What has Matto been up to?

Matthew Marjanović

MIT AI Lab

May 17, 2002

New sok Features

- Persistent Data: allocate storage which is preserved across death/respawn of a sok process.
- Dump and Restore: can dump the state of sok processes (persistent data and connections) to files, and restore them later.

New Sensory Input

- Tactile Sense: 22 FSR pressure sensors on hand, wired up to produce 6 analog tactile signals.
- Joint Pain: process monitors joint angle and produces a linear “pain” signal when joints are within 10% of their limits.

New Motor Output

- Old geometric musculoskeletal model is Out:

$$l_{AB} = \|\vec{p}_A - \vec{p}_B\|$$

$$\vec{\tau}_j = \vec{F} \times \vec{r} = \frac{F}{l_{AB}}(\vec{p}_B - \vec{p}_A) \times (\vec{p}_B - \vec{q}_j)$$

  - Point-to-point action; no notion of a tendon which could wrap around joints.
  - Anchor points must be placed away from skeleton.
- New simple musculoskeletal model is In:

$$l_m = \Sigma_j R_{mj} \theta_j$$

$$\tau_j = \Sigma_m R_{mj} F_m$$

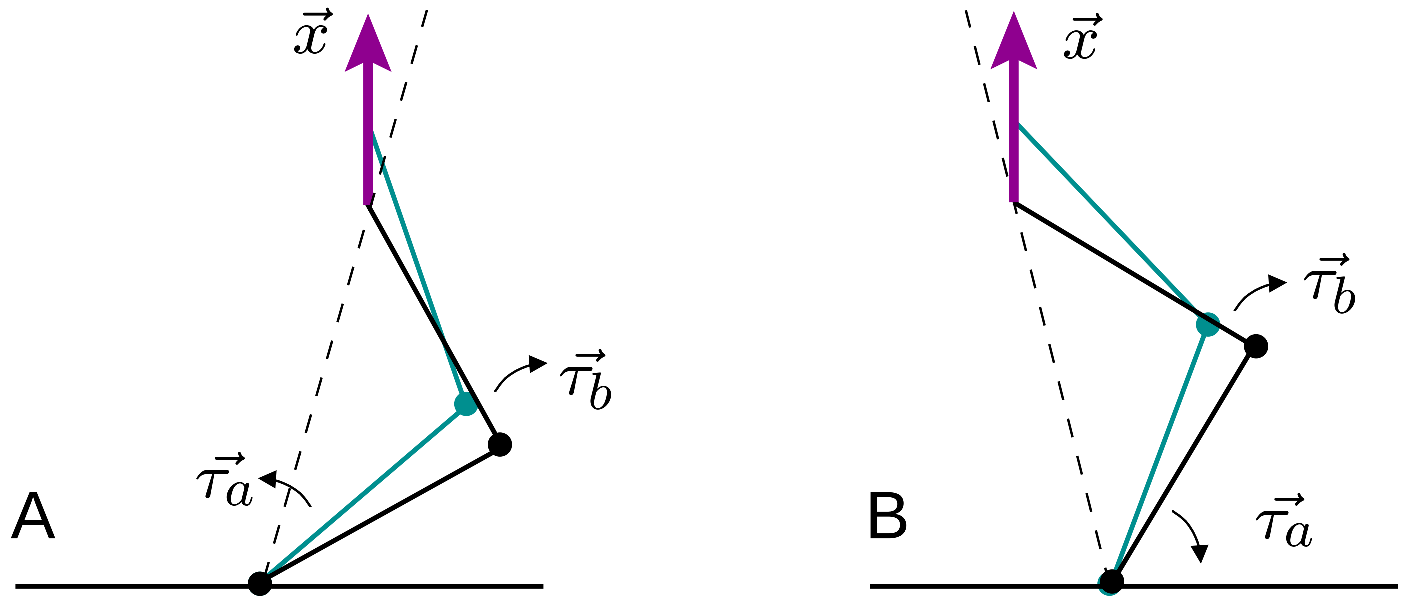
  - Still allows for polyarticular coupling.
  - No compilation necessary; coupling can be changed at runtime.

Polyarticular Muscles

Many muscles in the human body span more than one joint, e.g. biceps and triceps.

- *Kinematically* redundant.
- Positive effects on limb *dynamics:*
  - Single-joint linear springs alone cannot produce isotropic stiffness in end-point coordinates.
  - Polyarticulate muscles can make the musculoskeletal system more efficient (biomechanically).

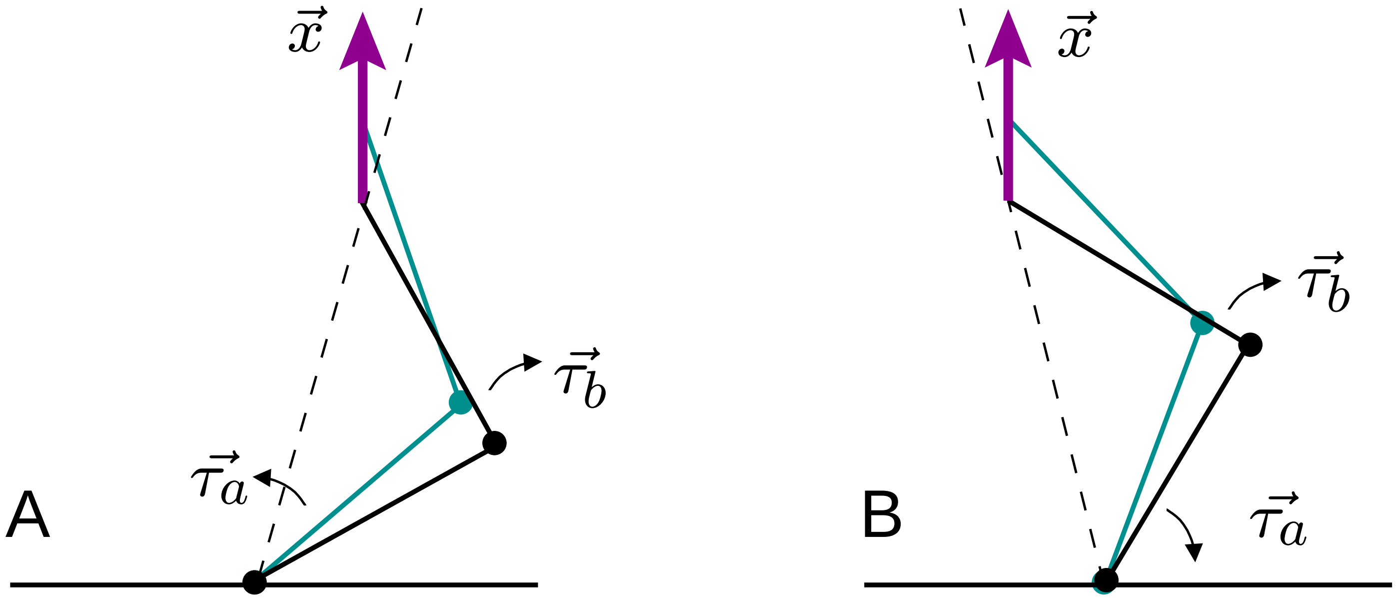
Mechanical Efficiency



To do work along $\vec{x}$ — that is, to apply a force along that vector — the arm must move into the blue configuration.

(A) The joints apply torques $\vec{\tau}_a$ and $\vec{\tau}_b$ in the same directions as they are displaced, thus both contributing to the output work.

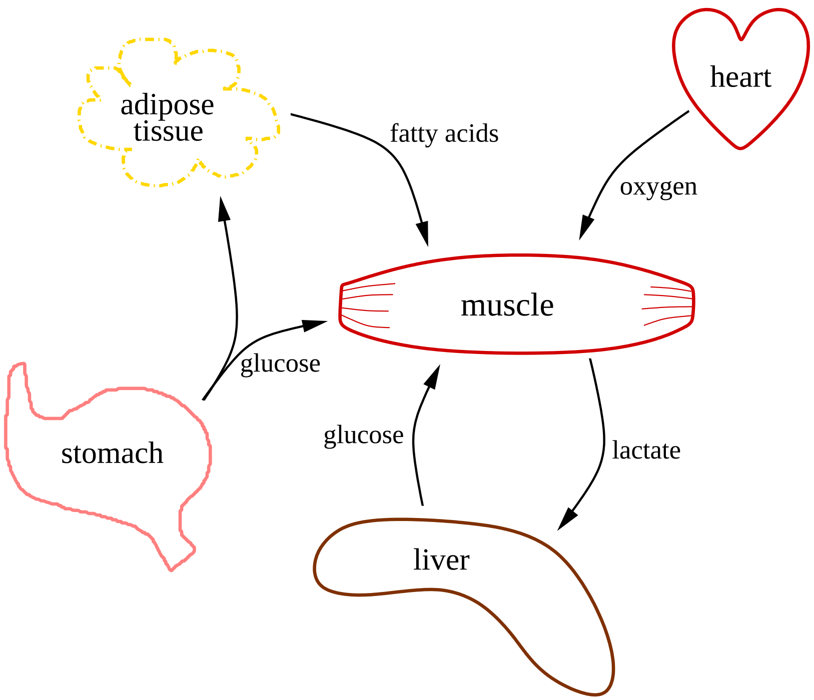
Mechanical Efficiency



(B) Joint a applies torque $\vec{\tau}_a$ in *opposition* to its displacement.

Joint a is *absorbing* energy, which must have been provided by joint b.

Fatigue Model

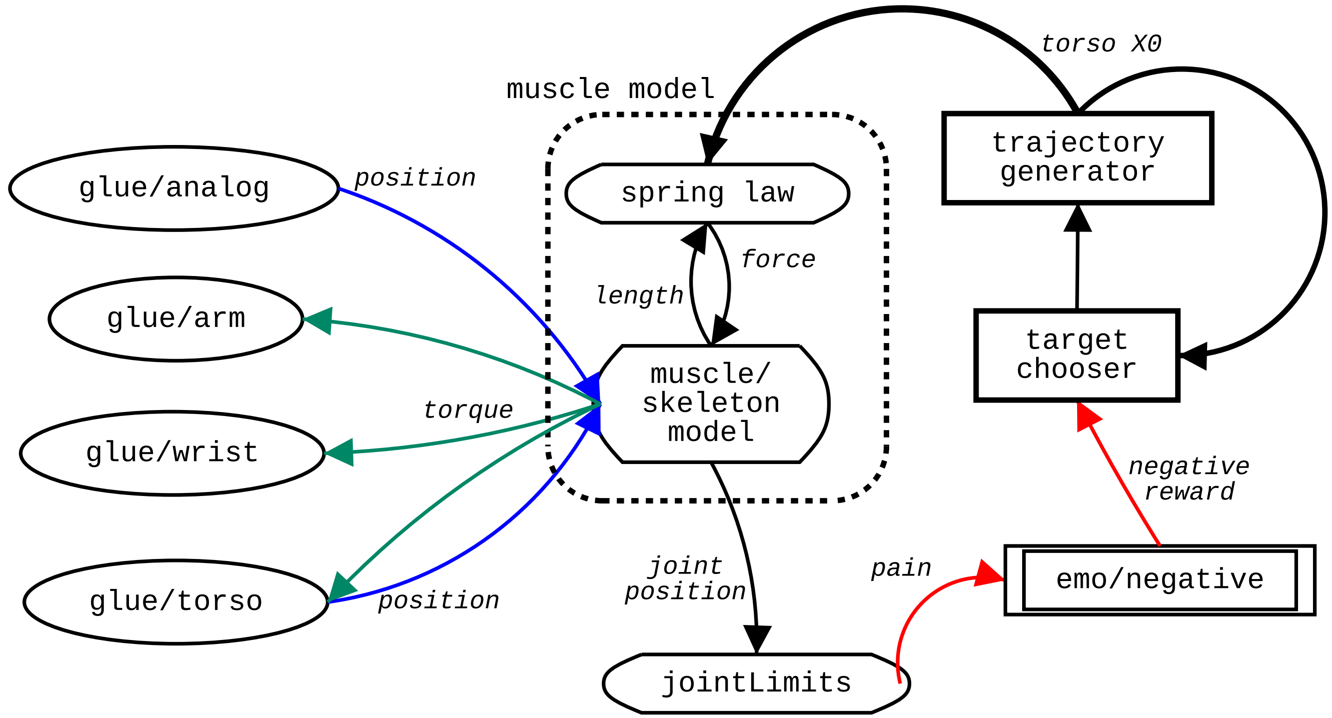


A complete metabolic model simulates major organs involved with energy production and consumption. Long-term exertion causes build-up of lactic acid in muscle tissue, leading to fatigue. Heart rate and fuel availability affect the rate of lactic acid accumulation.

Fatigue Model

- Lactic acid level in each muscle used to modulate force output.
- Level is exposed to the rest of the system, to be used as an indicator of fatigue or soreness.
- Hook for modifying the blood glucose level (simulating ingestion of food).
- Hook for injecting epinephrine into the system, which could be linked to an emotional system to provide a physically realized reaction to stress.

Task 0: Sitting up straight



- Chooser picks random $\vec{x}_0$ target, according to distribution.
- Smooth trajectory generator sends $\vec{x}_0$ stream to springs.
- Feedback from joint pain modifies the distribution.

Learning and Choosing from a Distribution

- Record samples $(\vec{x}_i, p_i)$, where $\vec{x}_i$ is a commanded position and p_i is a value $[0, 1]$ calculated from the reward/pain signal.
- Estimate $p(\vec{x})$ by summing over samples:

$$p(\vec{x}) = \frac{\Sigma p_i g(\vec{x} - \vec{x}_i)}{\Sigma g(\vec{x} - \vec{x}_i)}, \; where \; g = C \exp\left(\frac{-|\vec{x} - \vec{x}_i|^2}{\sigma^2}\right)$$

- To choose from the distribution:
  - Choose $\vec{x}$ from a uniform distribution over $[\vec{x}_{min}, \vec{x}_{max}]$.
  - Calculate $p = p(\vec{x})$.
  - Choose q uniformly from $[0, 1]$ (or $[0, p_{max}]$).
  - If $q \leq p$, return $\vec{x}$. Else, start over.

Task 1: Dealing with Gravity — Bias Force

- Gravity exerts a constant force on the body.
- To maintain a posture, motors must exert a counter-torque.

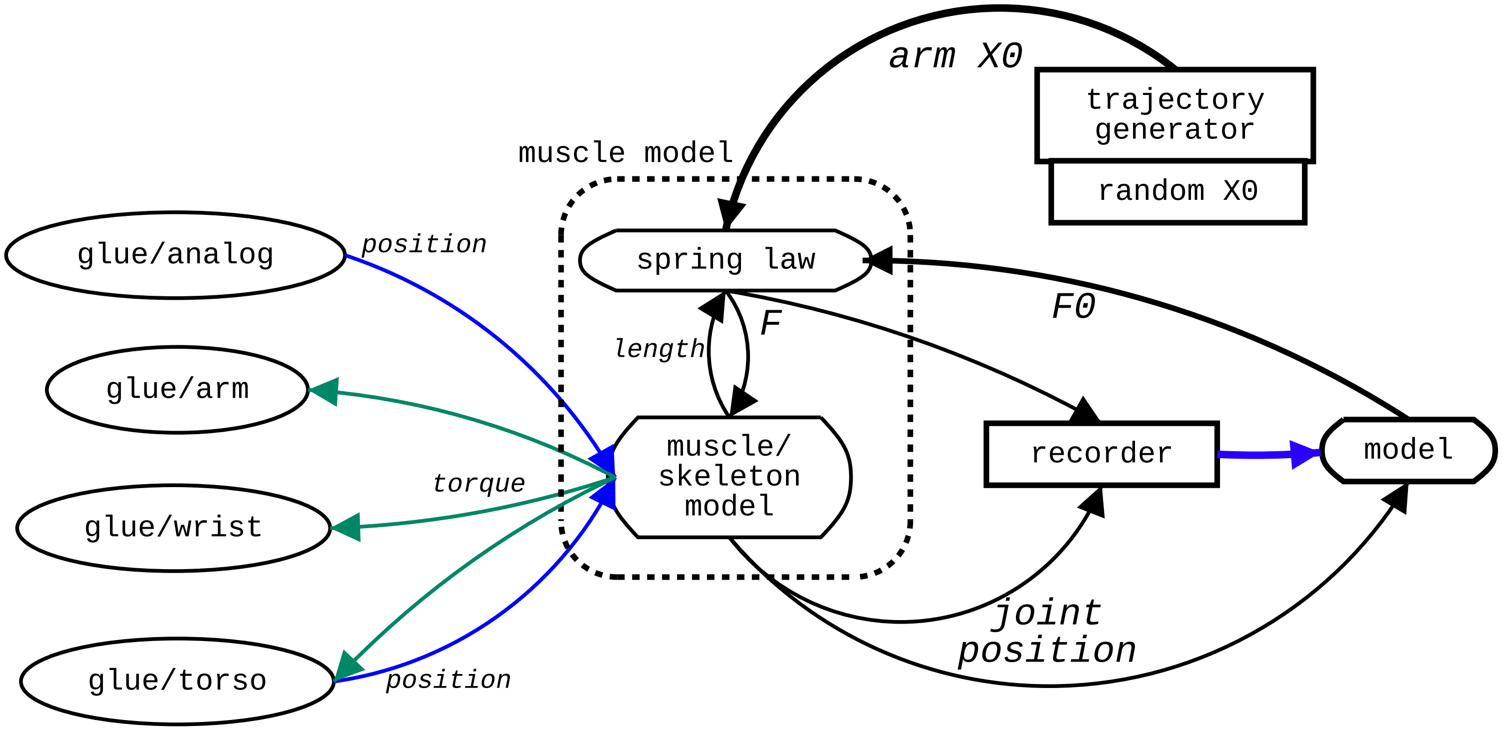
Basic Spring Law: $F = K(x - x_0)$

- For any raised posture, F must be non-zero, so position error is non-zero, so K must be large to keep error small.

Introduce a bias force: $F = K(x - x_0) + F_0$

- Let F_0 be the force needed to counteract gravity, $\vec{F}_0(\vec{\theta})$.
- If $\vec{F}_0(\vec{\theta})$ is accurate, limbs become "weightless".
- Stiffness can be lower; effective workspace becomes larger.

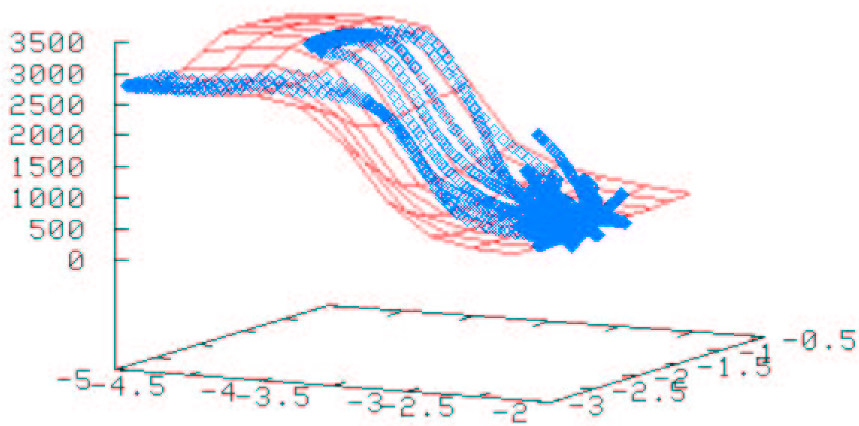
Learning the Bias Force

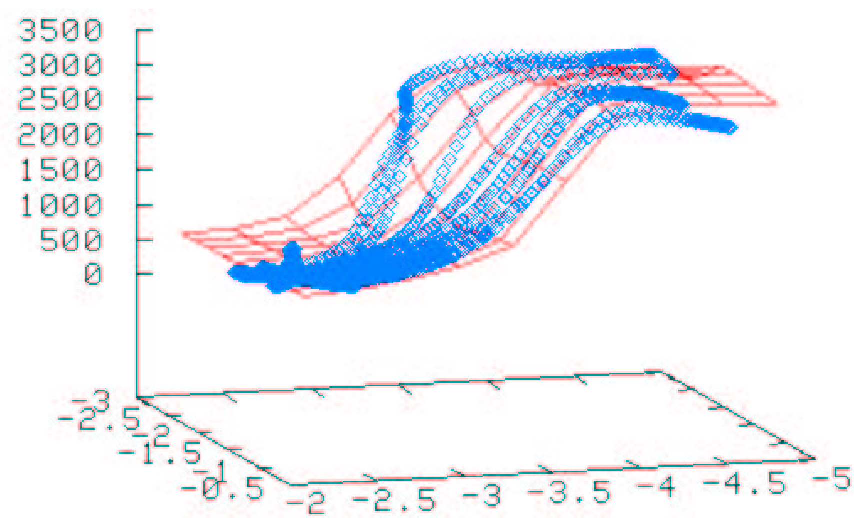


- Move arm randomly by commanding equilibrium position $\vec{x}_0$.
- Record $(\vec{\theta}, \vec{F})$ samples at 40 Hz.
- After the recording cache is full, generate a model and try again.

Bias Force Learning Results

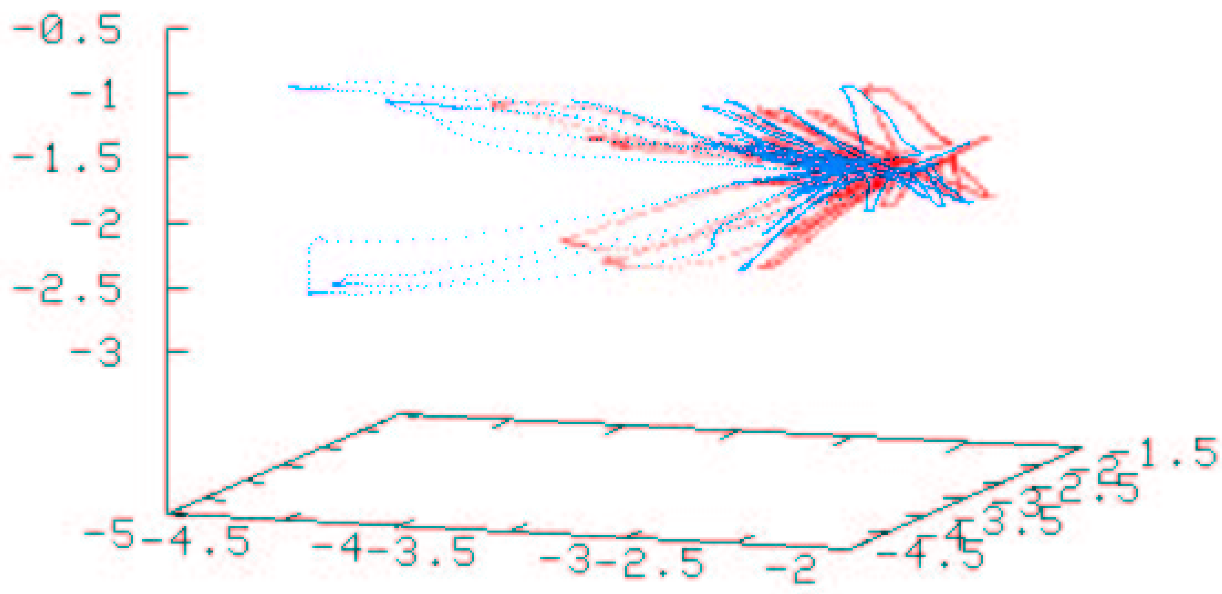
Only tried on lower three joints, $R^3 \rightarrow R^3$: tractable, graphable.





(Model learned on fourth run; 10,000 samples.)

Bias Force Learning Results



Effective workspace did increase:

- Maximum θ range: [2.8353, 3.2693, 2.7429]
- $(\vec{\theta}_{max} - \vec{\theta}_{min})$ during first run: [1.7332, 2.5049, 1.7598]
- $(\vec{\theta}_{max} - \vec{\theta}_{min})$ during fourth run: [2.6859, 2.5039, 2.0216]

Practical Limitations of Parzen Estimates

- Takes lots of samples to cover a 6-dimensional space.
- Samples are cheap, easy to collect, but...
- Calculating estimates takes time:
  (sole process, 800MHz PIII)
  - 10,000 samples, $R^6 \to R^6$: 0.0083ms = 120Hz
  - 100,000 samples, $R^6 \to R^6$: 0.076ms = 13Hz

Episodic Learning (i.e. “off-line”)

1. Record data.
2. Generate model.
3. Use model, while recording more data.
4. Merge model (discounted) and new data into new model.
5. Rinse, repeat.

Why?

- Can optimize model so that estimation is very fast (real-time controller).
- Can take as long as necessary to generate/update model, on a different processor, perhaps even while “sleeping”.

Trade Space for Time (and accuracy)

- Using collected samples, precompute y estimates over a lattice spanning the input space.
- For some input $\vec{x}$, compute $\vec{y}(\vec{x})$ by interpolating between the lattice points which surround $\vec{x}$.
- For N-dimensional domain, requires interpolation over (only) 2^N lattice points.

Drawbacks:

- For K lattice points per linear dimension, requires K^N total!

$$K = 5,\ N = 9 \Longrightarrow \sim 2 \times 10^6$$

- To generate, need to perform Parzen estimate for each lattice point!

Hyperspheres vs. Hypercubes

For S samples, S evaluations of $g()$ per lattice point.

OR, K^N evaluations of $g()$ per sample.

For sample $(\vec{x}_i, \vec{y}_i)$, only consider lattice points $\vec{x}$ such that $|\vec{x} - \vec{x}_i| < r_{max}$, where $r_{max} \simeq 2.5\sigma$.

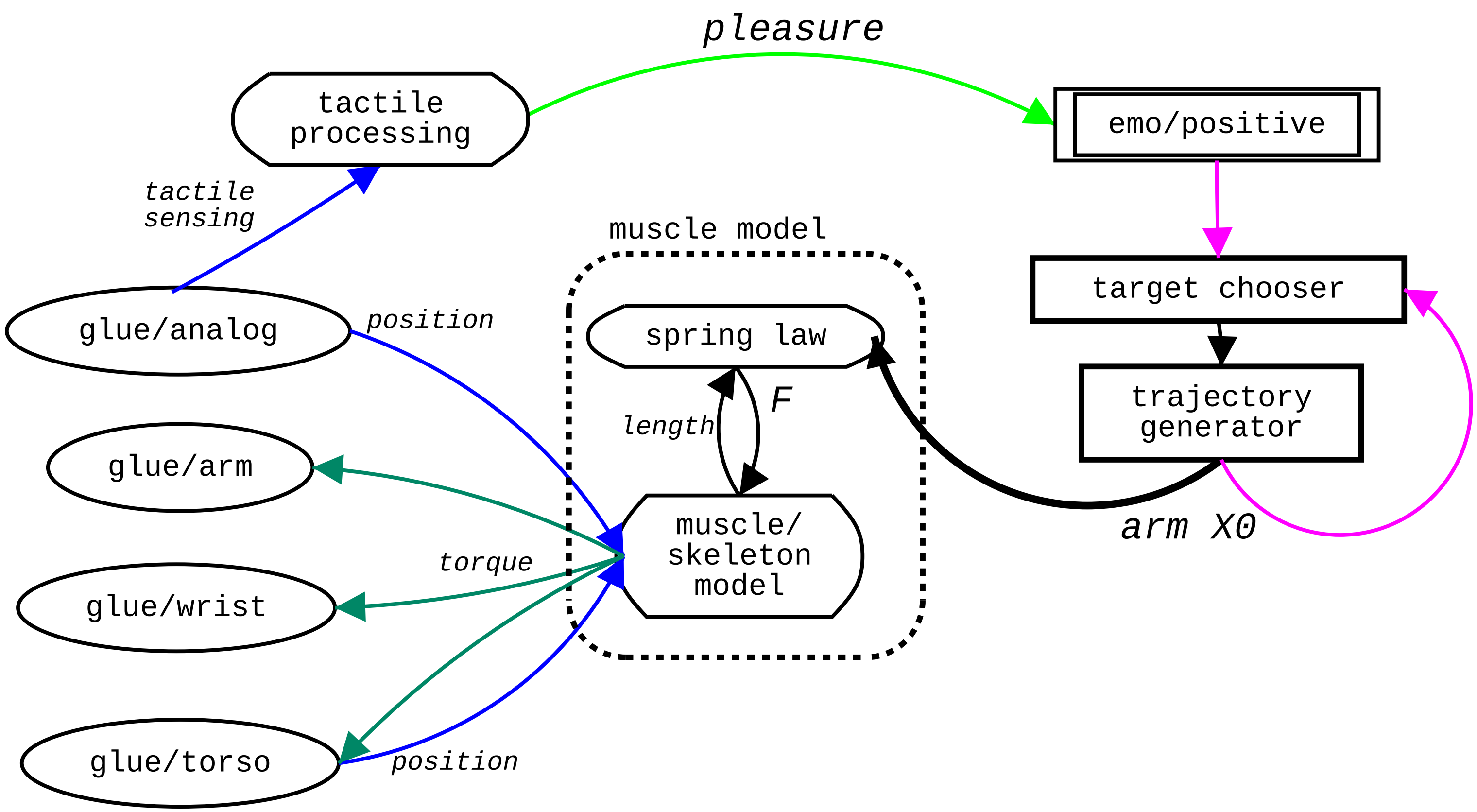
- For $r_{max} \approx \frac{1}{2}K$, volume of sphere grows as $2^{-N}K^N$.
- Matter of fact, volume of unit sphere drops as $\frac{1}{n!}$.

More Problems

- Still, a **lot** of memory:
  - $R^9 \rightarrow R^6$, $K = 5 \;\; \Rightarrow \;\; 2\, million\, pts \times 6 \times 4\, bytes \;\; \Rightarrow \;\; 48 MB$
  - Even more bytes: weighting factor, variance...
- What if I actually implement polyarticular muscles?
  Then $\vec{\theta} \mapsto \vec{F}_0$ is no longer a function!

Task 2: Blind Reaching (hardwired)

Another exercise in learning a distribution. . .



Phase 1: Do random reaching, but modify target distribution in response to tactile feedback.

Task 2: Blind Reaching (hardwired)

Phase 2: Add negative reinforcement from joint pain.

Phase 3: Incorporate bias force model.

- Note that as F_0 is learned, the relation of $\vec{x}_0$ to $\vec{\theta}$ will change. The distribution will need to evolve as F_0 develops.

Phase 4: Undo hardwiring.

Discovering models on the fly (“data mining”)

- Look at two (not so) randomly chosen ports, $\vec{x}$ and $\vec{y}$.
  - $\ldots$ the “hypothesis”
- Figure out appropriate time scales (FFT?).
- Decide if data streams are correlated:
  - Event correlation: simultaneous activity
  - Minimum in entropy of $(\vec{x}(t), \vec{y}(t + \Delta t))$ set versus delay Δt.
  - Construction of reliable function $\vec{x} \rightleftharpoons \vec{y}$.

Task 3: Hand-(Arm-Head)-Eye Correlation

“Learn where the hand is in visual space”, **or**
“Learn how to move the hand to a point in visual space”

$$(\vec{r}, \vec{e}, \vec{h}) \mapsto \vec{\theta}$$

$$(\vec{r}, \vec{e}, \vec{h}) \mapsto \vec{x}_0$$

Serves two purposes:

- Motor model: generate reach target to a visual stimulus.
- Sensory model: predict where the hand will be seen; explain a visual stimulus.

Two classes of models

- *control* — causal relationship between two perceptual or motor data streams
- *policy* — learning the reward/reinforcement from an activity or a context

Mechanisms for using models

- *implementing policy* — **inhibition**: subsuming random or reflexive control with context-sensitive initiation.
  - (+) reward — generative inhibition: let something else take over
  - (–) reward — protective inhibition: don't do something stupid
- *prediction/feed-forward control* — ???

Some More Modules in the works

- *reflex*
- *activator/context*
- *inhibitor*
- *subsumer*