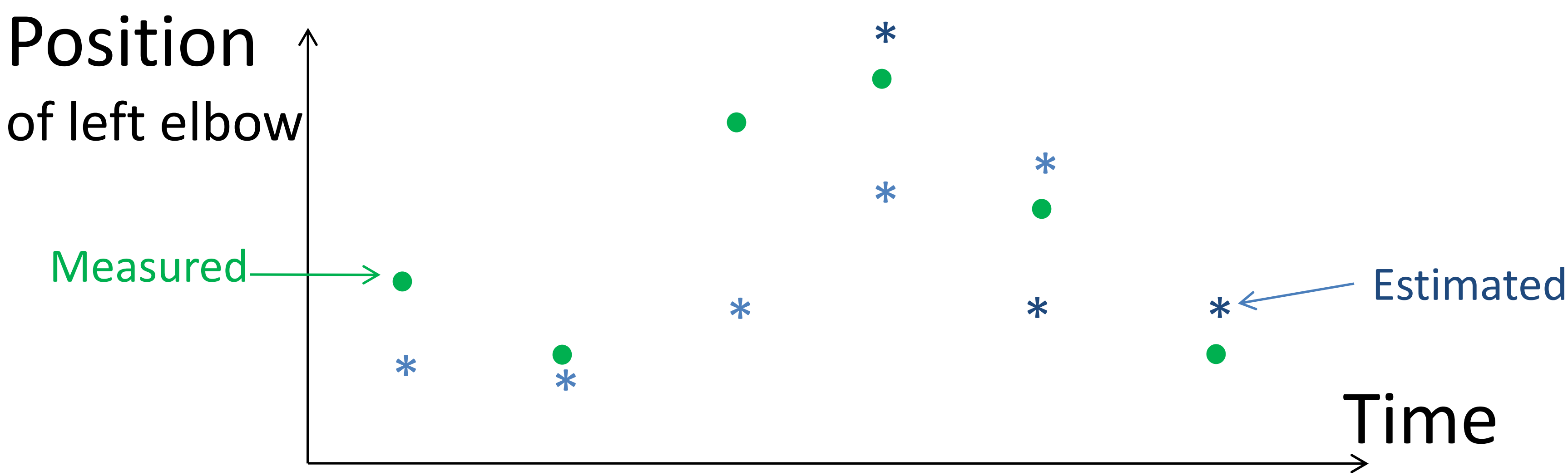
From $P(z_6 | y_1, y_2, y_3, y_4, y_5, y_6)$ → Compute $P(z_5 | y_1, y_2, y_3, y_4, y_5, y_6)$



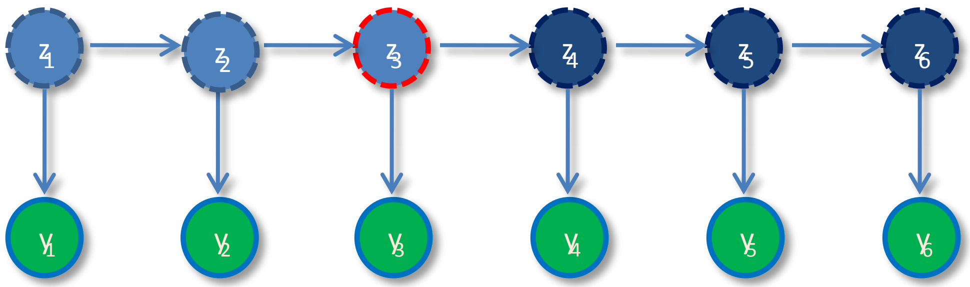
Sequential Learning (EM)



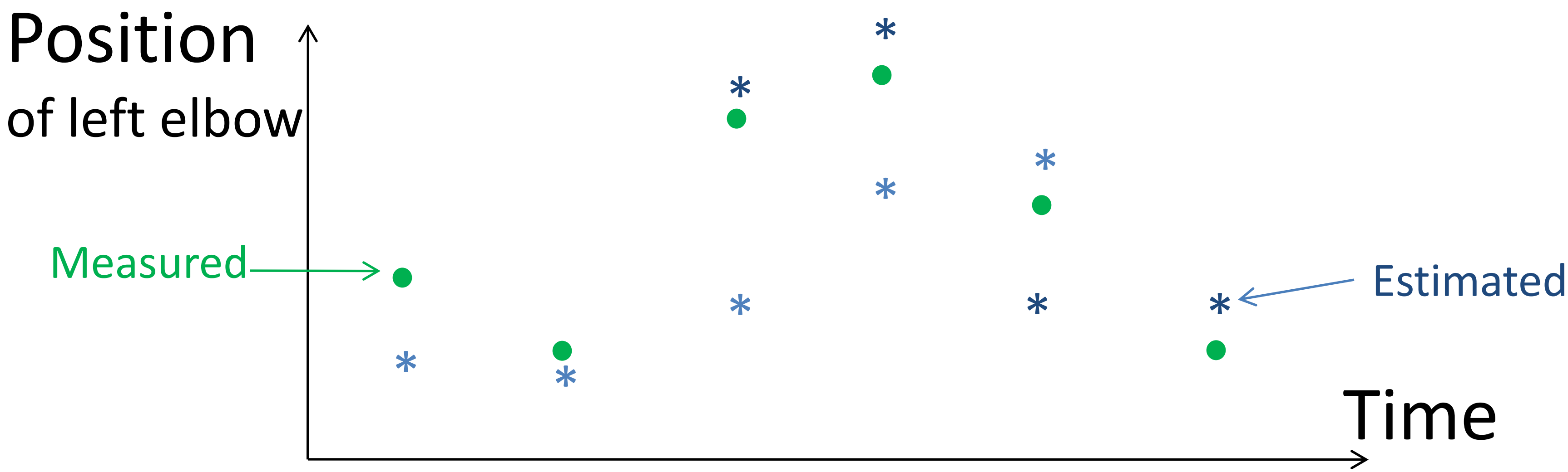
From $P(z_6 | y_1, y_2, y_3, y_4, y_5, y_6) \rightarrow$ Compute $P(z_4 | y_1, y_2, y_3, y_4, y_5, y_6)$



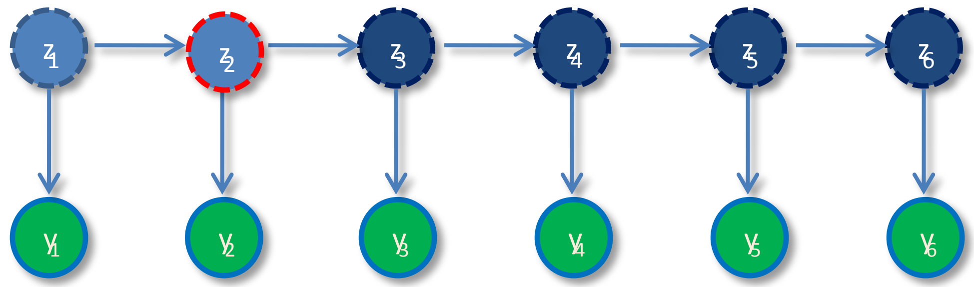
Sequential Learning (EM)



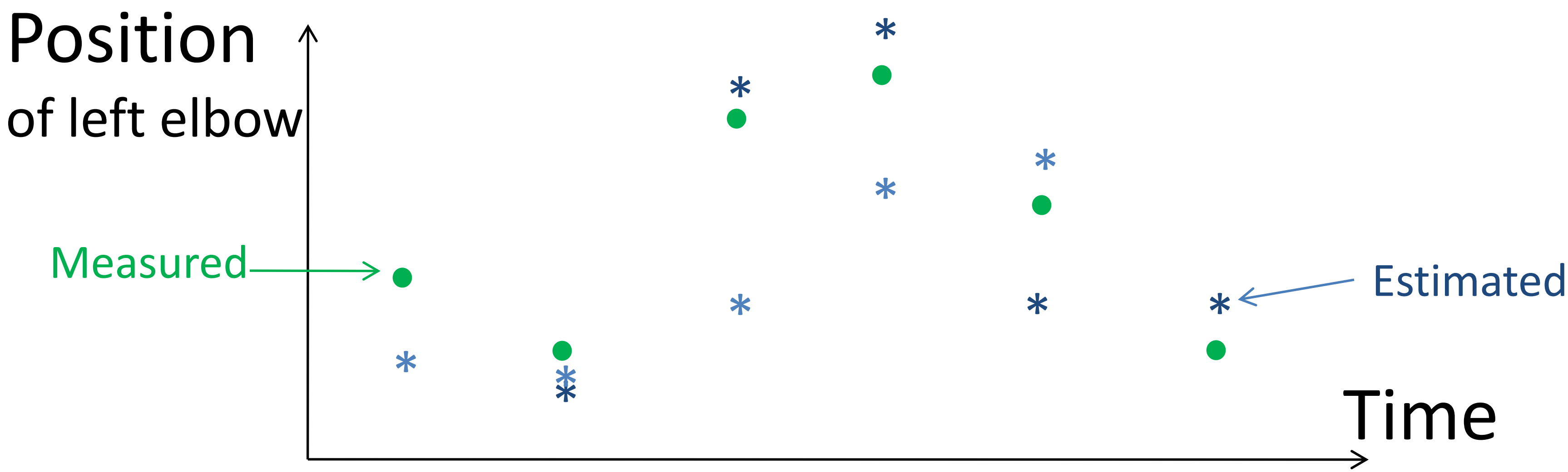
From $P(z_4 | y_1, y_2, y_3, y_4, y_5, y_6) \rightarrow$ Compute $P(z_3 | y_1, y_2, y_3, y_4, y_5, y_6)$



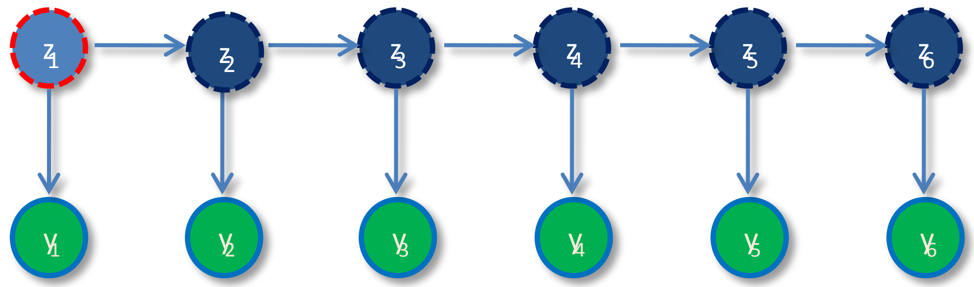
Sequential Learning (EM)



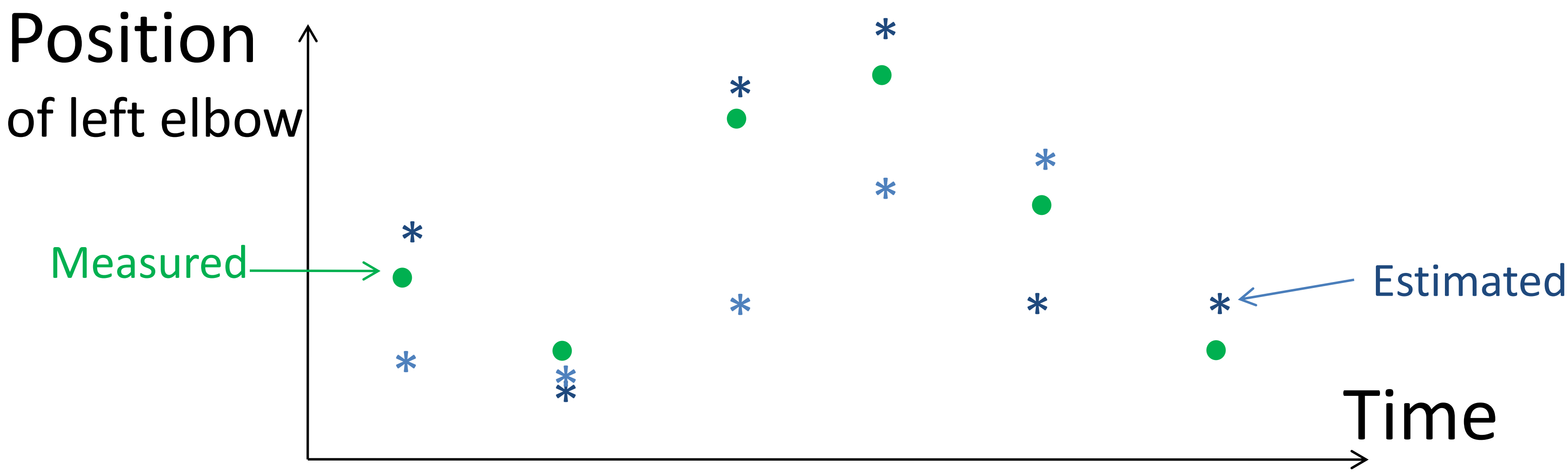
From $P(z_3 | y_1, y_2, y_3, y_4, y_5, y_6) \rightarrow$ Compute $P(z_2 | y_1, y_2, y_3, y_4, y_5, y_6)$



Sequential Learning (EM)

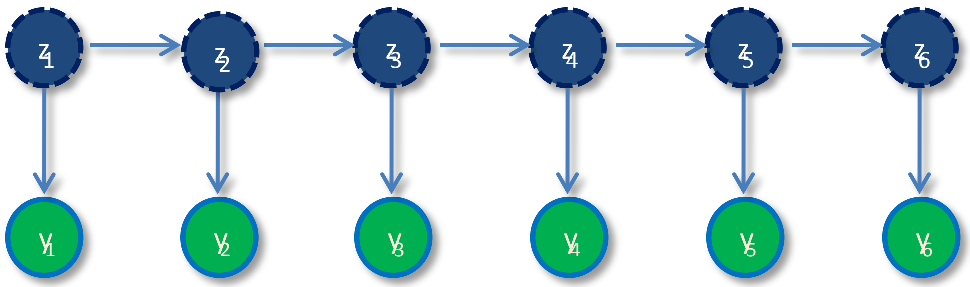


From $P(z_2 | y_1, y_2, y_3, y_4, y_5, y_6) \rightarrow$ Compute $P(z_1 | y_1, y_2, y_3, y_4, y_5, y_6)$



Carnegie Mellon

Sequential Learning (EM)



From all posterior $z_1, z_2, z_3, z_4, z_5, z_6$

$P(z_1 | y_1, y_2, y_3, y_4, y_5, y_6)$, $P(z_2 | y_1, y_2, y_3, y_4, y_5, y_6)$…

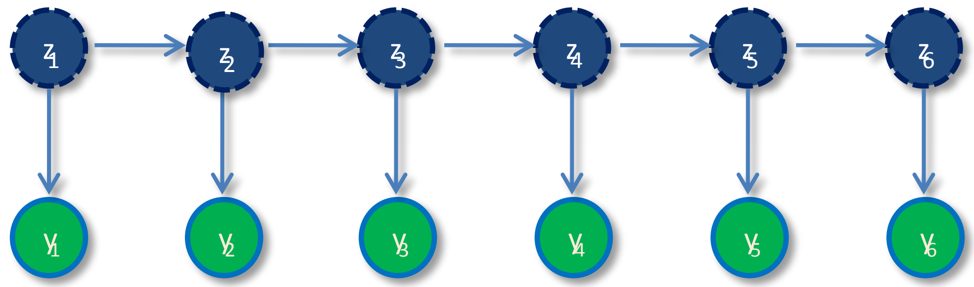
Compute sufficient statistics

$E[z_i]$

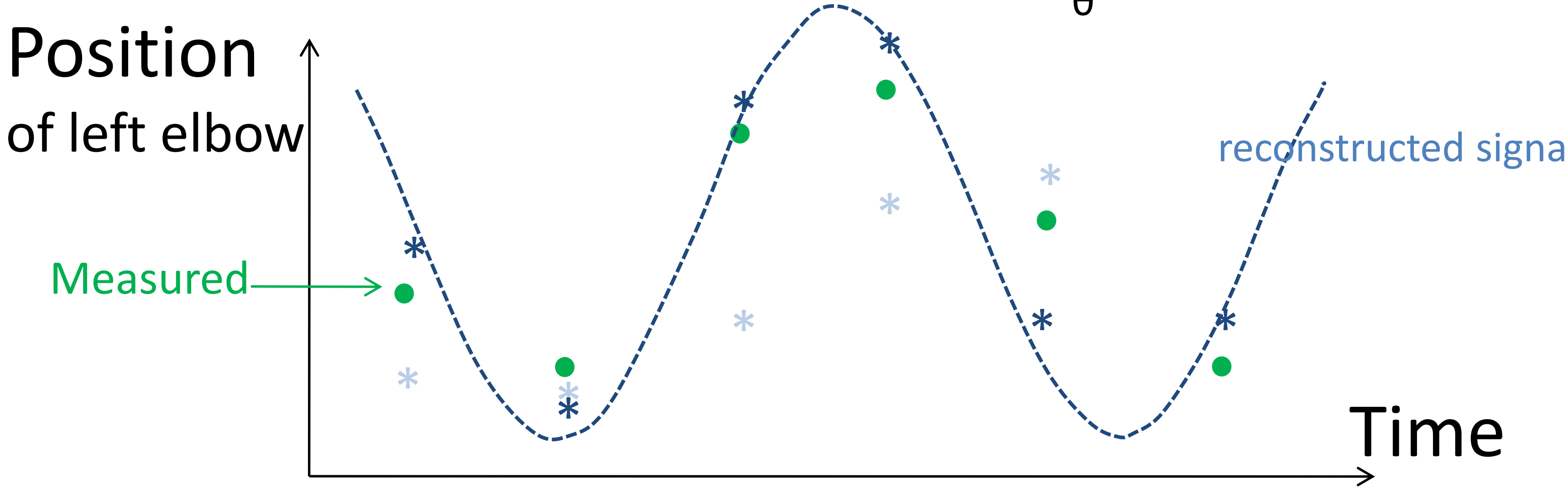
$E[z_i z_i']$

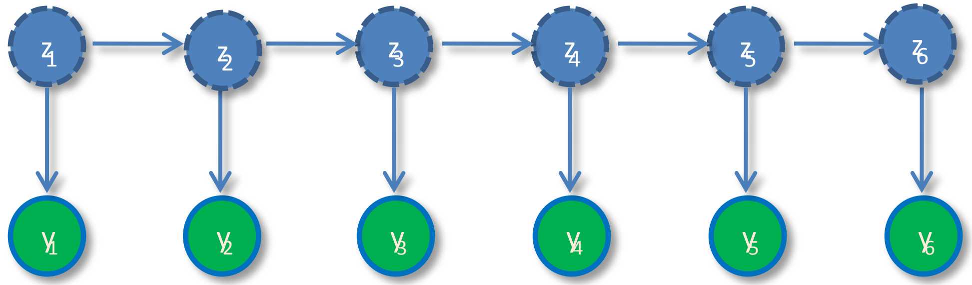
$E[z_{i-1} z_i']$

Sequential Learning (EM)



with sufficient statistics, compute $\arg\max_{\theta} \leftarrow \text{likelihood}(\theta)$





Speed Bottleneck:

sequential computation of posterior

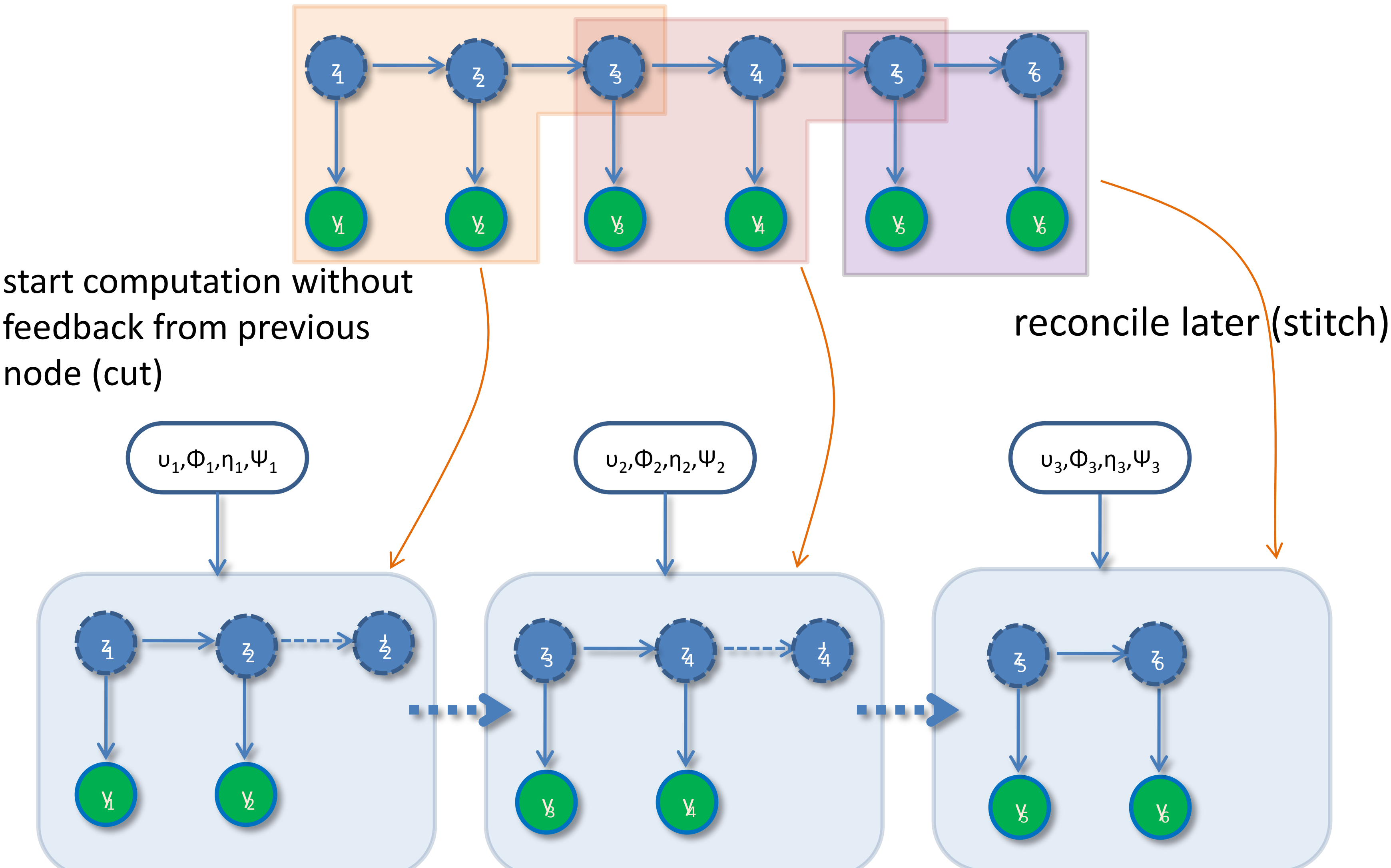
How to *parallelize* it?

“Leap of faith”

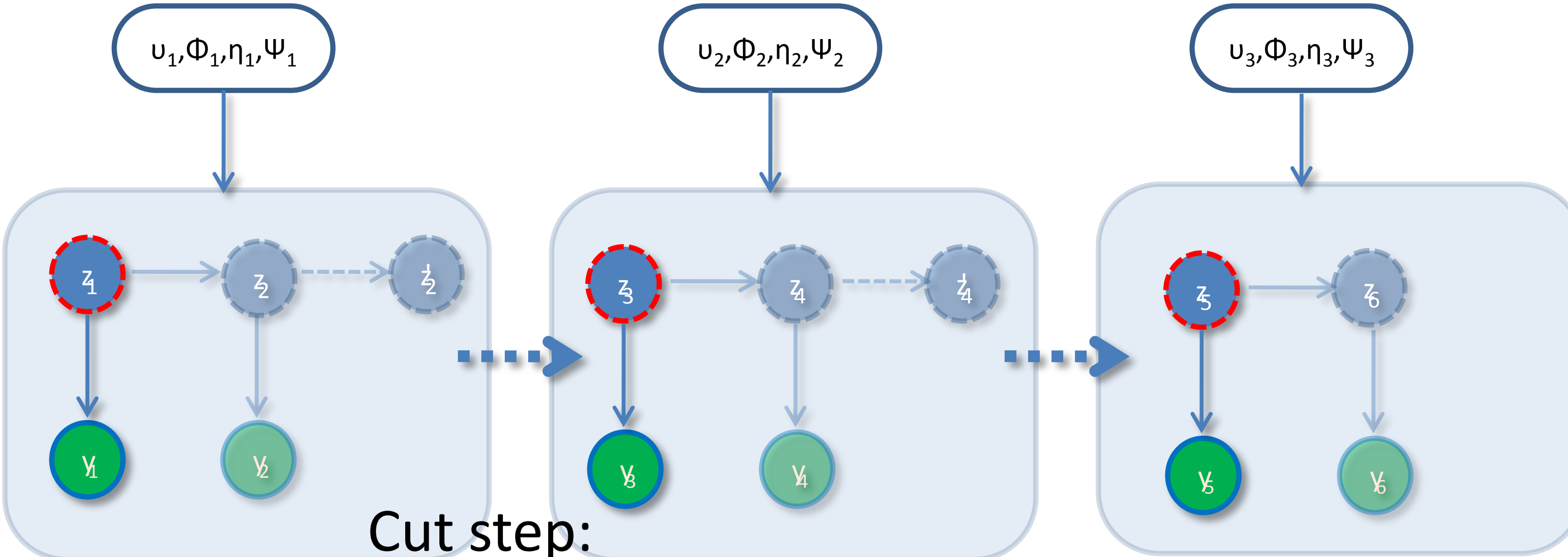
start computation without feedback from previous node (cut),

and reconcile later (stitch)

Proposed Method: Cut-And-Stitch

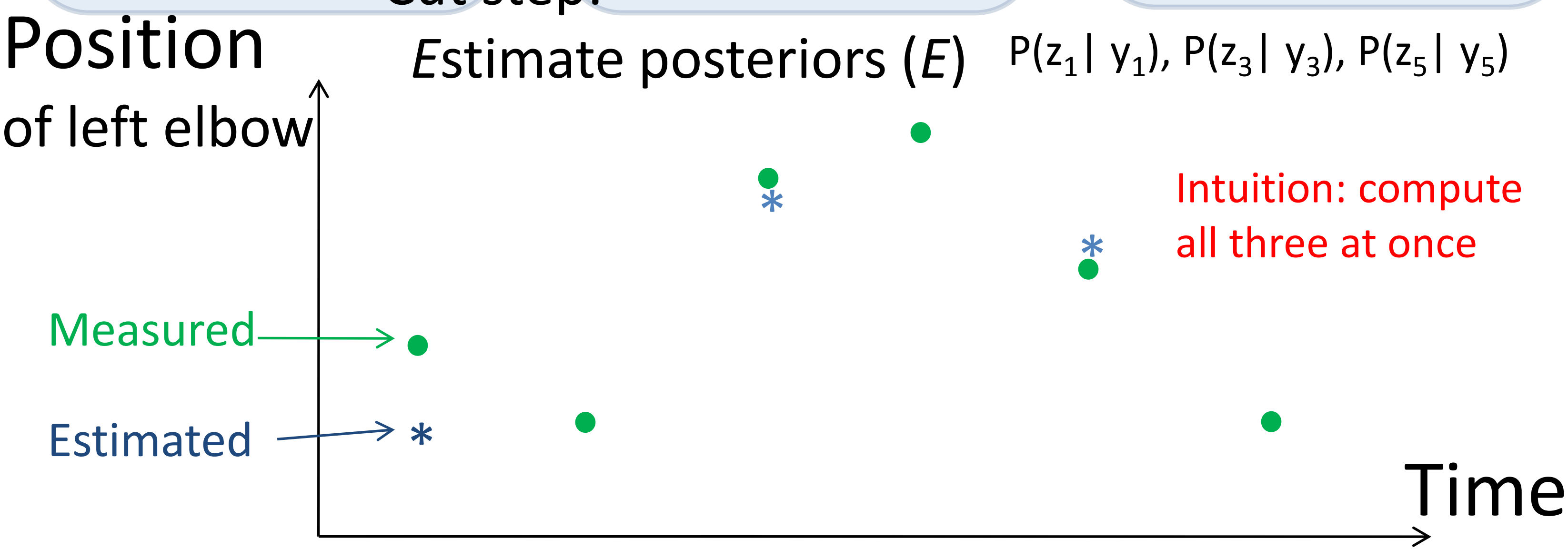


Cut-And-Stitch



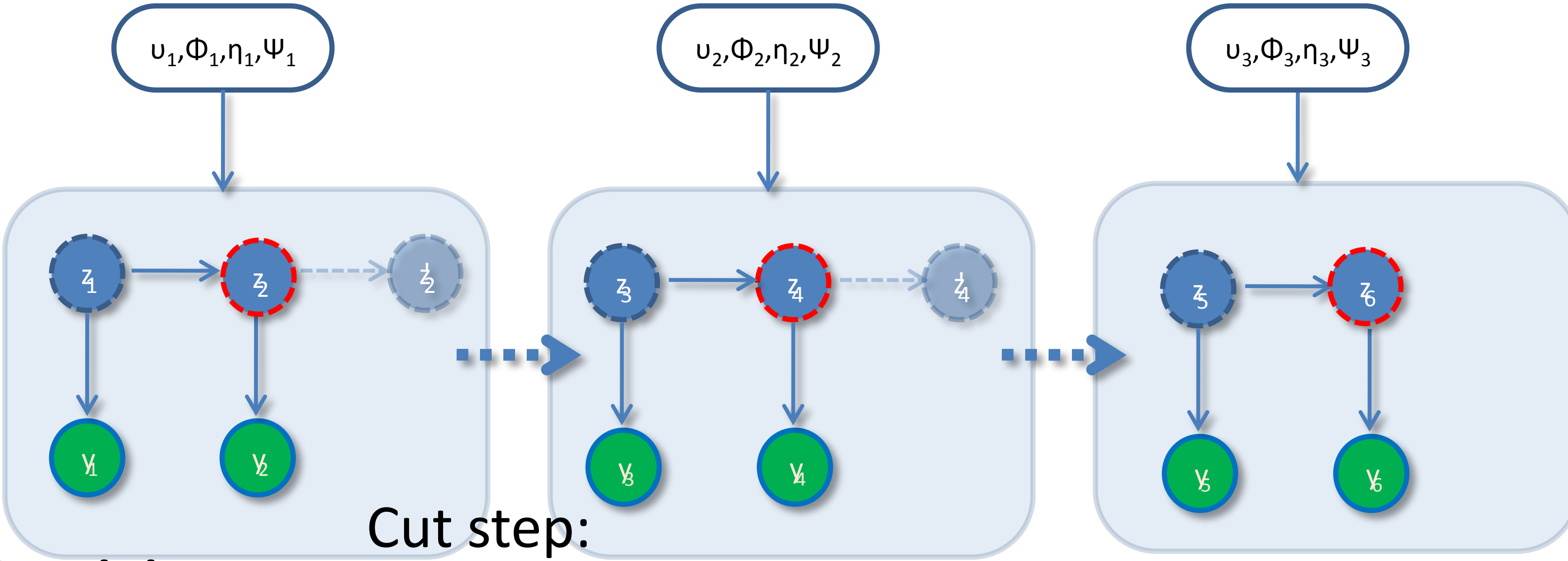
Cut step:

*E*stimate posteriors (*E*) $P(z_1 | y_1), P(z_3 | y_3), P(z_5 | y_5)$



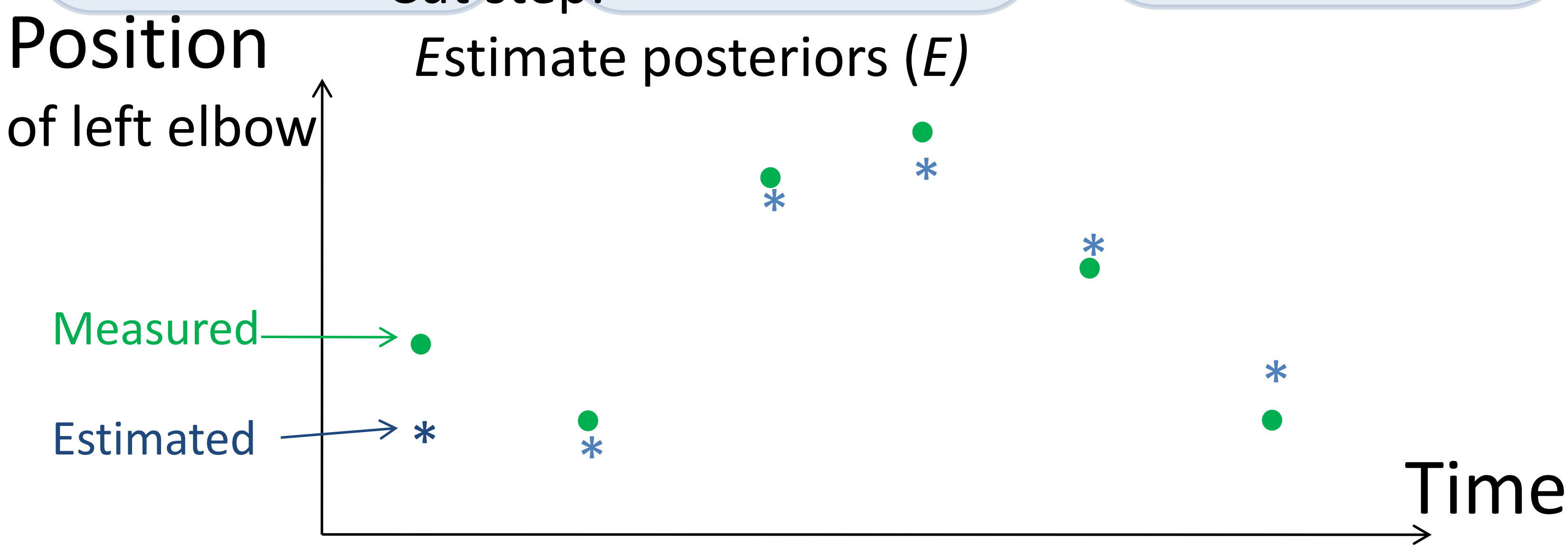
Intuition: compute all three at once

Cut-And-Stitch

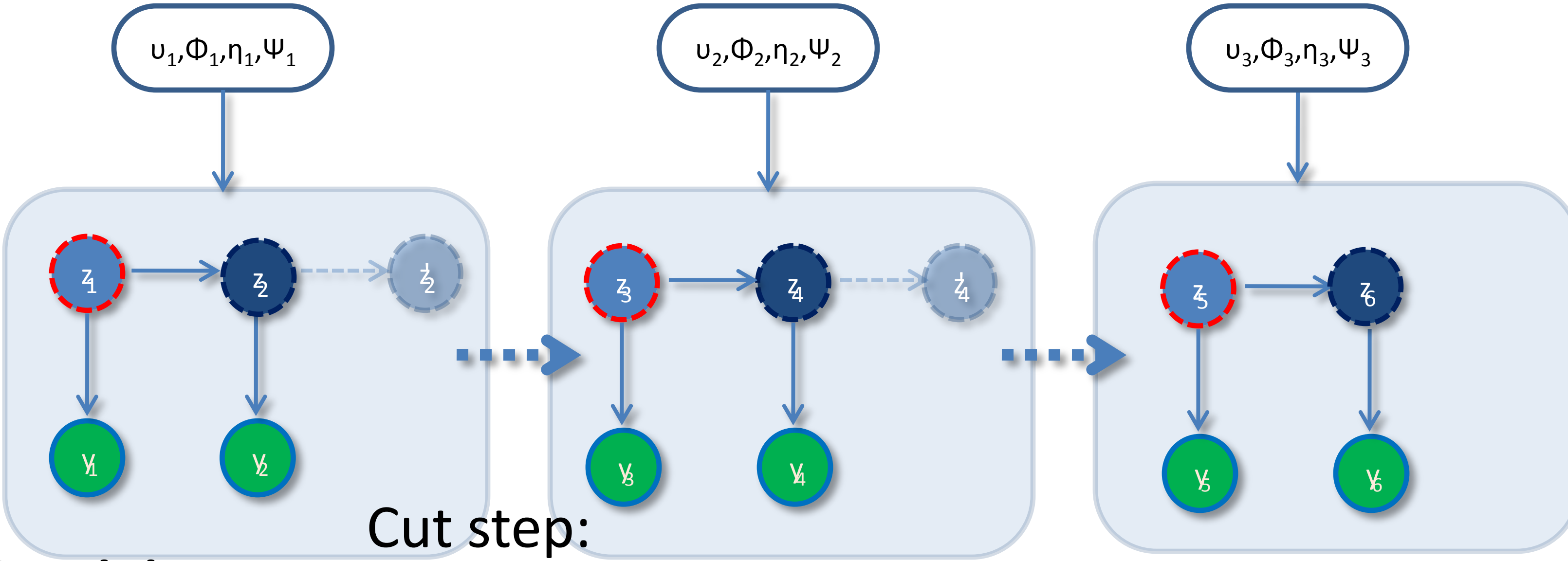


Cut step:

*E*stimate posteriors (*E*)

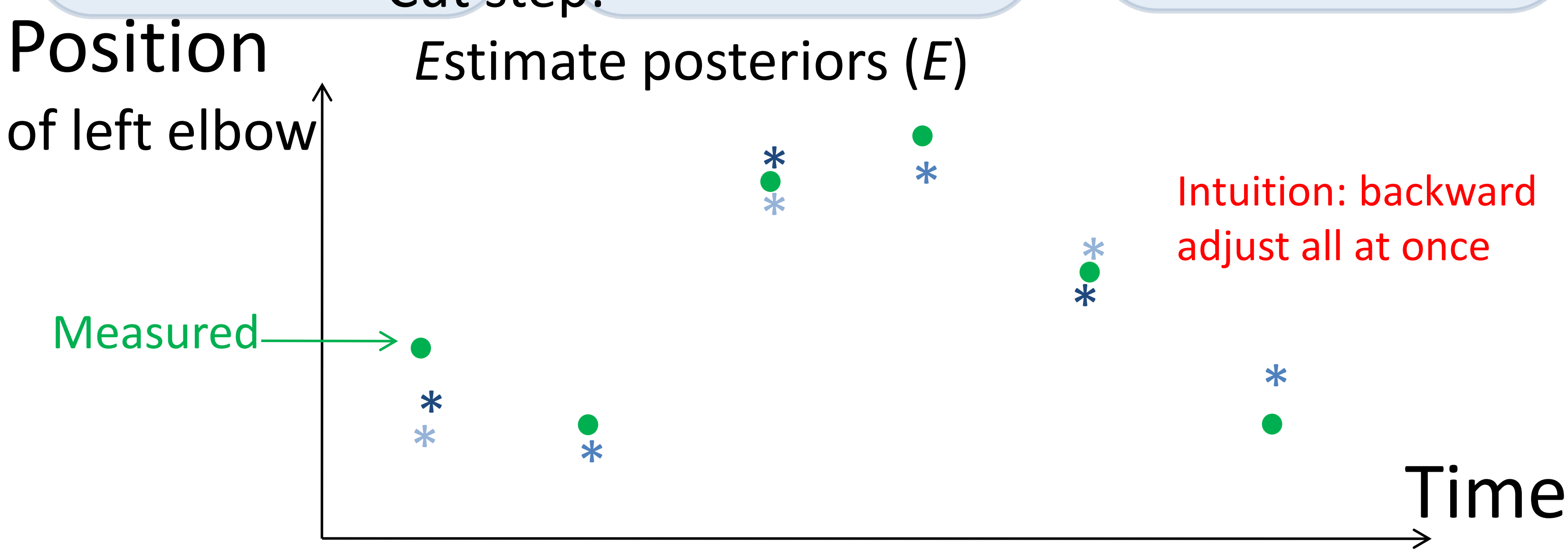


Cut-And-Stitch



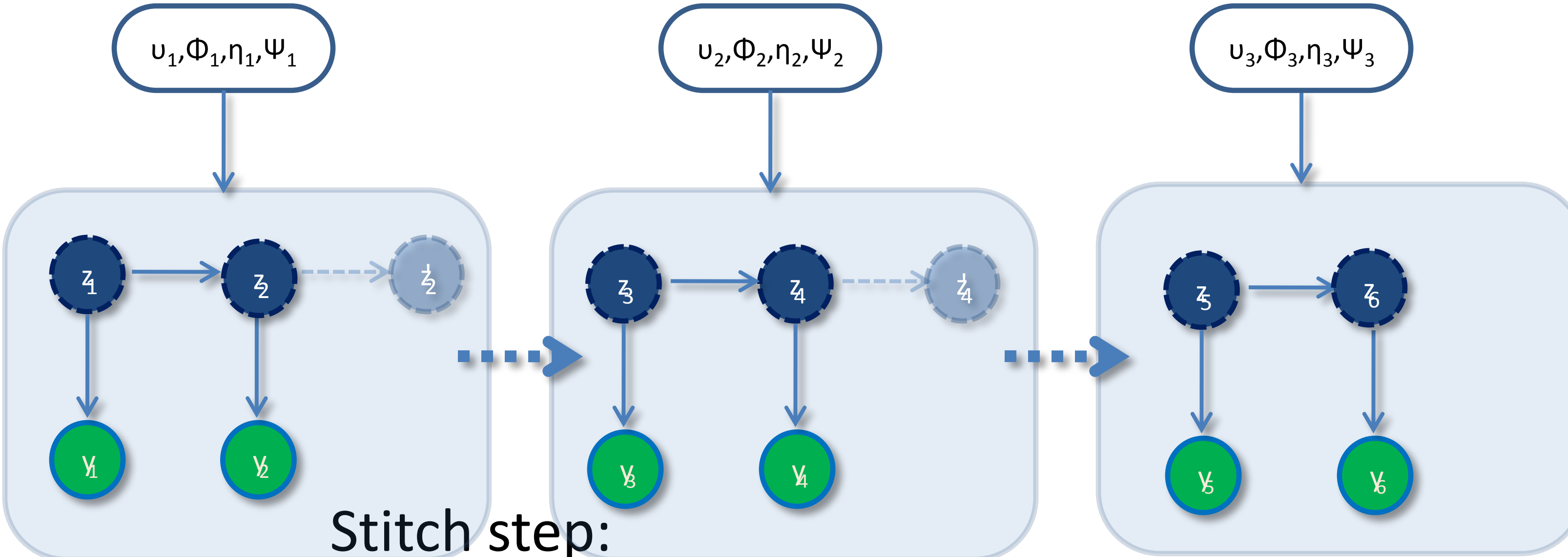
Cut step:

*E*stimate posteriors (*E*)



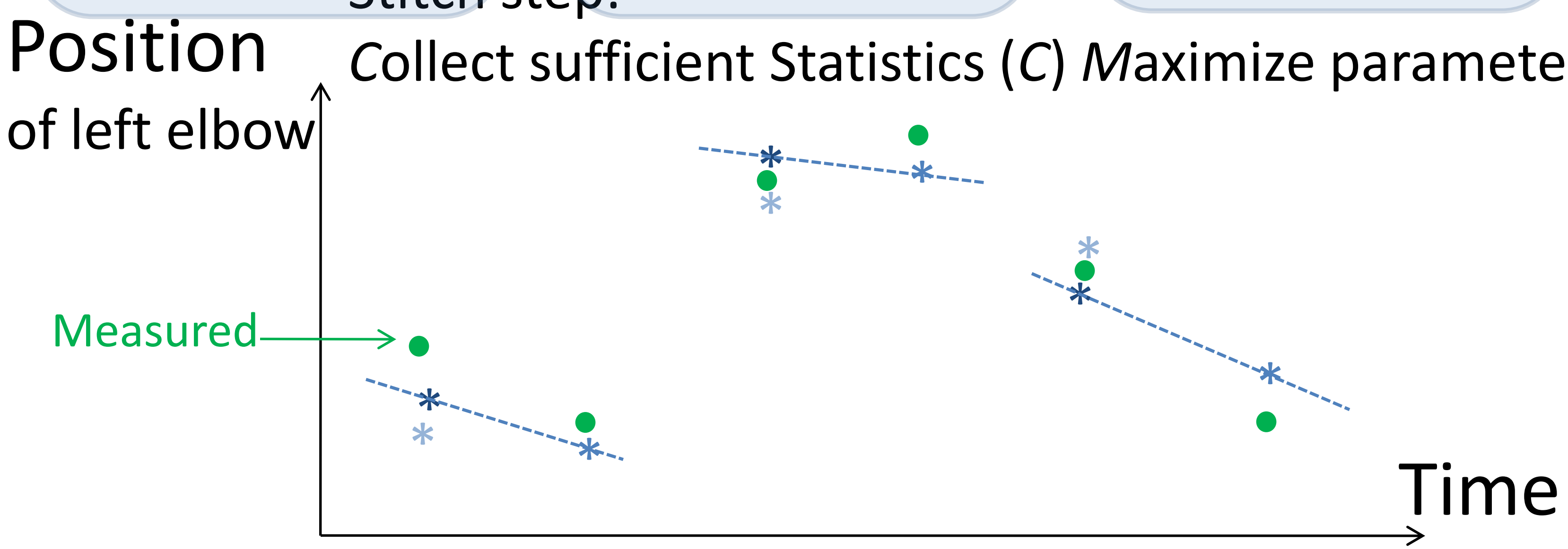
Intuition: backward adjust all at once

Cut-And-Stitch

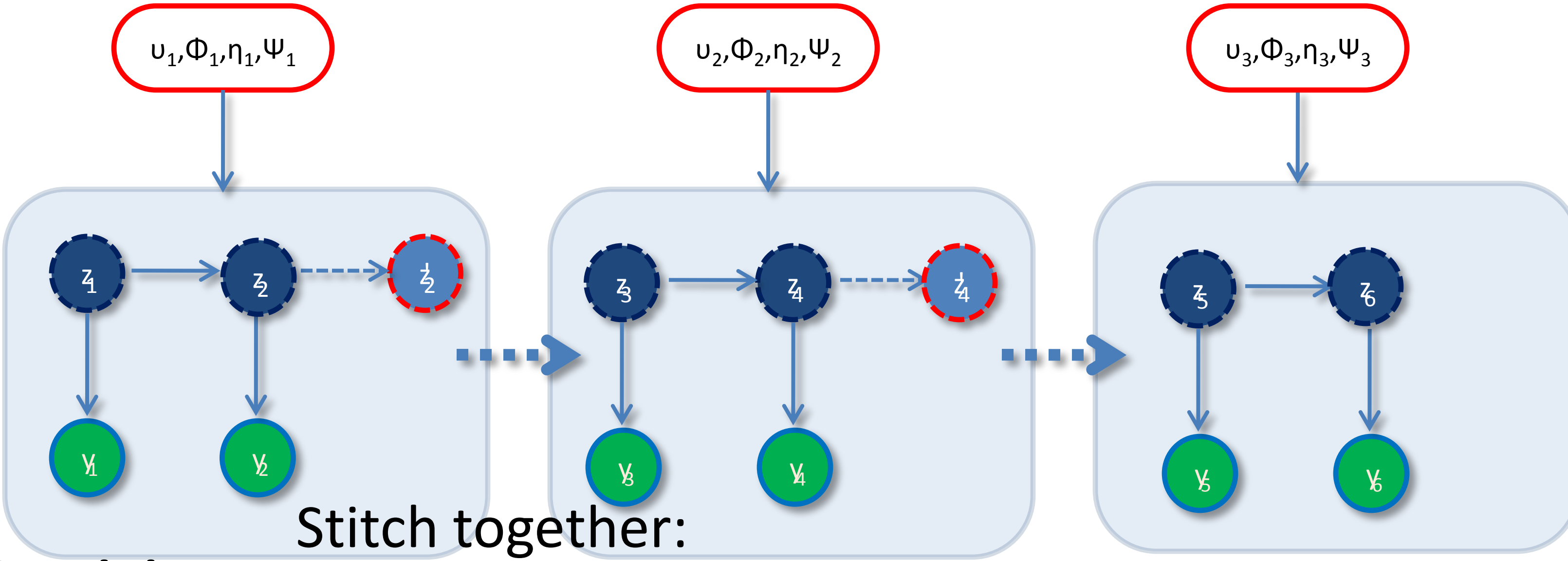


Stitch step:

*C*ollect sufficient Statistics (*C*) *M*aximize parameters (*M*)

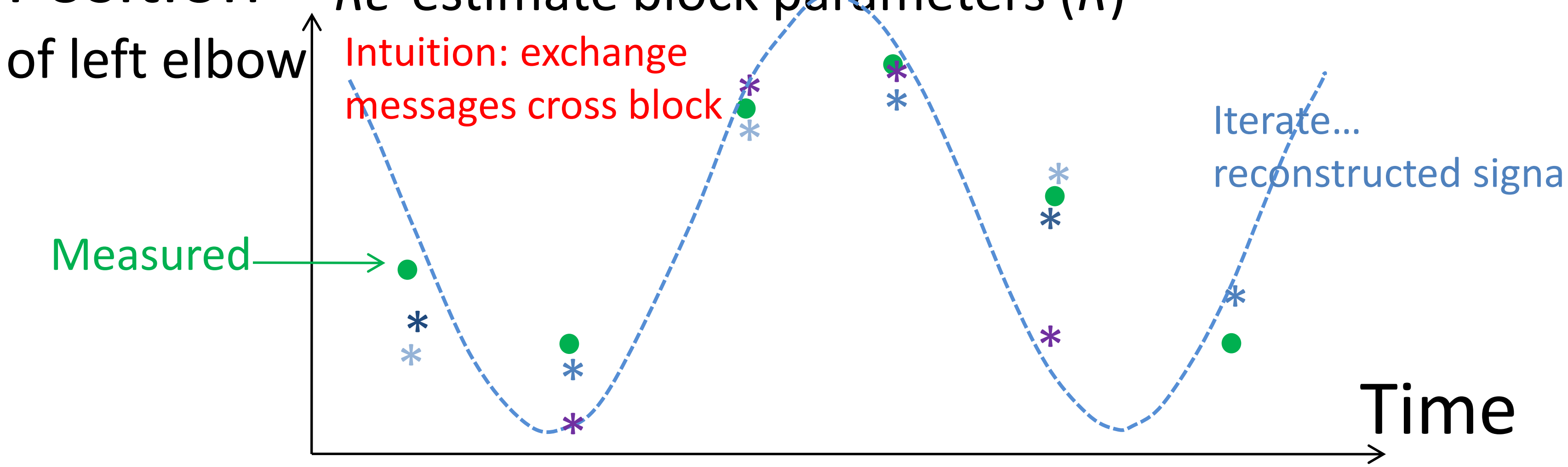


Cut-And-Stitch



Stitch together:

Re-estimate block parameters (*R*)



Outline

- Background
- Motivation: effortless motion stitching
- Parallel learning with Cut-And-Stitch
- Experiments and Results
- Conclusion

Experiments

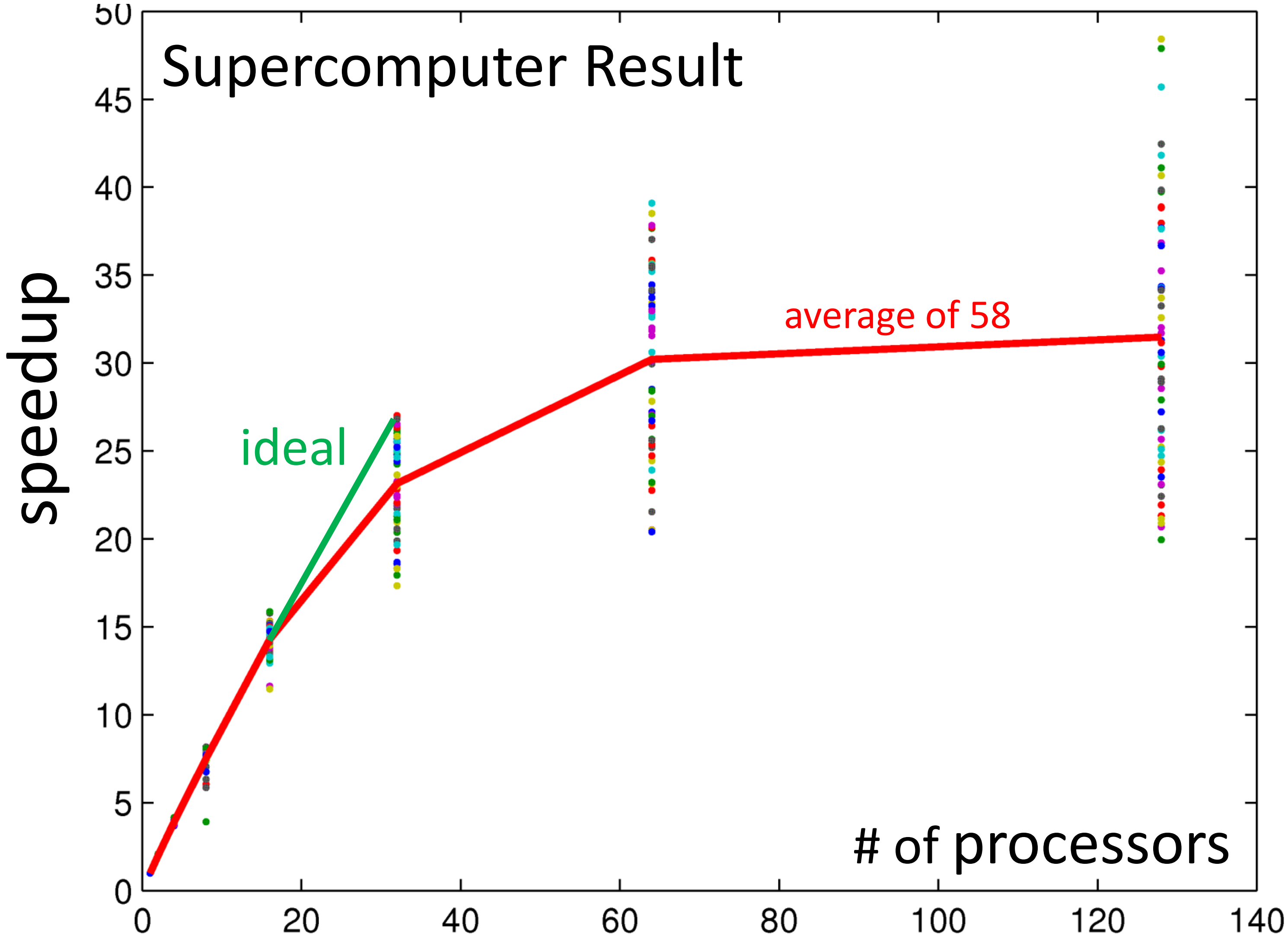
Q1: How much speed up can we get?

Q2: How good is the reconstruction accuracy?

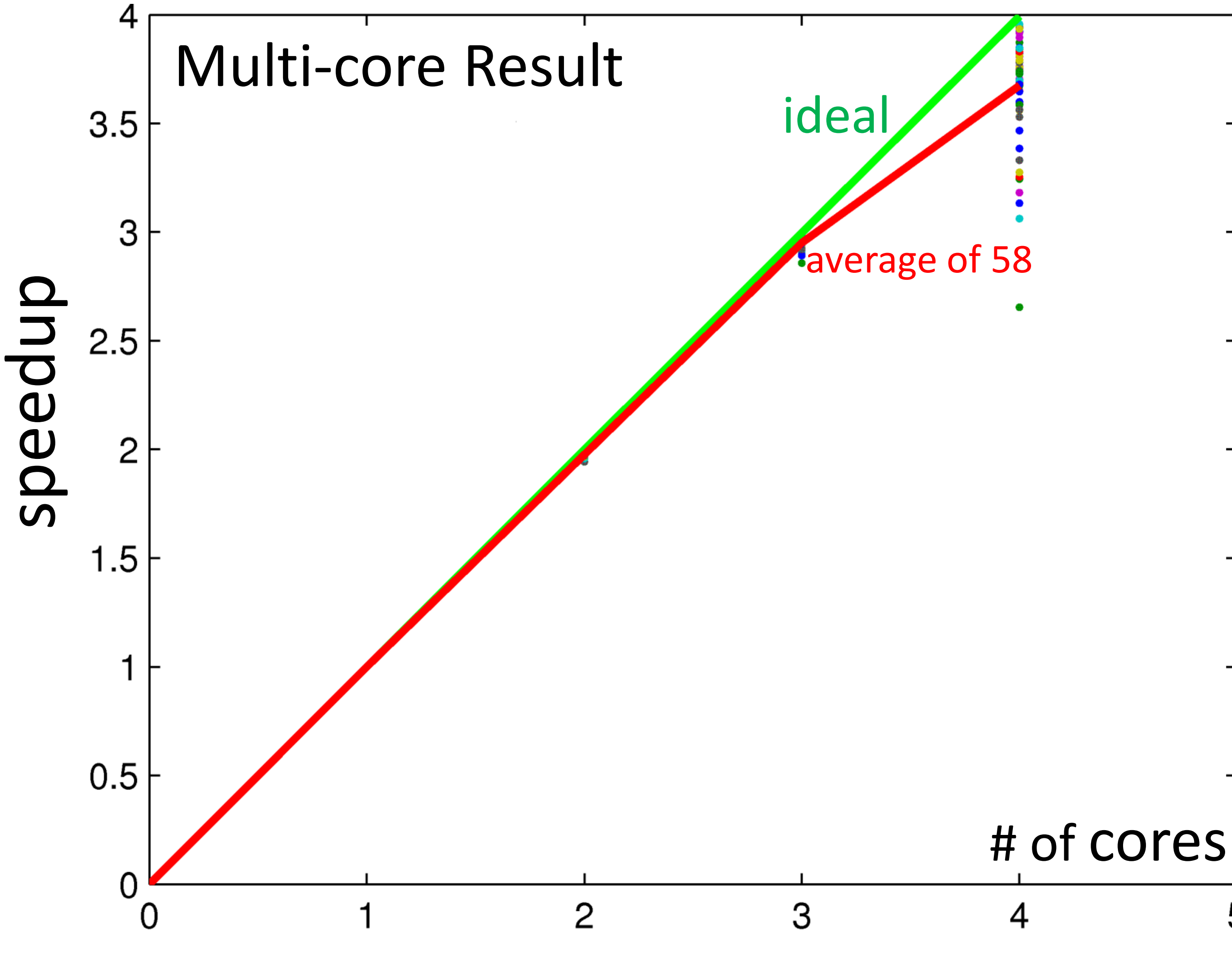
Experiments

- Dataset:
  - 58 human motion sequences, 200 – 500 frames
  - Each frame with 93 bone positions in body local coordinates
  - http://mocap.cs.cmu.edu
- Setup:
  - Supercomputer: SGI Altix system, distributed shared memory architecture
  - Multi-core desktop: 4 Intel Xeon cores, shared memory
- Task:
  - Learn the dynamics, hidden variables and reconstruct motion

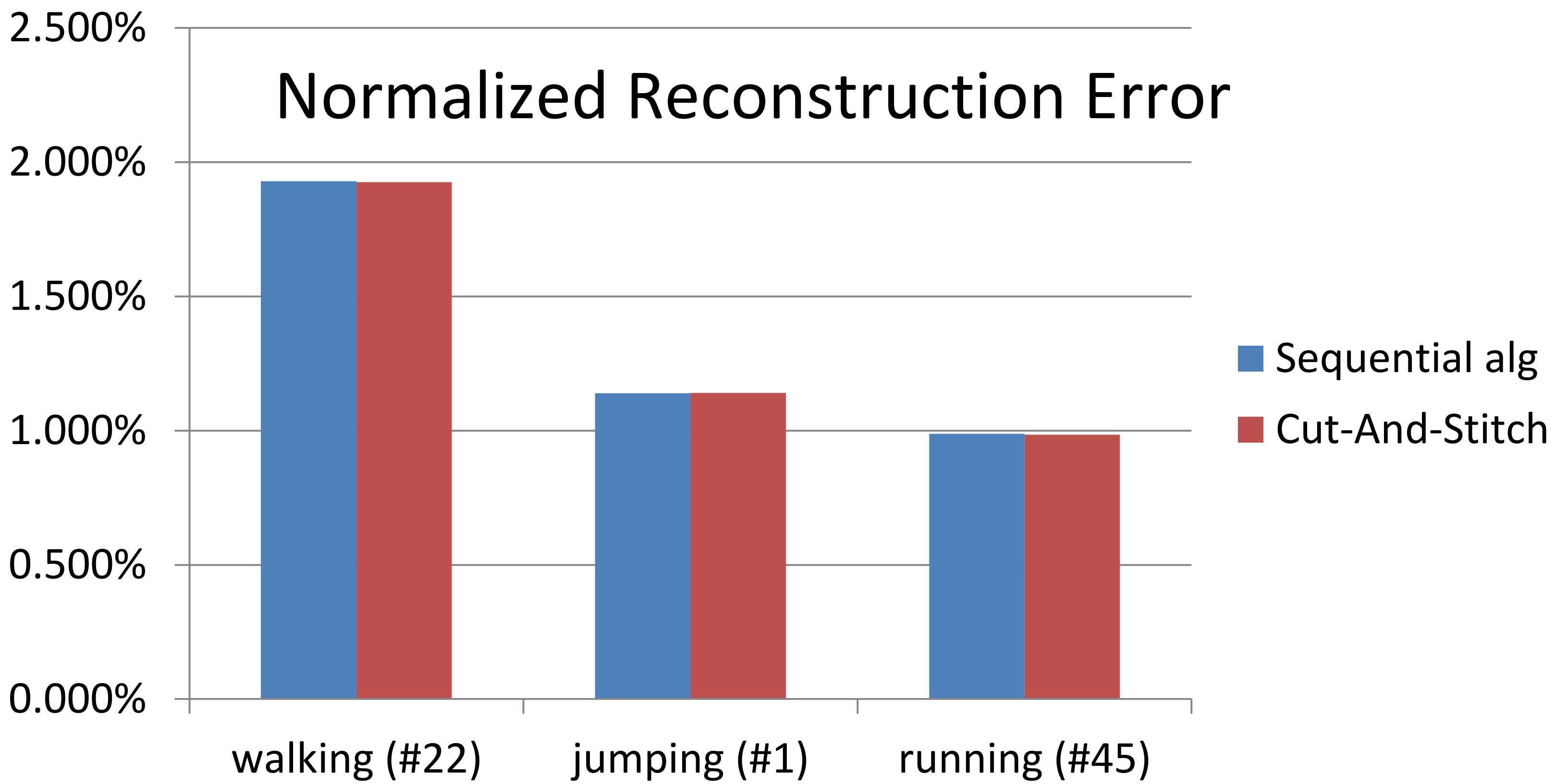
Q1: How much speed up?



Q1: How much speed up?



Q2: How good?



Result: ~ IDENTICAL accuracy

Conclusion & Contributions

- A distance function for motion stitching
  - Based on first principle: minimize effort
- General approximate parallel learning algorithm for LDS
  - Near linear speed up
  - Accuracy (NRE): ~ identical to sequential learning
  - Easily extended to HMM and other chain Markovian models
- Software (C++ w. openMP) and datasets: www.cs.cmu.edu/~leili/paralearn

Promising Extensions

- Extension
  - HMM
  - other Markov models (*similar graphical model*)
- Open Problem:
  - Can prove the error bound?

Thank you

- Questions