Probabilistic Tracking

Marcus A. Brubaker
mbrubake@cs.toronto.edu

Outline

The goal is to introduce the fundamentals of probabilistic, model-based tracking for computer vision

- Background
- Basic algorithms
  - Kalman filter
  - Particle filter
  - Sequential importance sampling
  - MAP/ML
  - Annealed “particle filter”

Problem

Tracking is the process of inferring information given a sequence of observations over time

What might we infer?

- Shape (e.g., size, curvature)
- Appearance (e.g., colour, texture)
- State (e.g., position or orientation)
- Dynamics (e.g., accelerating, stopped, etc)

Problem

Tracking is the process of inferring information given a sequence of observations over time

What are our observations?

- RGB images (e.g., standard cameras)
- Depth images (e.g., Kinect, Time-of-flight sensors)
- Transformations of the above (e.g., colour or gradient histograms, edges, feature points and descriptors, object detections, etc)

Problem

Tracking is the process of inferring information given a sequence of observations over time

What makes it hard?

- Observations are often noisy, misleading, indirect, cluttered, incomplete
- Data association is not always clear
- Problem is often inherently ambiguous, ie, no single right answer

Basics of Probability

Let a and b be continuous random variables

Probability density function $p(a)$ describes the *density* of probability at a point and is also called the probability distribution of a

Will often refer to $p(a)$ as the "probability of a" but this is not quite accurate, really mean the "probability density of a".

PDFs must satisfy: $p(a) \geq 0 \qquad \int p(a)da = 1$

Joint probability density function $p(a, b)$ is the probability density of both random variables taking on a particular value.

Conditional probability density function $p(a|b)$ is the probability density of one random variable given that the value of the other random variable is known.

Basics of Probability

Some properties that we'll use a lot:

Factorization:

$$p(a, b) = p(a|b)p(b) = p(b|a)p(a)$$

Bayes' Rule:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}$$

Marginalization:

$$p(a) = \int p(a, b)db$$

Independence: if, and only if, a and b are independent then

$$p(a, b) = p(a)p(b)$$

Basics of Probability

If variables are discrete, the same properties hold with small modifications

Probability function $p(a)$ where

$$0 \leq p(a) \leq 1 \qquad \sum_a p(a) = 1$$

Marginalization:

$$p(a) = \sum_b p(a, b)$$

Basics of Probability

The **expected value** of a function of a random variable is the average value

$$E[f(x)] = \int f(x)p(x)dx$$

Some common uses for expectation

Mean

$$\mu = E[x]$$
$$= \int xp(x)dx$$

(Co-)variance

$$\Sigma = E[(x-\mu)(x-\mu)^T]$$
$$= \int (x-\mu)(x-\mu)^T p(x)dx$$

Probabilistic Formulation of Tracking

Denote the **state** at time t as $\mathbf{x_t}$

- $\mathbf{x_t}$ might be the position and orientation of an object, the pose of a human body, the current appearance model of an object, etc

For convenience we'll denote a sequence of states from time 1 to t as

$$\mathbf{x_{1:t}} = (\mathbf{x_1}, \ldots, \mathbf{x_t})$$

Probabilistic Formulation of Tracking

Denote the **observation** at time t as $\mathbf{z_t}$

- Observations could include an entire image, filter responses (e.g., gradients), optical flow, feature points, detector responses, etc

Similarly we'll denote a sequence of observations from time 1 to t as

$$\mathbf{z_{1:t}} = (\mathbf{z_1}, \ldots, \mathbf{z_t})$$

Probabilistic Formulation of Tracking

We can now define what we're fundamentally interested in

The **posterior distribution** at time t is the conditional probability distribution of the states given the observations

$$p(\mathbf{x_{1:t}}|\mathbf{z_{1:t}})$$

Sometimes we don't need the full posterior. Instead we look at the **filtering distribution** at time t which is the probability of the most recent state given the observations up to that time

$$p(\mathbf{x_t}|\mathbf{z_{1:t}}) = \int \cdots \int \mathbf{p}(\mathbf{x_{1:t}}|\mathbf{z_{1:t}})\mathbf{dx_{t-1}} \cdots \mathbf{dx_1}$$

Probabilistic Formulation of Tracking

Given these distributions we can answer the key tracking questions:

- What is the most likely state?

$$\max_{\mathbf{x_t}} p(\mathbf{x_t}|\mathbf{z_{1:t}})$$

- What does the average state look like?

$$E[\mathbf{x_t}|\mathbf{z_{1:t}}] = \int \mathbf{x_t} \mathbf{p}(\mathbf{x_t}|\mathbf{z_{1:t}}) \mathbf{dx_t}$$

- Are there multiple, similarly probable states?

$$\frac{\partial}{\partial \mathbf{x_t}} p(\mathbf{x_t}|\mathbf{z_{1:t}}) = \mathbf{0}$$

Probabilistic Formulation of Tracking

We can ask similar questions about the motion (i.e., sequence of states):

- What is the most likely motion?

$$\max_{\mathbf{x_{1:t}}} p(\mathbf{x_{1:t}}|\mathbf{z_{1:t}})$$

- What does the average motion look like?

$$E[\mathbf{x_{1:t}}|\mathbf{z_{1:t}}] = \int \mathbf{x_{1:t}} \mathbf{p}(\mathbf{x_{1:t}}|\mathbf{z_{1:t}}) \mathbf{dx_t}$$

- Are there multiple, similarly probable motions?

$$\frac{\partial}{\partial \mathbf{x_{1:t}}} p(\mathbf{x_{1:t}}|\mathbf{z_{1:t}}) = \mathbf{0}$$

Probabilistic Formulation of Tracking

Lets be more specific about what the problem at hand is

Probabilistic tracking is the problem of (efficiently) computing the posterior (or filtering) distribution at each time t

Problem: what exactly is the posterior distribution?

Defining the Posterior

We can apply **Bayes' rule** to the posterior to express it in terms of:

The **likelihood** measures the how well the observations match the state.

The **prior** expresses how likely the states are without observations.

The **evidence** is a normalizing factor which doesn't depend on the states.

$$p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}{p(\mathbf{z}_{1:t})}$$

likelihood prior evidence

Defining the Posterior: Model Assumptions

To construct a likelihood and prior, we make some assumptions

The prior (or motion model) is assumed to be a **first-order Markov model** where the past is independent of the future given the present

$$p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Using this assumption we can rewrite the prior

$$\begin{aligned} p(\mathbf{x}_{1:t}) &= p(\mathbf{x}_t|\mathbf{x}_{1:t-1})p(\mathbf{x}_{1:t-1}) && \textit{(factorization)} \\ &= p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{1:t-1}) && \textit{(first-order Markov)} \\ &= p(\mathbf{x}_1)\prod_{i=2}^{t} p(\mathbf{x}_i|\mathbf{x}_{i-1}) \end{aligned}$$

Defining the Posterior: Model Assumptions

To construct a likelihood and prior, we make some assumptions

The observations are **conditionally independent** given the states

$$p(\mathbf{z}_{1:t}|\mathbf{x}_{1:t}) = \prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i)$$

Defining the Posterior: Model Assumptions

These assumptions serve three purposes

1. The prior can be easily specified with a single step motion model

$$p(\mathbf{x}_t|\mathbf{x}_{t-1})$$

2. The likelihood can be easily specified with a single observation model

$$p(\mathbf{z}_t|\mathbf{x}_t)$$

3. The posterior and filtering distributions can be written in a computationally convenient, recursive form

Defining the Posterior: Recursive form

Posterior distribution:

$$p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})}{p(\mathbf{z}_{1:t})} \qquad \textit{(Bayes' rule)}$$

$$\propto p(\mathbf{z}_{1:t}|\mathbf{x}_{1:t})p(\mathbf{x}_{1:t})$$

$$= \left(\prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i)\right) p(\mathbf{x}_1) \prod_{i=2}^{t} p(\mathbf{x}_i|\mathbf{x}_{i-1}) \qquad \textit{(assumptions)}$$

$$\propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{1:t-1}|\mathbf{z}_{1:t-1}) \qquad \textit{(recursion)}$$

previous posterior

Defining the Posterior: Recursive form

Filtering distribution:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \int p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}) d\mathbf{x}_{1:t-1} \quad \textit{(marginalization)}$$

$$\propto \int p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{1:t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{1:t-1}$$

(first-order Markov)

$$= p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) p(\mathbf{x}_{1:t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{1:t-1}$$

$$= p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{1:t-1} \quad \textit{(factorization)}$$

$$= p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) \quad \textit{(marginalization)}$$

prediction distribution

Defining the Posterior: Recursive form

The **prediction distribution** is the distribution of the next state given the previous observations

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_{t-1}, \mathbf{x}_t|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}$$

$$= \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}$$

Defining the Posterior: Recursive form

The recursive form is important theoretically and computationally:

- All information contained in past observations is represented in the previous posterior distribution
- Given only the previous posterior distribution and a new observation, we should be able to compute the next posterior distribution

Without it we would potentially need to reference all the past observations to create the next posterior distribution

Defining the Posterior: Filtering vs Smoothing

There are two different tasks to consider with temporal observations

Filtering is what we've mostly been talking about, looking at the distribution of the current state given the past (online)

Smoothing looks at the distribution of the current state given both the past and the future (batch)

The **smoothing distribution** is

$$p(\mathbf{x}_\tau|\mathbf{z}_{1:t}) \propto \frac{p(\mathbf{z}_\tau|\mathbf{x}_\tau)p(\mathbf{x}_\tau|\mathbf{z}_{1:\tau-1})p(\mathbf{x}_\tau|\mathbf{z}_{\tau+1:t})}{p(\mathbf{x}_\tau)}$$

Defining the Posterior: Filtering vs Smoothing

There are two different tasks to consider with temporal observations

Filtering is what we've mostly been talking about, looking at the distribution of the current state given the past (online)

Smoothing looks at the distribution of the current state given both the past and the future (batch)

We're going to focus on filtering

Defining the Posterior: Likelihoods

There are *many* different kinds of likelihoods that have been used in computer vision for tracking

We can't cover them all here but just to give you some idea:

- Feature points
- Image templates or detector responses
- Histograms of colour, gradients, etc
- Image edges
- Background subtraction
- ...

Constructing the "right" likelihood for a tracking problem is probably one of the most important pieces but it's also one of the hardest

Defining the Posterior: Motion Models

While there are a large number of likelihoods, the number of (commonly used) motion models are much smaller and we'll talk about the most common

Before we talk about specific examples, lets go over why motion models are important

- Observations can be noisy or misleading
- Posteriors often have multiple, equally likely, modes or explanations

A good motion model (or prior) helps a tracker choose between competing explanations and observations when they're not helpful

Defining the Posterior: Motion Models

What makes a *good* motion model?

Two main qualities:

- Generic - applicable to a wide number of tasks, actions, subjects, etc
- Informative - provides good information about good and bad motions

Bearing this in mind, lets look at some commonly used motion models

Defining the Posterior: Motion Models

For simplicity assume that $\mathbf{x}_t$ is the 2D position of an object in the image

The simplest model is *Brownian motion*, ie, random Gaussian noise

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \eta \qquad \eta \sim \mathcal{N}(\mathbf{0}, \Sigma)$$

In other terms

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = |2\pi\Sigma|^{-0.5} \exp\left(-\frac{1}{2}(\mathbf{x}_t - \mathbf{x}_{t-1})^T\Sigma^{-1}(\mathbf{x}_t - \mathbf{x}_{t-1})\right)$$

Defining the Posterior: Motion Models

Brownian motion is fairly *generic*, only assuming that the current and previous positions are nearby

It's also very "jerky", more than we really expect motion to be, making it relatively *uninformative*

Defining the Posterior: Motion Models

Other common motion models tend to be *higher order*, that is, they look at motion over several frames

But this violates the first-order Markov assumption!

Can always “stack” the state to include some history:

$$\mathbf{y}_t = \begin{pmatrix} \mathbf{x}_t \\ \mathbf{x}_{t-1} \\ \vdots \\ \mathbf{x}_{t-p} \end{pmatrix}$$

Defining the Posterior: Motion Models

Other motion models:

Constant Velocity:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + d(\mathbf{x}_{t-1} - \mathbf{x}_{t-2}) + \eta$$

Damped Spring:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \kappa(\hat{\mathbf{x}} - \mathbf{x}_{t-1}) + d(\mathbf{x}_{t-1} - \mathbf{x}_{t-2}) + \eta$$

Auto-Regressive:

$$\mathbf{x}_t = \mu + \sum_{i=1}^{p} A_i \mathbf{x}_{t-i} + \eta$$

Defining the Posterior: Motion Models

Models for more complex motions (or states) can be learned or, in some cases, built from first principles

- To learn a motion model, can effectively treat it as regression problem
- When tracking 3D motion, can use physics to derive motion models

Inference: Kalman Filter

Now that we have a posterior (in terms of a likelihood and motion model), the task of tracking is to effectively compute and use this posterior

Unfortunately, computing and analyzing a general posterior is hard so we typically use approximations

However, these is one case where the posterior can be computed easily in closed form

Inference: Kalman Filter

Assume the entire model is Gauss-Linear

That is, the motion model is linear plus Gaussian noise and the observations are linear in the state plus Gaussian noise

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \eta_x \qquad \eta_x \sim \mathcal{N}(\mathbf{0}, \Sigma_x)$$

$$\mathbf{z}_t = B\mathbf{x}_t + \eta_z \qquad \eta_z \sim \mathcal{N}(\mathbf{0}, \Sigma_z)$$

Inference: Kalman Filter

The transition and observation densities can be written as

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|A\mathbf{x}_{t-1}, \Sigma_x)$$
$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{z}_t|B\mathbf{x}_t, \Sigma_z)$$

where

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Inference: Kalman Filter

Then it can be shown that the prediction distribution is also Gaussian

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})d\mathbf{x}_{t-1}$$

$$= \int \mathcal{N}(\mathbf{x}_t|A\mathbf{x}_{t-1}, \Sigma_x)\mathcal{N}(\mathbf{x}_{t-1}|\mu_{t-1}^{+}\Sigma_{t-1}^{+})d\mathbf{x}_{t-1}$$

$$= \mathcal{N}(\mathbf{x}_t|\mu_t^{-}, \Sigma_t^{-})$$

Inference: Kalman Filter

Using that, the posterior at the next time is:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = cp(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t-1})$$

$$= c\mathcal{N}(\mathbf{z}_t|B\mathbf{x}_t, \Sigma_z)\mathcal{N}(\mathbf{x}_t|\mu_t^-, \Sigma_t^-)$$

$$= \mathcal{N}(\mathbf{x}_t|\mu_t^+, \Sigma_t^+)$$

Inference: Kalman Filter

The form of the means and variances are given below for reference (try to derive them on your own)

$$\mu_t^- = A\mu_{t-1}^+$$

$$\Sigma_t^- = A\Sigma_{t-1}^+ A^T + \Sigma_x$$

$$\mu_t^+ = \mu_t^- + K_t(\mathbf{z}_t - B\mu_t^-)$$

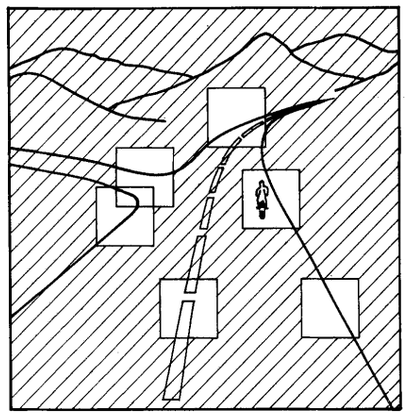
$$\Sigma_t^+ = (I - K_t)\Sigma_t^-(I - K_t)^T + K_t\Sigma_z K_t^T$$

$$K_t = \Sigma_{t-1}^- B^T(B\Sigma_{t-1}^- B^T + \Sigma_z)^{-1}$$

Inference: Kalman Filter

Kalman filter in computer vision

- Lane following [Dickmanns and Graefe, "*Dynamic Monocular Machine Vision*" 1988]



- Monocular 3D motion estimation [Broida et al, "*Recursive 3D Estimation from a Monocular Image Sequence*" 1990]

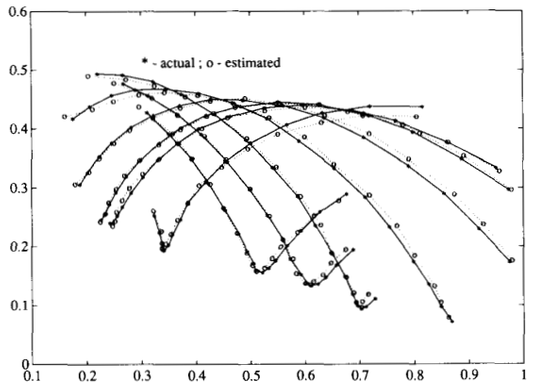


Fig. 7. Actual and estimated image point trajectories.

Inference: Beyond the Kalman Filter

Most tracking problems in computer vision don't fit nicely into a Gauss-Linear model

- Non-linear observations (e.g., perspective projection)
- Non-Gaussian observation noise (e.g., detection failures, false detections, appearance changes)
- Non-linear dynamics (e.g., human motion)

Kalman filter does not (directly) apply to these situations

Inference: Beyond the Kalman Filter

Approximate versions of the Kalman Filter have been derived for non-linear dynamics

Extended Kalman Filter (EKF): Approximate the non-linear dynamics with a local linearization of the dynamics (i.e., set A to be the Jacobian of the non-linear dynamics)

Unscented Kalman Filter (UKF): Instead of linearizing the dynamics, draw a number of samples, simulate the dynamics, and compute the mean and covariance of the new points

Both the EKF and the UKF have problems when the dynamics are not close to linear

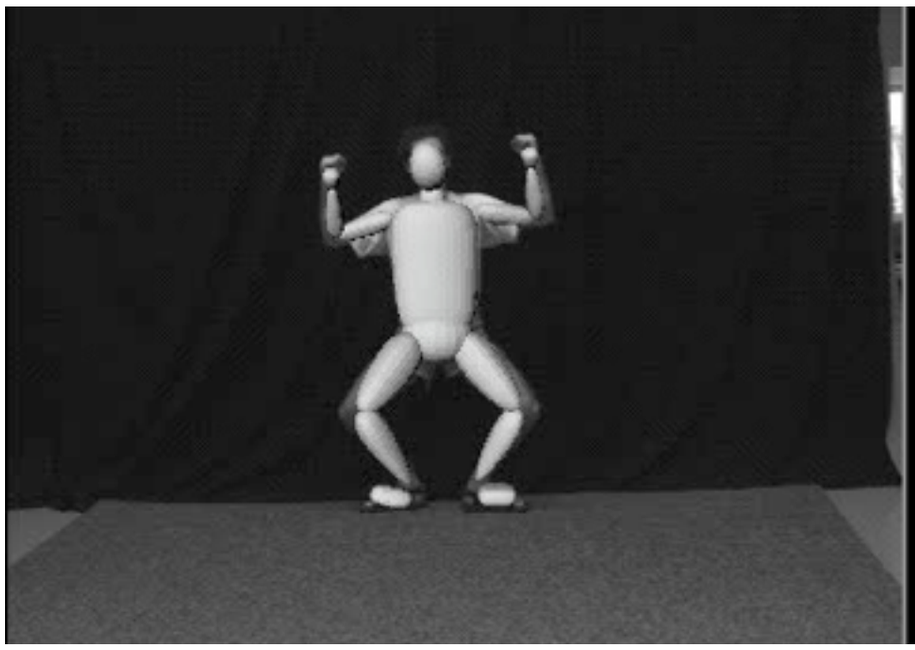
Inference: Beyond the Kalman Filter

The Kalman Filter has one fundamental flaw for computer vision problems

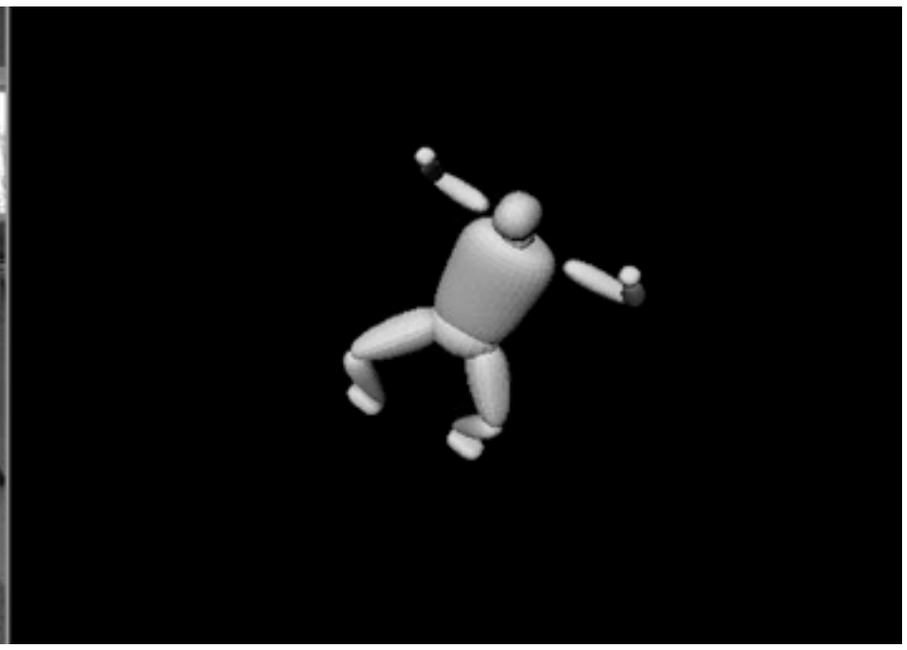
Posteriors in computer vision are almost never unimodal (much less Gaussian)!

Inference: Non-Gaussian Posteriors

image



3D model
(camera view)

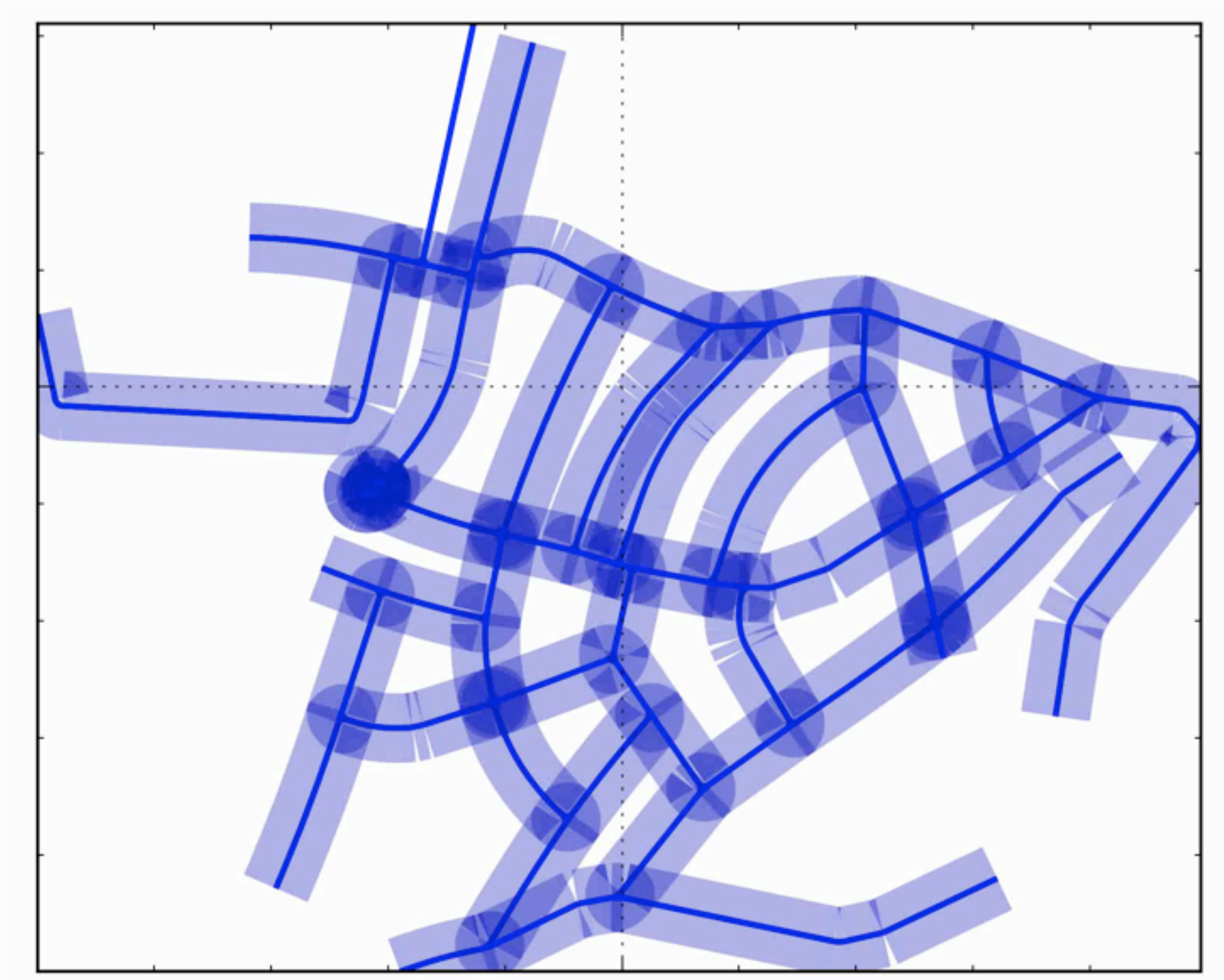


3D model
(top view)

Many 3D configurations may be consistent with a given image.

[courtesy of Cristian Sminchisescu]

Inference: Non-Gaussian Posteriors



Inference: Non-Gaussian Posteriors

So if we care about representing the posterior properly, we need an inference algorithm which can handle non-Gaussian distributions

But if we end up picking the most likely state at the end of the day, do we really need to worry about representing the full posterior?

YES: tracking (filtering) is a recursive process.
If we fail to represent important details now,
tracking may fail later!

Inference: Particle Filter

A particle filter performs inference by approximating the distribution with a (weighted) set of points and makes very few assumptions about the model

In it's simplest form it requires two things from the model:

- Draw samples from the motion model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$
- Evaluate the likelihood function $p(\mathbf{z}_t|\mathbf{x}_t)$

Inference: Particle Filter

A particle filter maintains a set of samples (or particles) which approximate the posterior distribution

If we can draw a set of samples $\mathcal{S}_t = \{\mathbf{x}_t^{(j)}\}$ where $\mathbf{x}_t^{(j)} \sim p(\mathbf{x}_t|\mathbf{z}_{1:t})$

The **Monte Carlo approximation** allows us to compute

$$E_{\mathcal{S}_t}[f(\mathbf{x}_t)] = \frac{1}{N}\sum_{j=1}^{N} f(\mathbf{x}_t^{(j)})$$

$$\overset{N\to\infty}{=} \int f(\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t})d\mathbf{x}_t$$

$$= E_{p(\mathbf{x}_t|\mathbf{z}_{1:t})}[f(\mathbf{x}_t)]$$

Inference: Particle Filter

We don't know how to draw samples from $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ in general

A particle filter uses **importance sampling** where samples are drawn from the **importance distribution** $q(\mathbf{x}_t)$ and weighted to correct the difference

That is

$$\mathcal{S}_t = \{\mathbf{x}_t^{(j)}, w_t^{(j)}\}_{j=1}^N$$

Where the weights are the ratio of the distributions

$$\begin{aligned} w_t^{(j)} &= w(\mathbf{x}_t) \\ &= \frac{p(\mathbf{x}_t|\mathbf{z}_{1:t})}{q(\mathbf{x}_t)} \end{aligned}$$

Inference: Particle Filter

Expectation with these weighted samples

$$E_{\mathcal{S}_t}[f(\mathbf{x}_t)] = \sum_{j=1}^{N} w_t^{(j)} f(\mathbf{x}_t^{(j)})$$

$$\overset{N\to\infty}{=} \int w(\mathbf{x}_t) f(\mathbf{x}_t) q(\mathbf{x}_t) d\mathbf{x}_t$$

$$= \int \frac{p(\mathbf{x}_t|\mathbf{z}_{1:t})}{q(\mathbf{x}_t)} f(\mathbf{x}_t) q(\mathbf{x}_t) d\mathbf{x}_t$$

$$= \int p(\mathbf{x}_t|\mathbf{z}_{1:t}) f(\mathbf{x}_t) d\mathbf{x}_t$$

$$= E_{p(\mathbf{x}_t|\mathbf{z}_{1:t})}[f(\mathbf{x}_t)]$$

Inference: Particle Filter

A set of properly weighted samples can also be thought of as a more direct approximation to the posterior

$$p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) \approx \sum_{j=1}^{N} w_{t-1}^{(j)} \delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(j)})$$

This is important because of the prediction distribution

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}) d\mathbf{x}_{t-1}$$

$$\approx \sum_{j=1}^{N} w_t^{(j)} p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)})$$

Inference: Particle Filter

Here is a simple version of a particle filter (aka, the Condensation algorithm [Isard and Blake, IJCV 1998]):

1) Given the filtering distribution at the previous time represented by a set of weighted samples $\mathcal{S}_t = \{\mathbf{x}_t^{(j)}, w_t^{(j)}\}_{j=1}^N$ and a new observation $\mathbf{z}_t$

2) For $i = 1, \ldots, N$ draw samples from the prediction distribution and weight them by the likelihood

a) Pick particle *j* with probability $w_{t-1}^{(j)}$

b) Sample a new state from the motion model

$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(j)})$$

c) Evaluate the likelihood of the new sample to get its weight

$$\hat{w}_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t^{(i)})$$

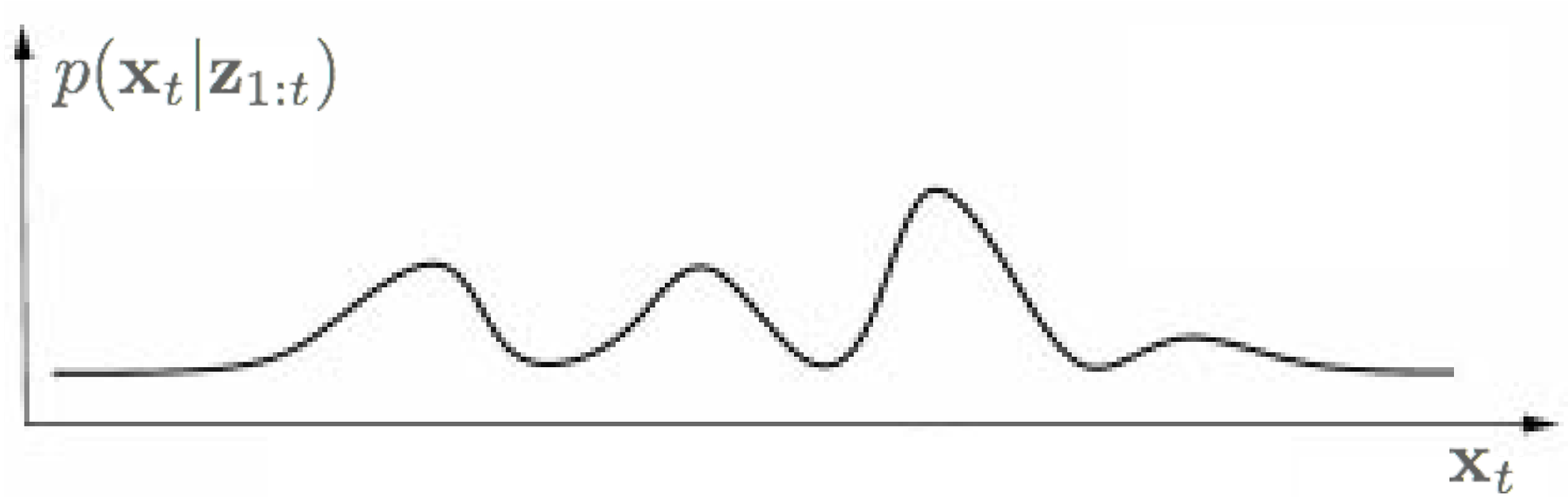
3) Normalize the weights

$$w_t^{(i)} = \left(\sum_{j=1}^N w_t^{(j)} \right)^{-1} w_t^{(i)}$$

Inference: Particle Filter

Lets take a closer look at what's happening here

Consider the following posterior:



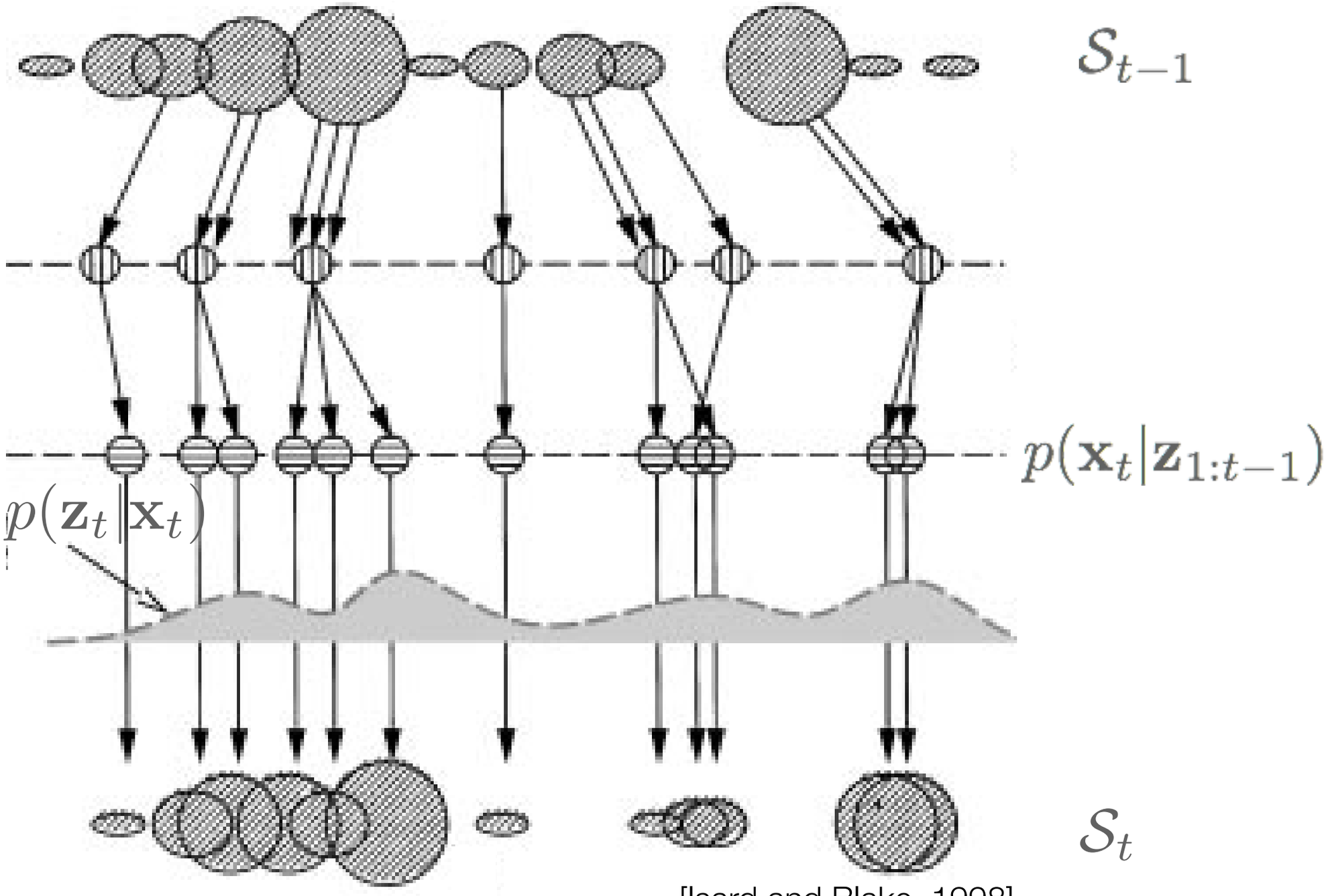
$$\mathcal{S}_t = \{\mathbf{x}_t^{(j)}, w_t^{(j)}\}_{j=1}^{N}$$

Inference: Particle Filter

Randomly select particles

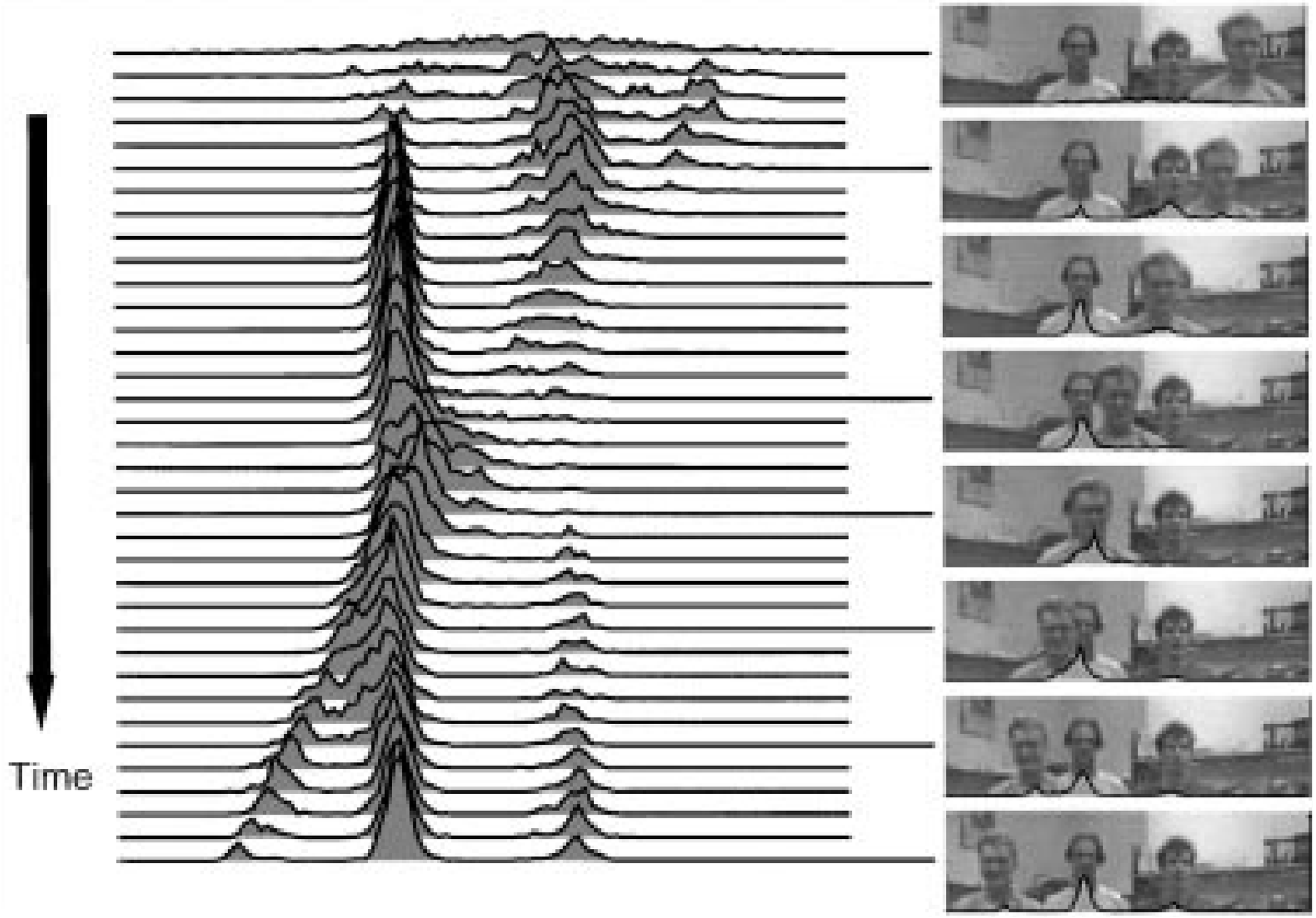
Sample from motion dynamics

Weight particles by likelihood



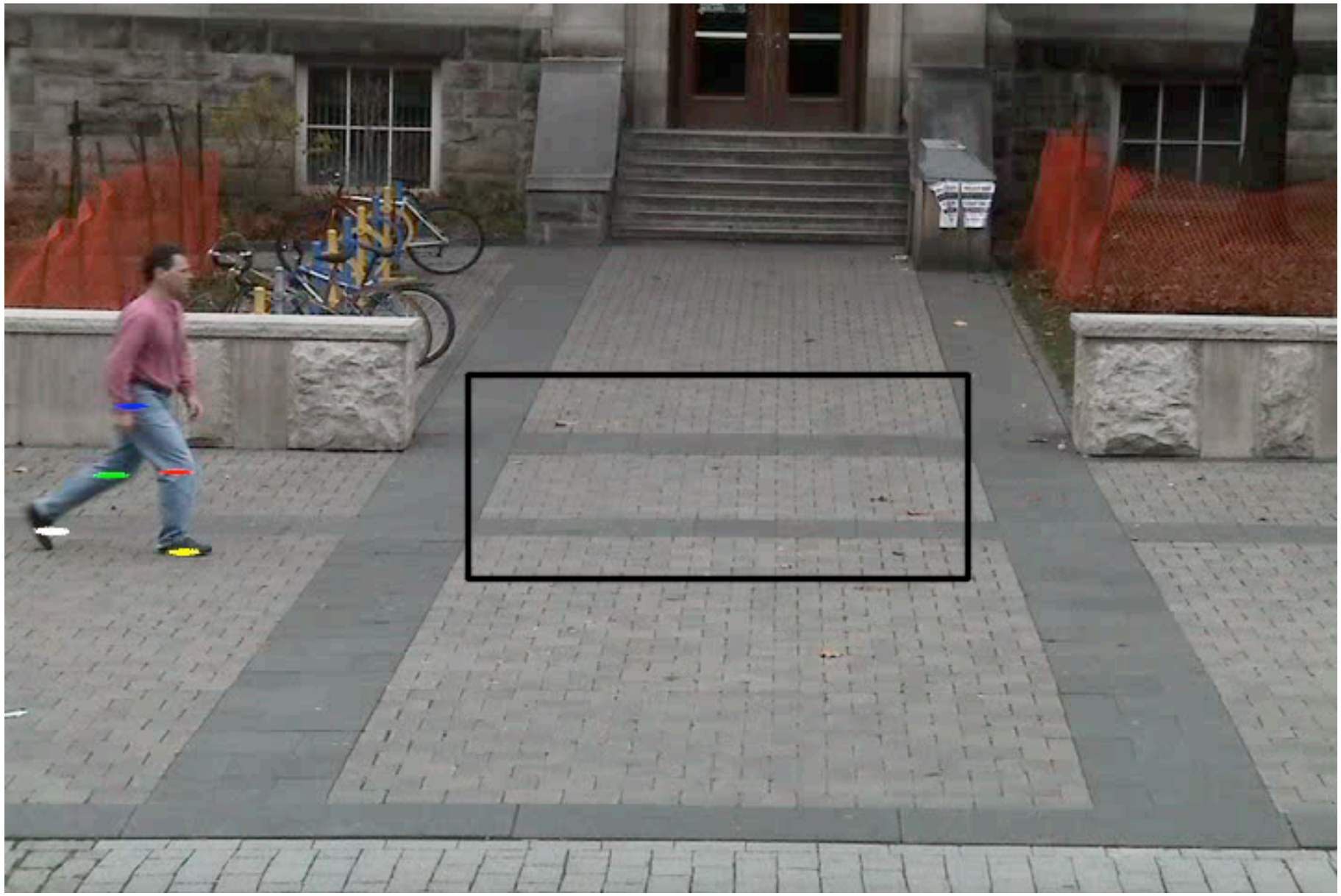
[Isard and Blake, 1998]

Inference: Particle Filter



[Isard and Blake, 1998]

Inference: Particle Filter



Inference: Particle Filter

Particle filters theoretically can perform inference on any model so long as we can sample from the motion model, evaluate the likelihood and use enough particles

So, how many particles do we need? How do we know if we have enough?

First, how many particles are needed to accurately represent a posterior

- Imagine a Gaussian distribution with a full covariance matrix, need more than d^2 independent samples to get a non-degenerate estimate of the covariance

What if we have multiple modes?

What if we have “heavy tails”?

What if we have a uniform distribution?

Inference: Particle Filter

The number of particles needed to represent a distribution depends on the *entropy* (kind of like the volume) of the distribution, not the dimensionality of the parameters

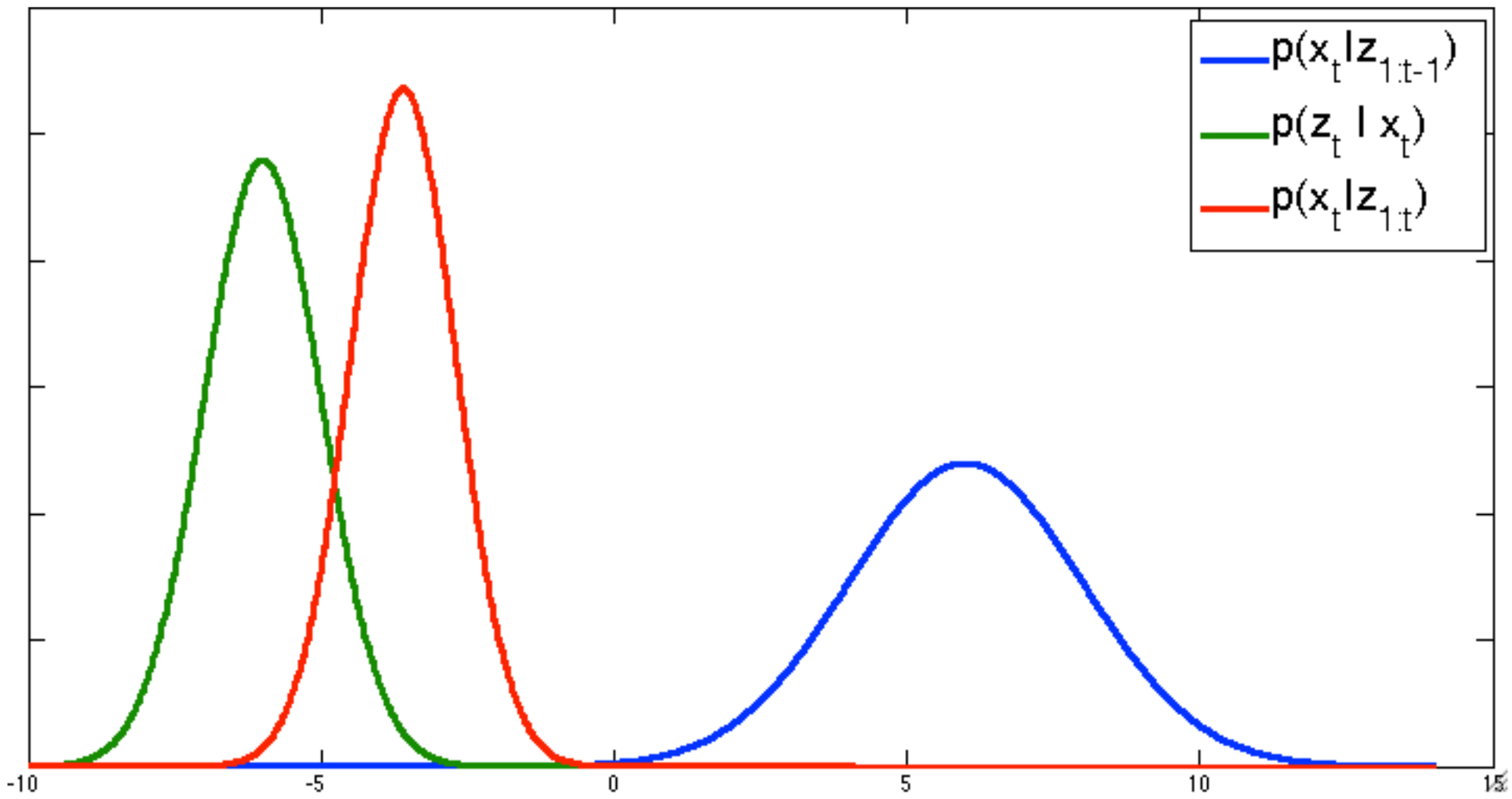
Bad news is that entropy typically scales linearly with the number of dimensions and the number of particles needed scales exponentially with the entropy

But this is just how many particles are needed to represent a distribution, it's actually a lower bound on how many are needed for a particle filter

Inference: Particle Filter

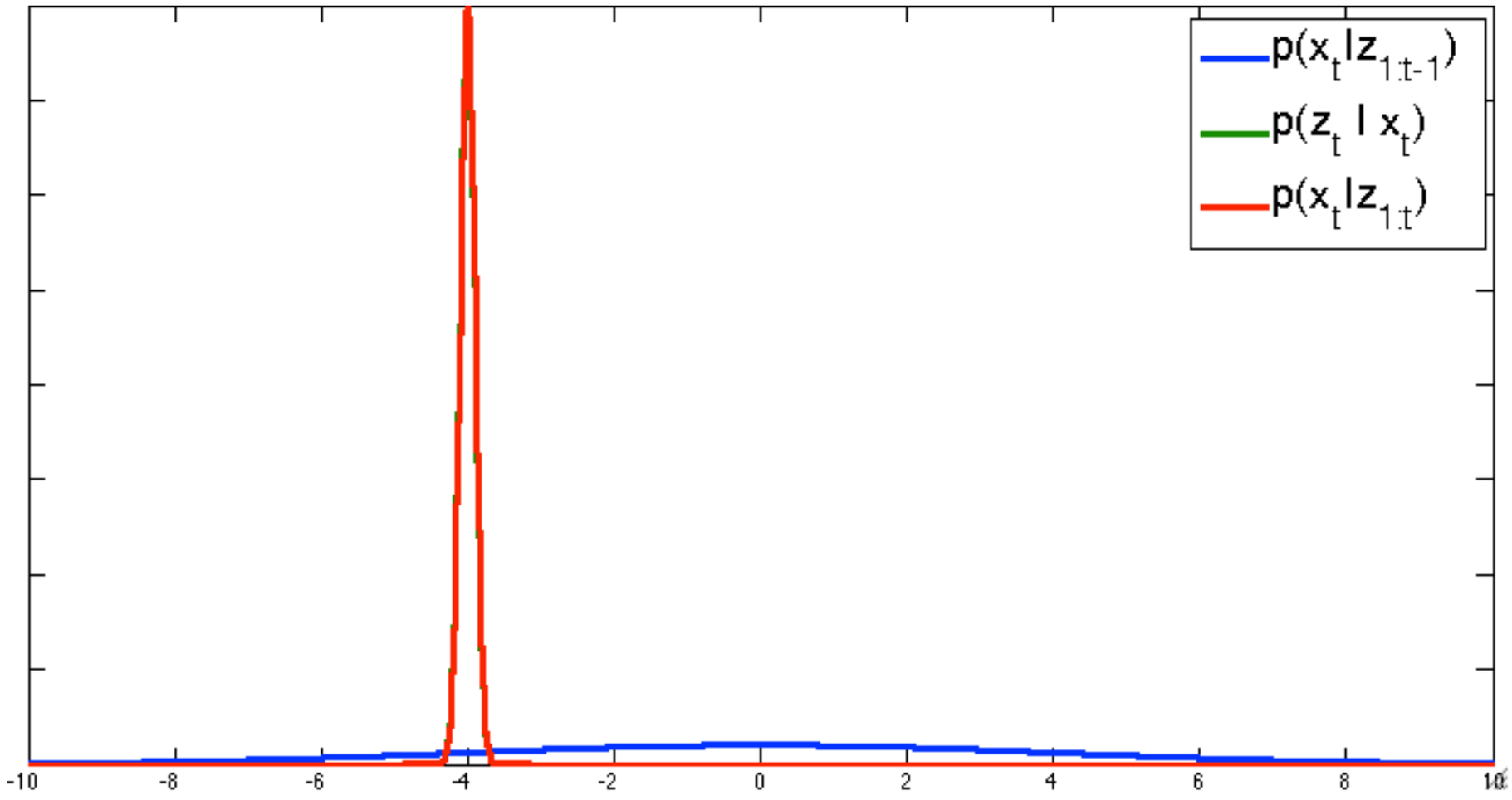
The problem is that the prediction distribution may be far away from the likelihood

Samples will never come close to the likelihood or the true posterior!



Inference: Particle Filter

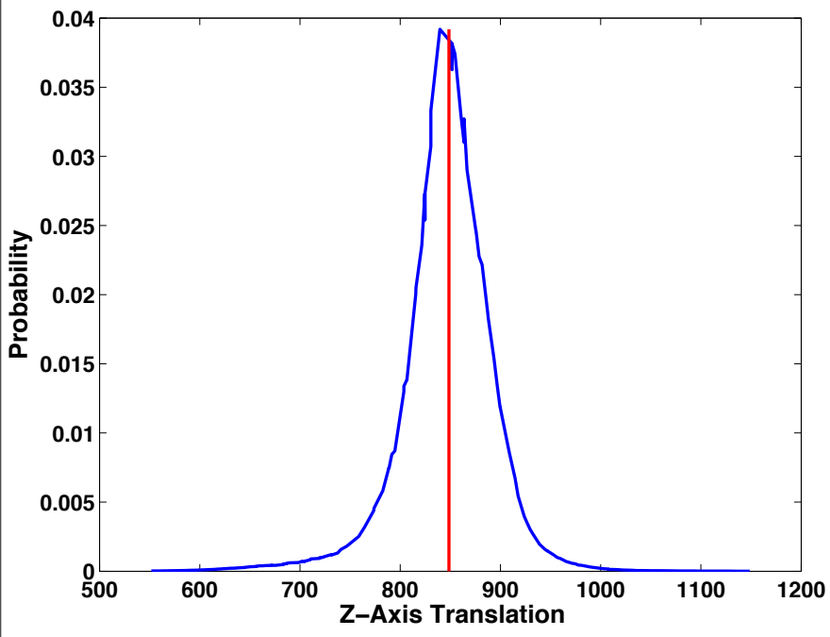
Or the prediction distribution may be broad and the likelihood very peaky

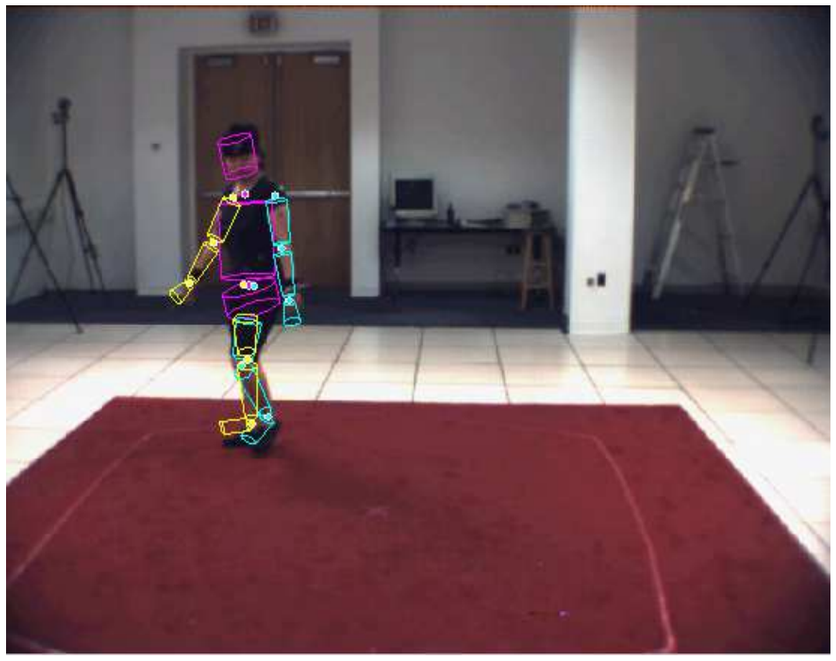


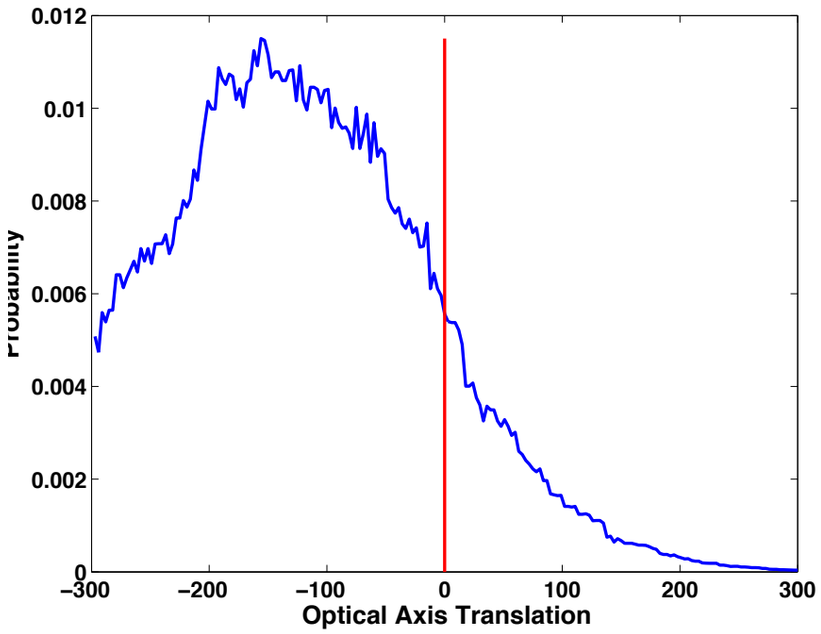
Inference: Particle Filter

Unfortunately, this is all too common in computer vision

This is from a likelihood function in human pose estimation







Inference: Particle Filter

How can we know if this happens?

If we have N weighted samples, how do we know if they're any good?

The **effective sample size** is an estimate of the number of independent samples

$$N_{eff} \approx \frac{1}{\sum_{j=1}^{N} (w^{(j)})^2}$$

If the weights are equal, i.e., $w^{(j)} = N^{-1}$, then $N_{eff} = N$

If one weight is large and the rest are small then $N_{eff} \approx 1$

Inference: Particle Filter

So, how many particles should you use?

As many as you can afford!

What if it still doesn't work?

Improve the model (i.e., reduce its entropy)

Broaden the prediction distribution by changing the motion model

Or...

Inference: Sequential Importance Sampling

The simple particle filter is an example of what's called a *Sequential Importance Sampling* algorithm

The more general class of algorithms will give us more flexibility

To see the connection, consider importance sampling from the posterior

$$p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t}) \propto \left(\prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i) \right) p(\mathbf{x}_{1:t})$$

Inference: Sequential Importance Sampling

If the *importance distribution* used is the prior $\mathbf{x}_{1:t} \sim p(\mathbf{x}_{1:t})$

To draw a sample $\mathbf{x}_{1:t}^{(i)}$ from the prior:

1) Draw sample from the initial distribution $\mathbf{x}_1^{(i)} \sim p(\mathbf{x}_1)$

2) For $\tau = 2, \ldots, t$ sample from the motion model

$$\mathbf{x}_\tau^{(i)} \sim p(\mathbf{x}_\tau | \mathbf{x}_{\tau-1}^{(i)})$$

Inference: Sequential Importance Sampling

The *importance weights* when using the prior

$$w(\mathbf{x}_{1:t}) = \frac{p(\mathbf{x}_{1:t}|\mathbf{z}_{1:t})}{p(\mathbf{x}_{1:t})}$$

$$\propto \frac{\left(\prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i)\right) p(\mathbf{x}_{1:t})}{p(\mathbf{x}_{1:t})}$$

$$= \prod_{i=1}^{t} p(\mathbf{z}_i|\mathbf{x}_i)$$

Inference: Sequential Importance Sampling

So to draw an importance sample and compute it's weight

1) Draw sample from the initial distribution $\mathbf{x}_1^{(i)} \sim p(\mathbf{x}_1)$
2) Evaluate the initial weight $w_1^{(i)} = p(\mathbf{z}_1|\mathbf{x}_1^{(i)})$
3) For $\tau = 2, \ldots, t$

$$\mathbf{x}_\tau^{(i)} \sim p(\mathbf{x}_\tau|\mathbf{x}_{\tau-1}^{(i)})$$

$$w_\tau^{(i)} = w_{\tau-1}^{(i)} p(\mathbf{z}_\tau|\mathbf{x}_\tau^{(i)})$$

Inference: Sequential Importance Sampling

So given a set of weighted importance samples at time *t-1*

$$\mathcal{S}_{t-1} = \{\mathbf{x}_{t-1}^{(j)}, w_{t-1}^{(j)}\}_{j=1}^{N}$$

Updating with a new observation is easy, for each sample j

1) $\mathbf{x}_t^{(j)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)})$

2) $w_t^{(j)} = w_{t-1}^{(j)} p(\mathbf{z}_t|\mathbf{x}_t^{(j)})$

As *t* increases the number of effective samples decreases, potentially very quickly. This is know as the problem of **particle depletion**.

Inference: Sequential Importance Sampling

To avoid particle depletion SIS can incorporate **resampling**

Given a set of particles $\mathcal{S}_t = \{\mathbf{x}_t^{(j)}, w_t^{(j)}\}_{j=1}^N$ we can create a new set

For $i = 1, \ldots, \hat{N}$

1) Sample index *j* with probability $w_t^{(j)}$

2) $\hat{\mathbf{x}}_t^{(i)} = \mathbf{x}_t^{(j)}, \quad \hat{w}_t^{(i)} = \hat{N}^{-1}$

If the original set was properly weighted, the new set $\hat{\mathcal{S}}_t = \{\hat{\mathbf{x}}_t^{(j)}, \hat{w}_t^{(j)}\}_{j=1}^{\hat{N}}$ is properly weighted as well

Inference: Sequential Importance Sampling

To get the simple particle filter (the Condensation algorithm), perform SIS with resampling at every every step (check this yourself!)

So what does the SIS perspective give us?

- We don't have to resample at every step, just take each particle, sample from the motion model and multiply the weight by the likelihood
- More importantly, we can use a different importance distribution

Inference: Sequential Importance Sampling

Here is a (more) general version of a particle filter based on these ideas

1) If desired (eg, based on ESS) resample the particle set

2) For each particle j in set $\mathcal{S}_{t-1} = \{\mathbf{x}_{t-1}^{(j)}, w_{t-1}^{(j)}\}_{j=1}^{N}$

a) $\mathbf{x}_t^{(j)} \sim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)}, \mathbf{z}_t)$

b) $w'^{(j)}_t = w_{t-1}^{(j)} \dfrac{p(\mathbf{z}_t|\mathbf{x}_t^{(j)})p(\mathbf{x}_t^{(j)}|\mathbf{x}_{t-1}^{(j)})}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)}, \mathbf{z}_t)}$

3) Normalize the weights

$$w_t^{(j)} = w'^{(j)}_t \left(\sum_{i=1}^{N} w'^{(i)}_t \right)$$

Inference: Sequential Importance Sampling

How do we set the importance/proposal distribution?

If $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)}, \mathbf{z}_t) = p(\mathbf{x}_t^{(j)}|\mathbf{x}_{t-1}^{(j)})$ then we're back to the basic algorithm

Ideally, we'd like $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)}, \mathbf{z}_t) \propto p(\mathbf{z}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(j)})$ but this generally won't be something we can sample from (if it was we'd be done!)

For specific problems we can often exploit domain knowledge to make better proposal distributions (eg, object detections, background blobs, etc)

Inference: Sequential Importance Sampling

Other particle filter/SIS variations:

Rao-Blackwellized Particle Filter: Reduce posterior entropy by analytically integrating out state variables [Khan et al, CVPR 2004]

Auxiliary Particle Filter: Peek ahead at the observation in order to build a better proposal distribution [Pitt and Shephard, JASA 1999]

SIS with MCMC: Use MCMC sampling to improve the particle set at each iteration [Liu and Chen, JASA 1998; Choo and Fleet, ICCV 2001]

MAP Tracking

There are many other tracking algorithms people have explored which are non-probabilistic

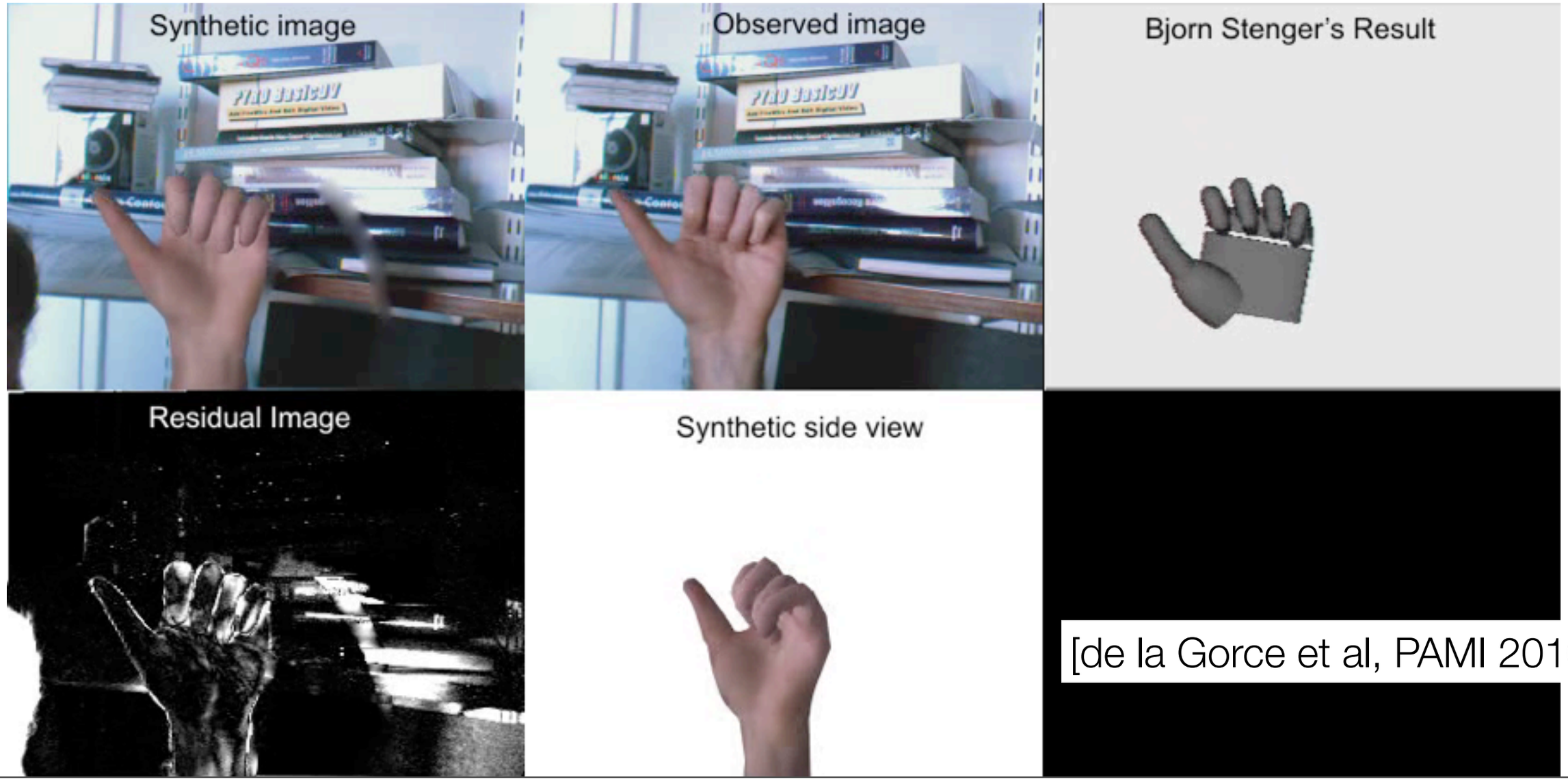
They focus on finding the maxima of the filtering distribution at each time, given the maxima at the previous time

$$\begin{aligned}\mathbf{x}_t^{MAP} &= \arg\max_{\mathbf{x}_t} p(\mathbf{x}_t|\mathbf{x}_{t-1}^{MAP}, \mathbf{z}_{1:t}) \\ &= \arg\max_{\mathbf{x}_t} p(\mathbf{z}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{x}_{t-1}^{MAP})\end{aligned}$$

These algorithms basically constitute different ways to maximize a function

MAP Tracking

Here is an example of a gradient based optimization MAP tracker which uses a detailed model of hand shape and appearance



MAP Tracking

Some algorithms that have been used successfully

- Gradient based local optimization
- Iterative-least squares local optimization
- Particle-swarm optimization
- Annealed “particle filter”

Annealed Particle Filter

The annealed particle filter is a particle filter-like algorithm, however it's sample set is *not* properly weighted

This is more than a theoretical problem: it can result in unexpected tracking failures when the likelihood function is ambiguous or misleading

However, it is one of the most popular algorithms in (generative) human pose tracking

Annealed Particle Filter

The APF begins the with basic particle filtering algorithm:

1) Given the filtering distribution at the previous time represented by a set of weighted samples $\mathcal{S}_t = \{\mathbf{x}_t^{(j)}, w_t^{(j)}\}_{j=1}^{N}$ and a new observation $\mathbf{z}_t$

2) For $i = 1, \ldots, N$ draw samples from the prediction distribution and weight them by the likelihood

a) Pick particle *j* with probability $w_{t-1}^{(j)}$

b) Sample a new state from the motion model

$$\mathbf{x}_t^{(i,0)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$$

Annealed Particle Filter

Then APF then iterates a process of diffusion and reweighting using an *annealed* version of the likelihood function

The diffusion distribution is typically a Gaussian $T_l(\mathbf{x}|\mathbf{x}') = \mathcal{N}(\mathbf{x}|\mathbf{x}', \Sigma_l)$

For $l = 1, \ldots, L$

For $i = 1, \ldots, N$, $w_t^{(i,l)} = \left(p(\mathbf{z}_t|\mathbf{x}_t^{(i,l-1)})\right)^{\alpha_l}$

Normalize the weights and resample the particle set

For $i = 1, \ldots, N$, $\mathbf{x}_t^{(i,l)} \sim T_l(\mathbf{x}_t|\mathbf{x}_t^{(i,l-1)})$

For $i = 1, \ldots, N$

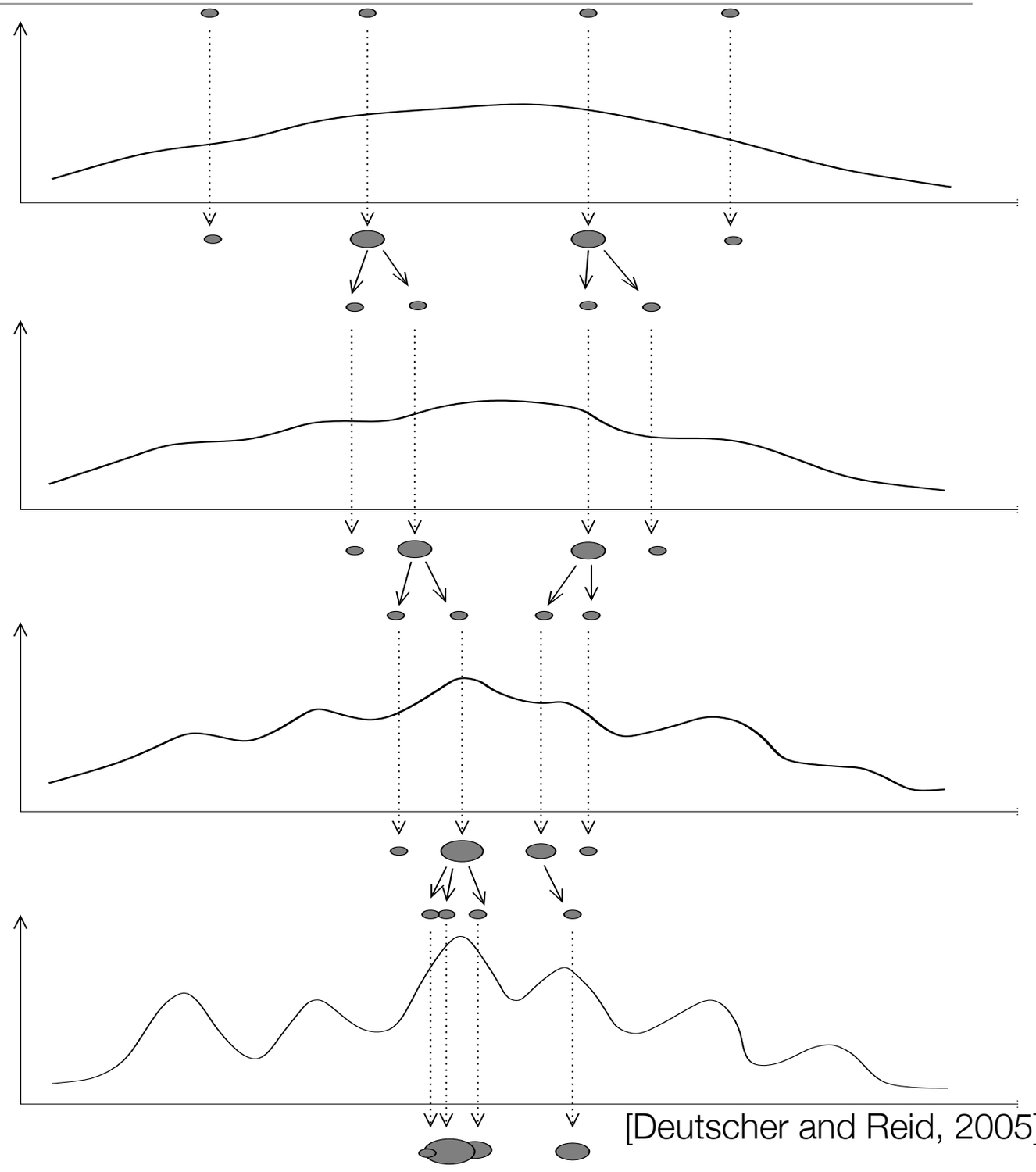
$w_t^{(i)} = p(\mathbf{z}_t|\mathbf{x}_t^{(i,L)})$

$\mathbf{x}_t^{(i)} = \mathbf{x}_t^{(i,L)}$

Annealed Particle Filter

This annealing process first smooths out the likelihood function and gradually roughens it, making it easier to search

As the annealing proceeds, the samples converge to a local mode



Annealed Particle Filter

The APF is typically run with fewer particles than the PF, but has a similar cost due to the annealing process

Because (in its original form) it only anneals the likelihood function, the motion model is effectively meaningless and the estimated states can make very large jumps

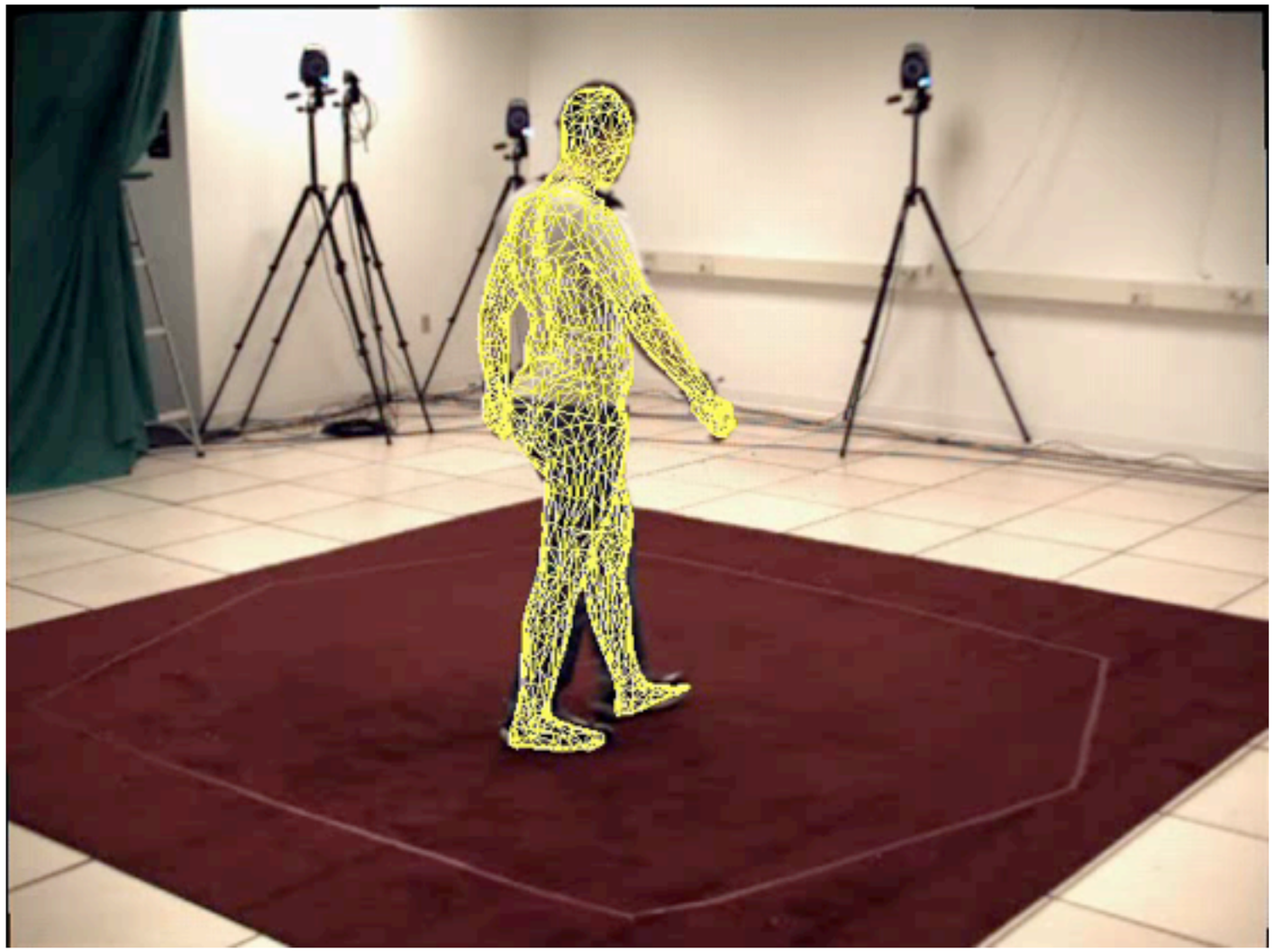
This can be fixed by incorporating the motion model into the weighting, e.g.,

$$w_t^{(i,l)} = \left(p(\mathbf{z}_t|\mathbf{x}_t^{(i,l-1)}) \sum_{j=1}^{N} w_{t-1}^{(i)} p(\mathbf{x}_t^{(i,l-1)}|\mathbf{x}_{t-1}^{(j)}) \right)^{\alpha_l}$$

The APF can also be altered to properly weight the particles [Gall et al, JMIV 2007]

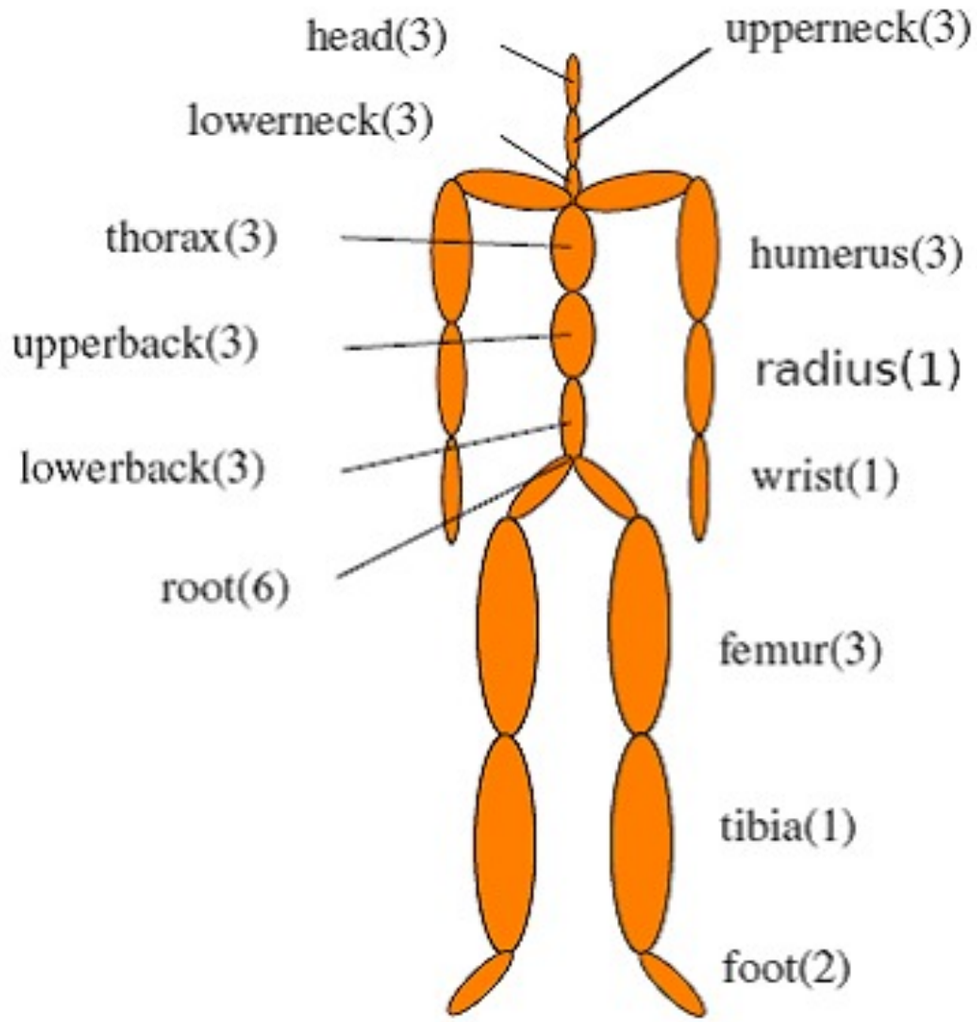
Annealed Particle Filter

The APF has had the most success in human motion estimation from multiple cameras (typically at least 4 with wide baseline)



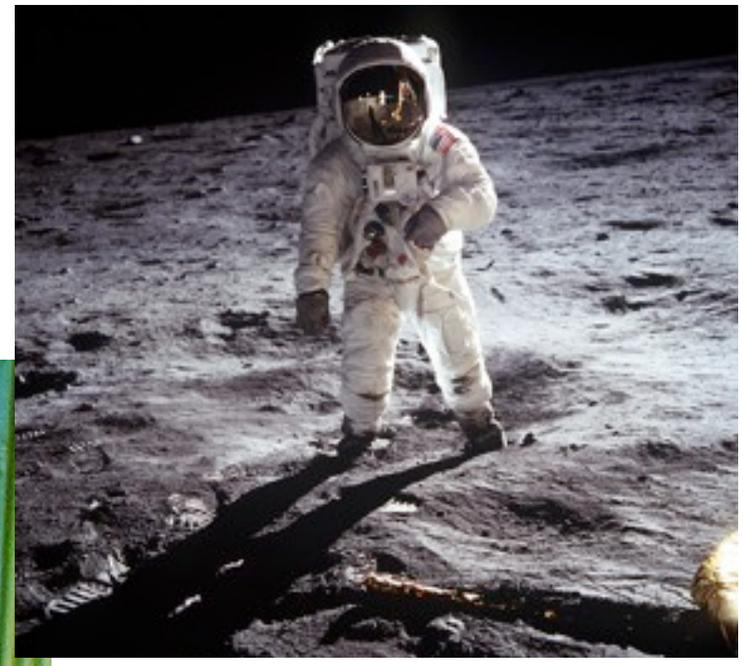
[Gall et al, IJCV 2010]

Challenges: High-dimensional pose



People have many degrees of freedom, comprising an articulated skeleton overlaid with soft tissue and deformable clothing.

Challenges: Appearance, size and shape



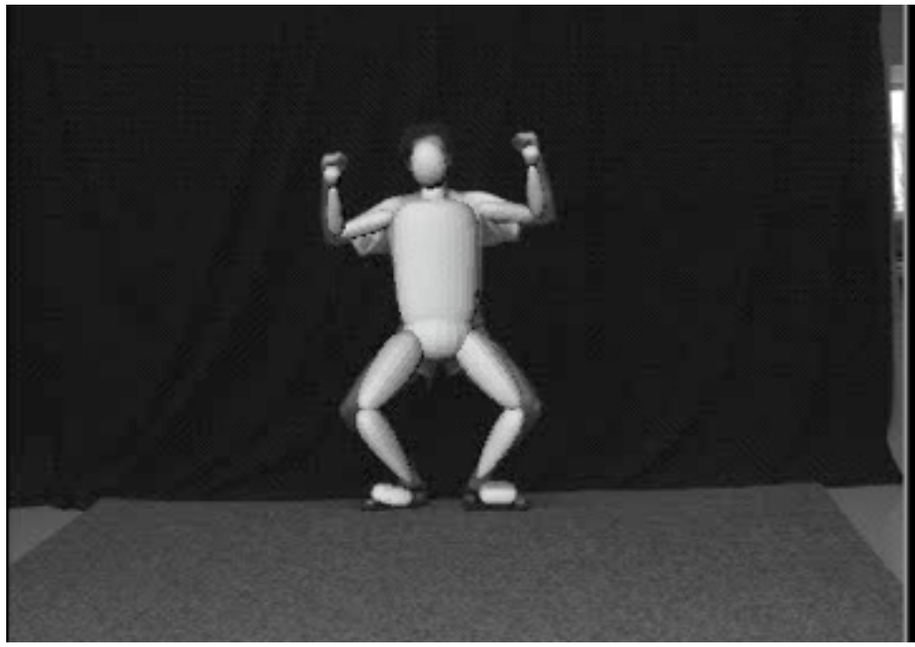
People come in all shapes and sizes, with highly variable appearance.

Challenges: Noisy and missing data

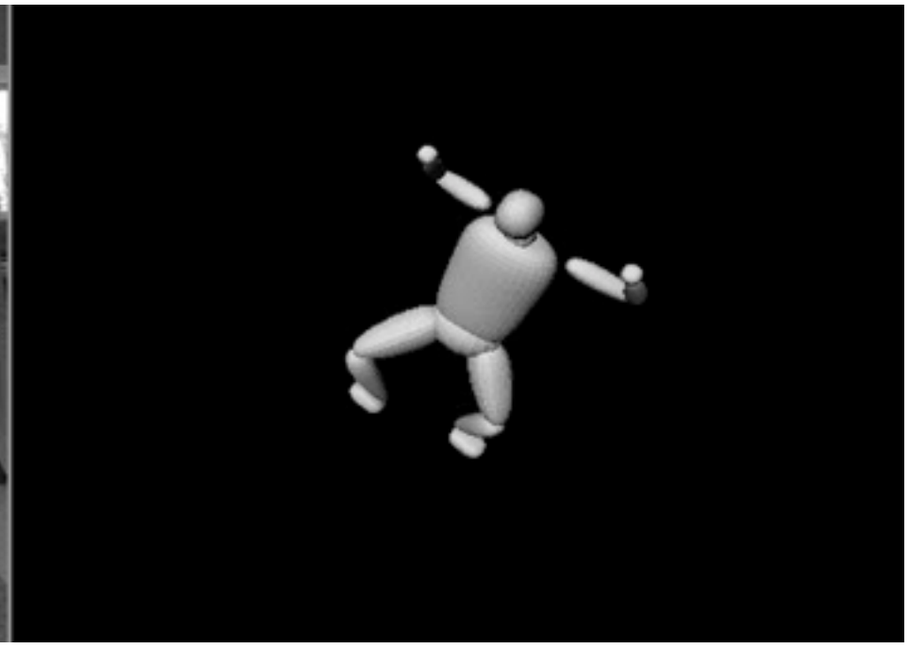
Ambiguities in pose are commonplace, e.g., due to:
background clutter, apparent similarity of parts,
occlusions, loose clothing …

Challenges: Depth and reflection ambiguities

image



3D model
(camera view)

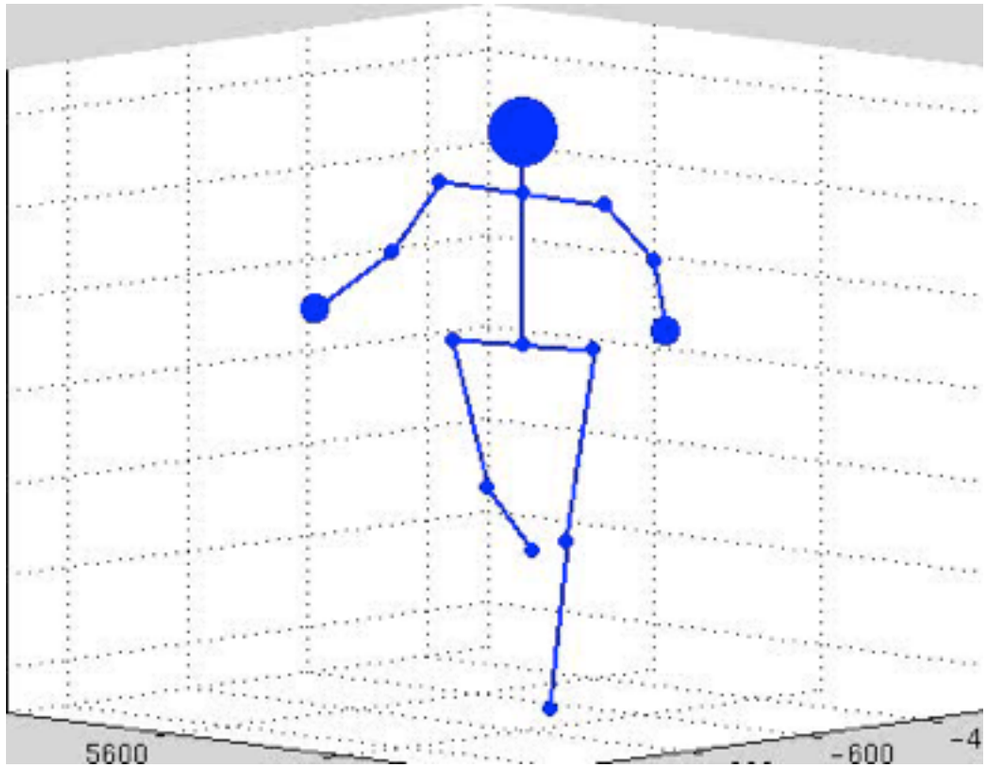


3D model
(top view)

Many 3D configurations may be consistent with a given image.

[courtesy of Cristian Sminchisescu]

Kinematic models in tracking



[Poon & Fleet 2001]

- Motion Model: damped 2nd order Markov model with Beta process noise and joint angle limits
- Observations: steerable pyramid coefficients (image edges)
- Inference: hybrid Monte Carlo particle filter

Kinematic models in tracking

[Urtasun, Fleet, Hertzmann & Fua, 2005]

- Motion Model: non-linear latent model of the pose manifold, with 2nd order Gauss-Markov model for temporal evolution
- Observations: tracked 2D patches on body (WSL tracker)
- Inference: MAP estimation (hill climbing)

Physics-based models

Physics-based motion models naturally account for:

- balance and body lean (e.g., on hills)
- sudden accelerations (e.g., collisions)
- static contact (e.g., avoiding footskate)
- variations in style due to changes in speed and mass distribution (e.g., carrying an object)
- …

Goal: use dynamics to model key physical properties of motion for 3D people tracking

Physics-based models: Humanoid Robots

Active control strategies used with humanoid robots:

- energetically inefficient (highly geared, low center of mass, …)
- tedious to design and implement
- ZMP-based stability criteria

Usually produce characteristically inhuman motion.

[Kawada Industries HRP-2, Robodex 2003]

Physics-based models: Computer animation

Learning physics-based models from mocap data using space-time optimization:

- high-dimensional models (stiffness, damping, muscle preferences, …)
- challenging optimization
- generalization

[Liu, Hertzmann & Popovic, 2006]

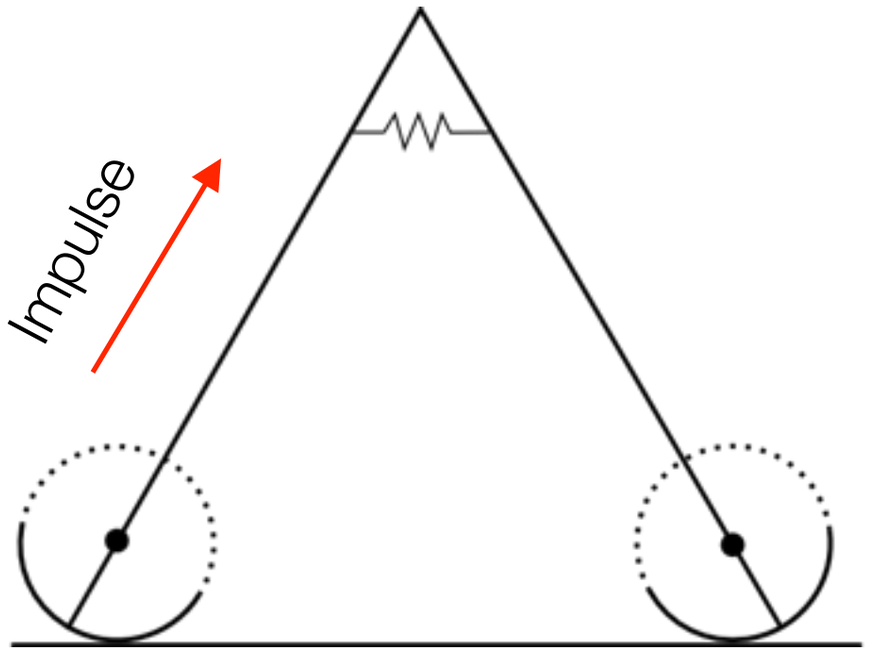
Physics-based models: Passive dynamics

Passive dynamic robotic walkers have been built which exhibit human-like gaits, with similar efficiency.

[McGeer 1990]

[Collins & Ruina 2005]

The Anthropomorphic Walker



[McGeer 1990; Kuo 2001/02]

Anthropomorphic Walker

- 2D model with rigid bodies for the torso and each leg
- forces can be added with a spring between the legs and an impulsive toe-off

Key Properties:

- when powered, exhibits a human-like preferred speed-step length relationship
- invariant to total mass and leg length (approximately)

A physics-based model of human motion

To use the anthropomorphic walker for tracking we need

- equations of motions,
- a prior distribution of spring stiffness and impulse which produce natural 2D motions,
- a 3D pose model consistent with the underlying dynamics, and
- a likelihood function to relate the 3D pose model to the image.

Dynamics of the anthropomorphic walker

Equations of motion govern the dynamics of leg orientations and ground contact, given lengths and relative masses:

- Generalized coordinates: $\mathbf{q} = (\phi_1, \phi_2)^T$
- Equations of motion:

$$\mathcal{M}\ddot{\mathbf{q}} = f_s(\kappa) + f_g + f_c$$

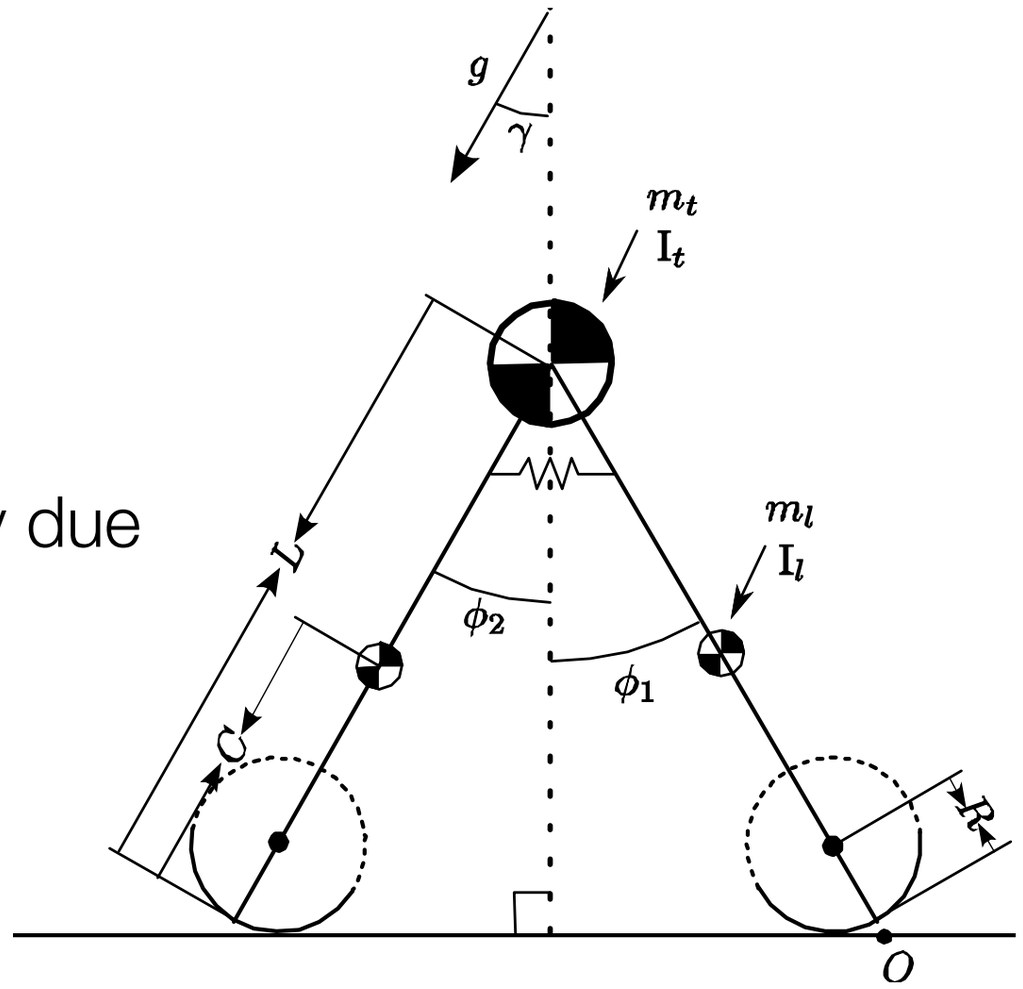
generalized mass matrix; acceleration; spring forces; gravity and constraints forces due to

- Contact: instantaneous change in velocity due to (inelastic) contact with ground:

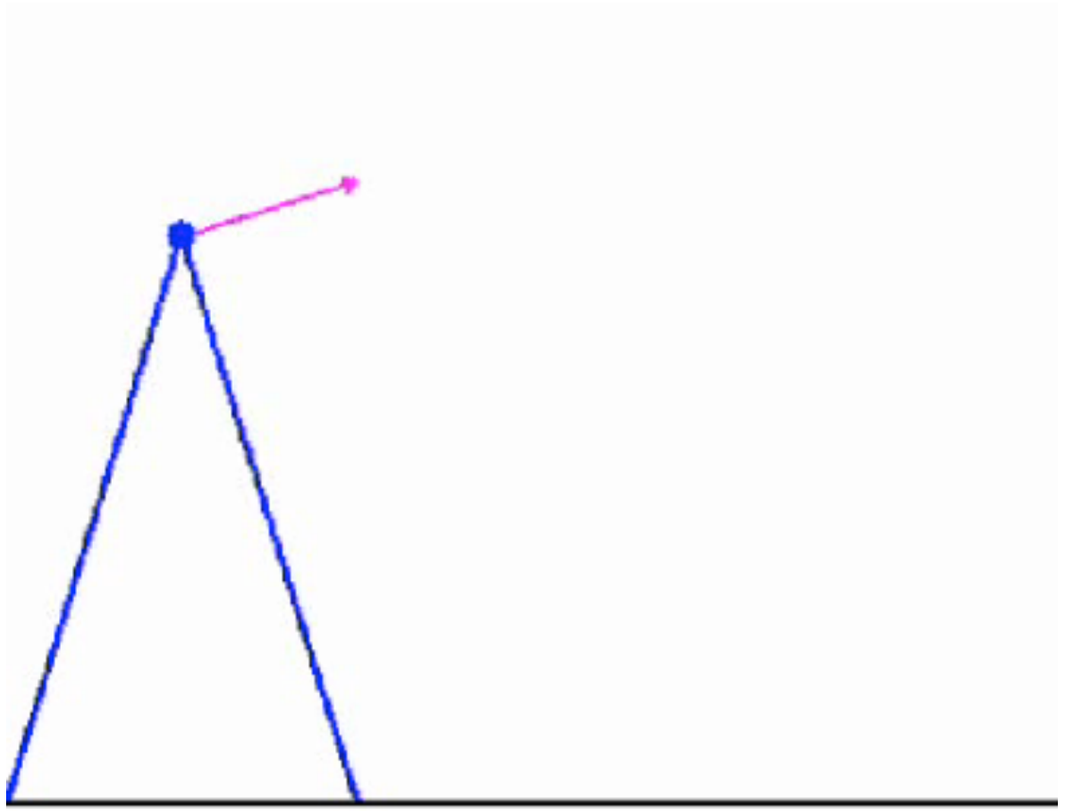
$$\mathcal{M}^+\dot{\mathbf{q}}^+ = \mathcal{M}^-\dot{\mathbf{q}}^- + \mathbf{S}(\iota)$$

post-contact velocity; pre-contact velocity; impulse

Given initial conc ntrol parameters, the equations of motion are integrated to find the time-varying pose.



Simulation



Simulation with constant stiffness and impulse

Control

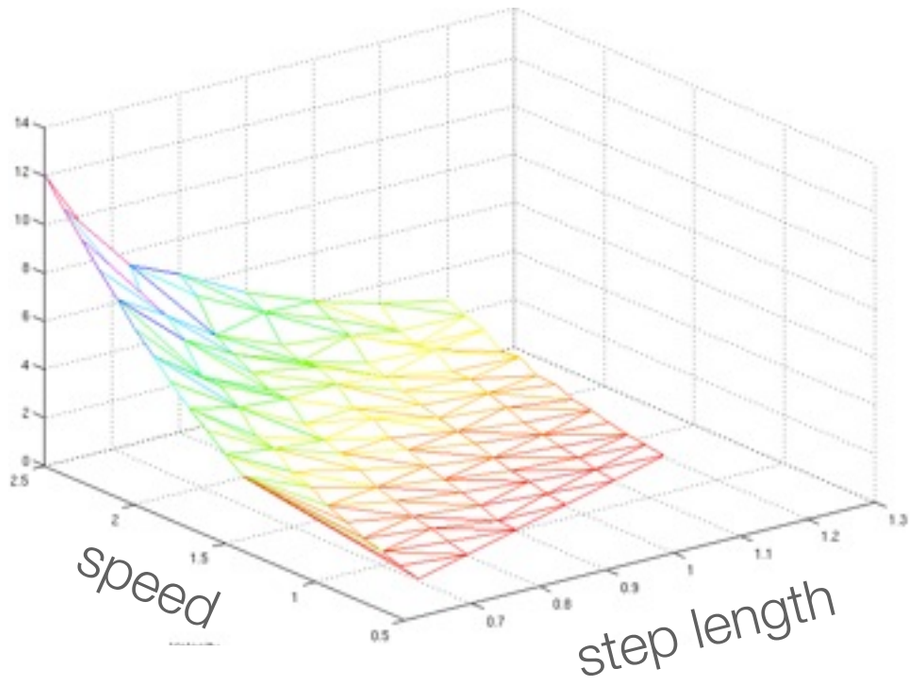
Using optimization, parameters (ι, κ) and initial state $(q, \dot{q})$ can be found which generate cyclic motions at different speeds s and step lengths ℓ

$$\min_{\iota,\kappa,q,\dot{q}} f(\iota, \kappa, q, \dot{q}; s, \ell)$$

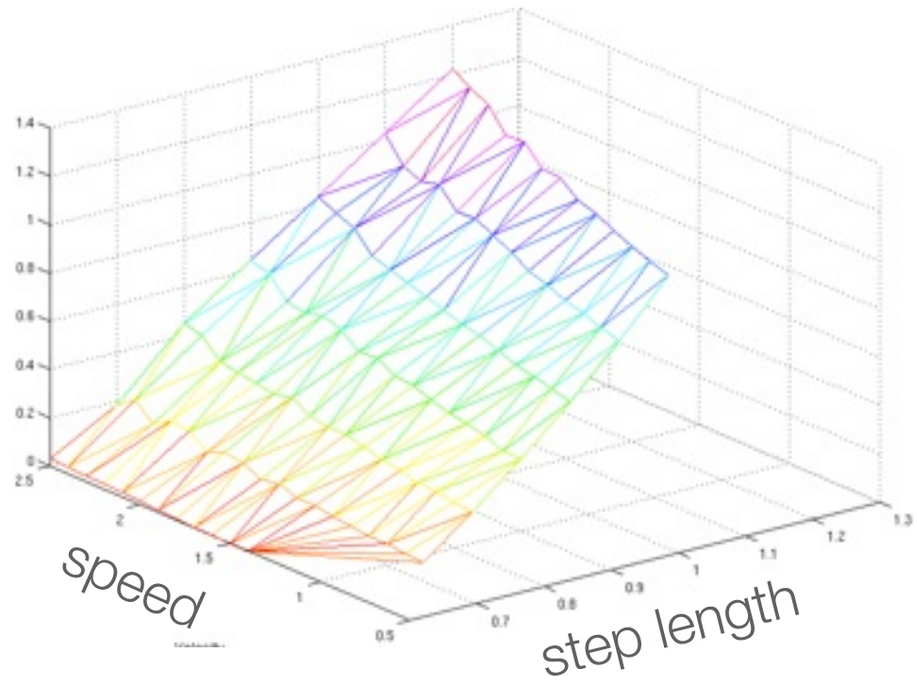
Speed: 4.0 km/hr; Step length: 0.875m

Stochastic Control

Spring Constant



Impulse Magnitude

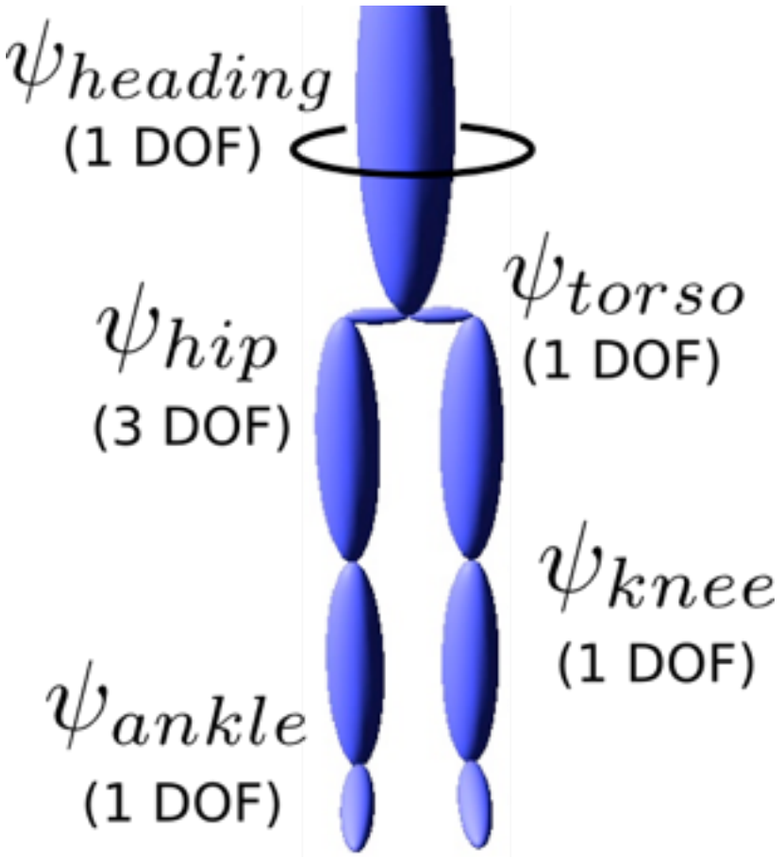


In tracking, the dynamics parameters are unknown and a simple prior, based on these optimizations, is used.

3D kinematic model

Kinematic parameters include the relative orientations of torso, thigh, knee and ankle.

- the dynamical model constrains contact of stance foot, the two thigh angles
- other parameters modeled as smooth, second-order Markov processes.
- limb lengths are assumed to be static



Bayesian people tracking

Image observations: $\mathbf{z}_{1:t} \equiv (\mathbf{z}_1, \ldots, \mathbf{z}_t)$

State: $\mathbf{s}_t = [d_t, k_t]$

Posterior distribution:

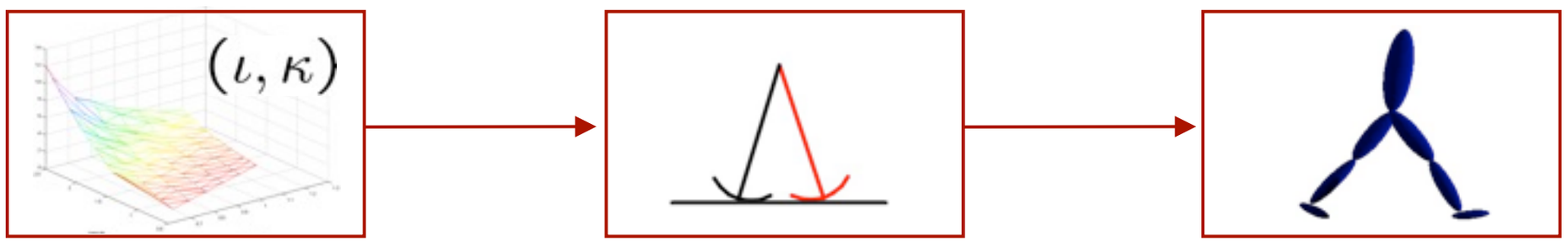
dynamics pose

$$p(\mathbf{s}_{1:t} \,|\, \mathbf{z}_{1:t}) \;\propto\; p(\mathbf{z}_t \,|\, \mathbf{s}_t)\; p(\mathbf{s}_t \,|\, \mathbf{s}_{1:t-1})\; p(\mathbf{s}_{1:t-1} \,|\, \mathbf{z}_{1:t-1})$$

likelihood transition posterior

Sequential Monte Carlo inference:

- Sampling from the transition density



sample control parameters

simulate dynamics

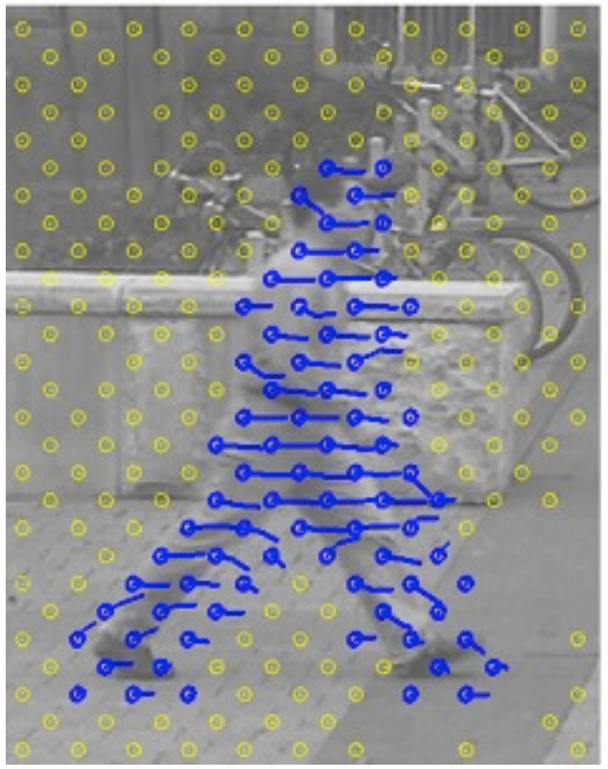
sample kinematics, given dynamics

- Resample when the effective number of samples becomes small

Measurement / Observations

Foreground Model

Background Model



Optical Flow

Foreground / Background Model

Calibration and initialization

Camera calibrated with respect to ground plane.

Gravity assumed to be normal to the ground.

Body position, pose and dynamics coarsely hand-initialized.

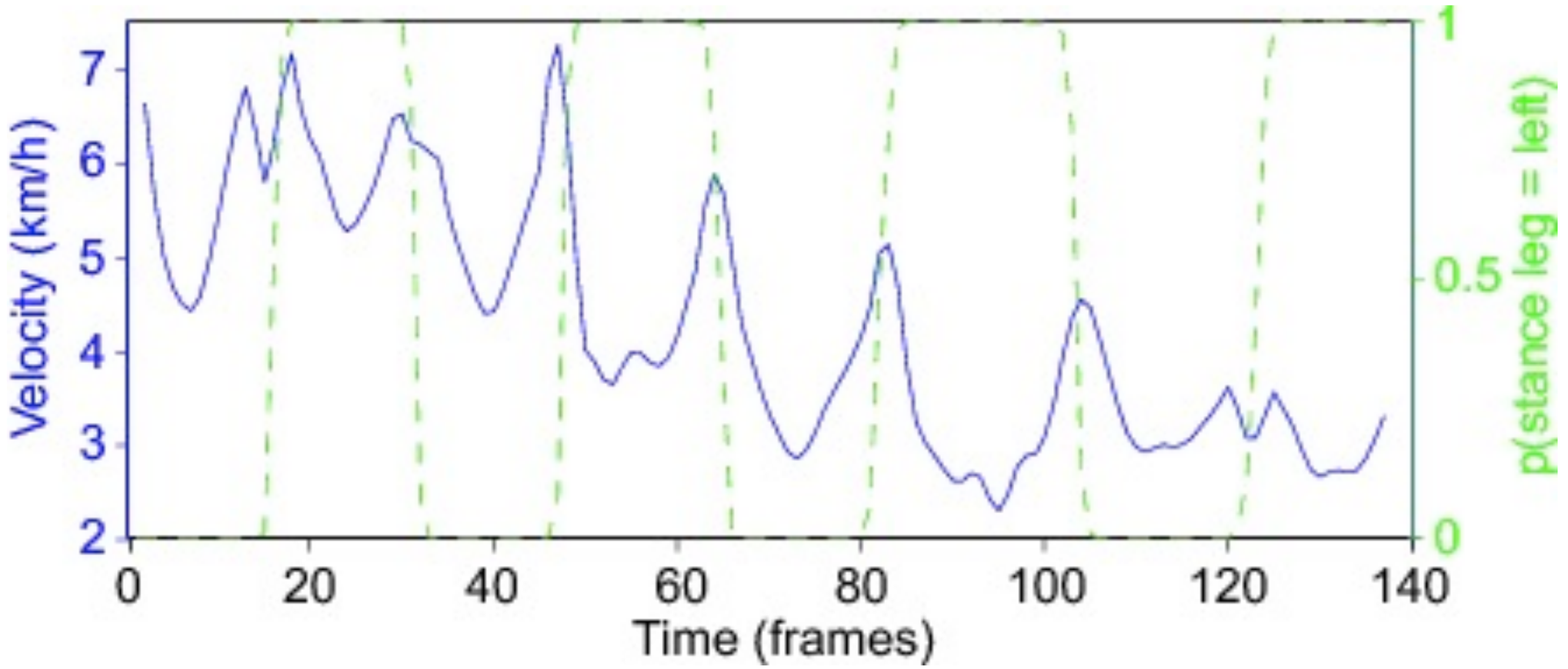
Excluding likelihood evaluations, runs at ~15 fps (5000 particles)

Experiment 1: Changing speed

Input data.

Experiment 1: Changing speed

Speed and support transfer versus time.

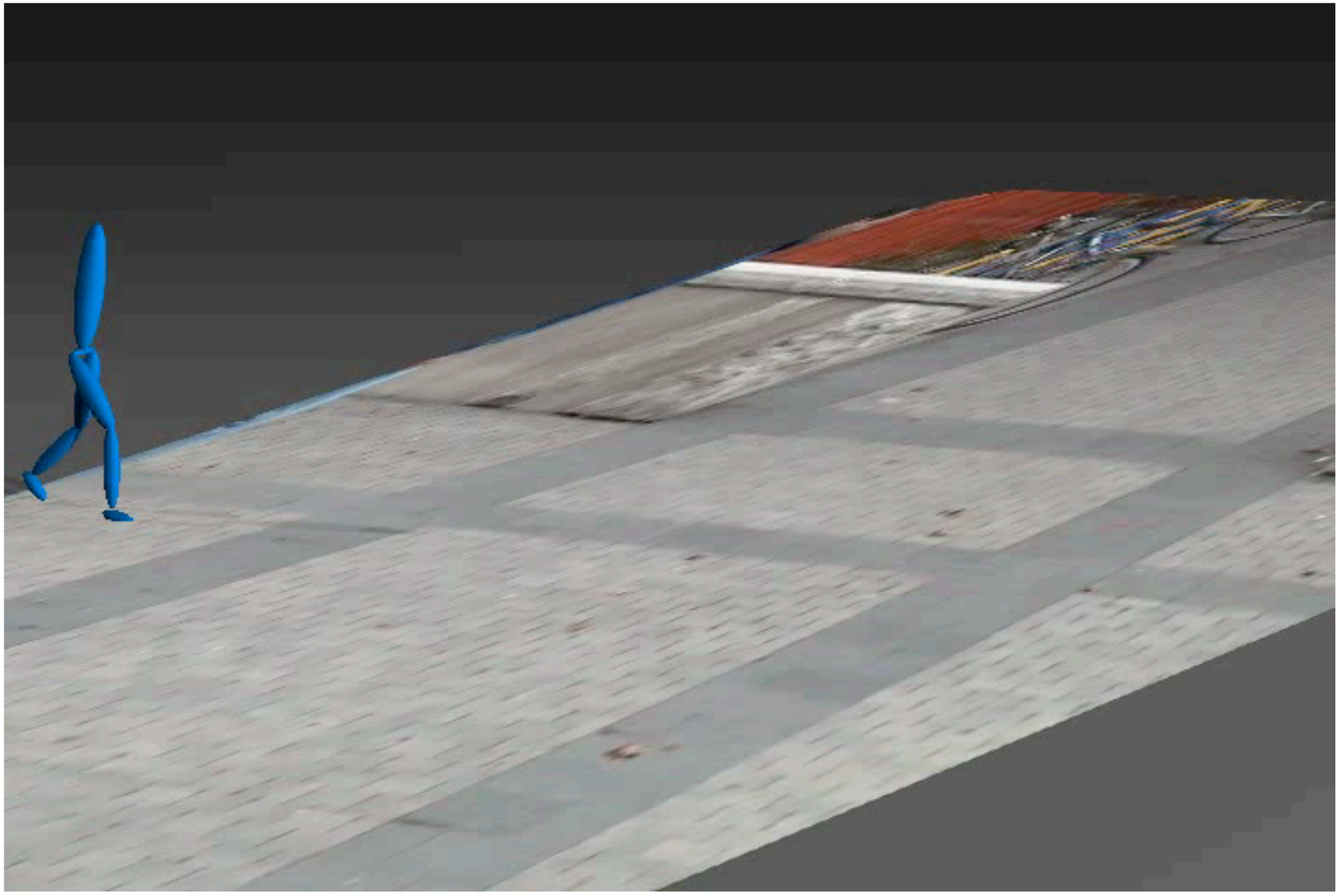


Experiment 1: Changing speed

Approximate MAP trajectory

(half speed)

Experiment 1: Changing speed



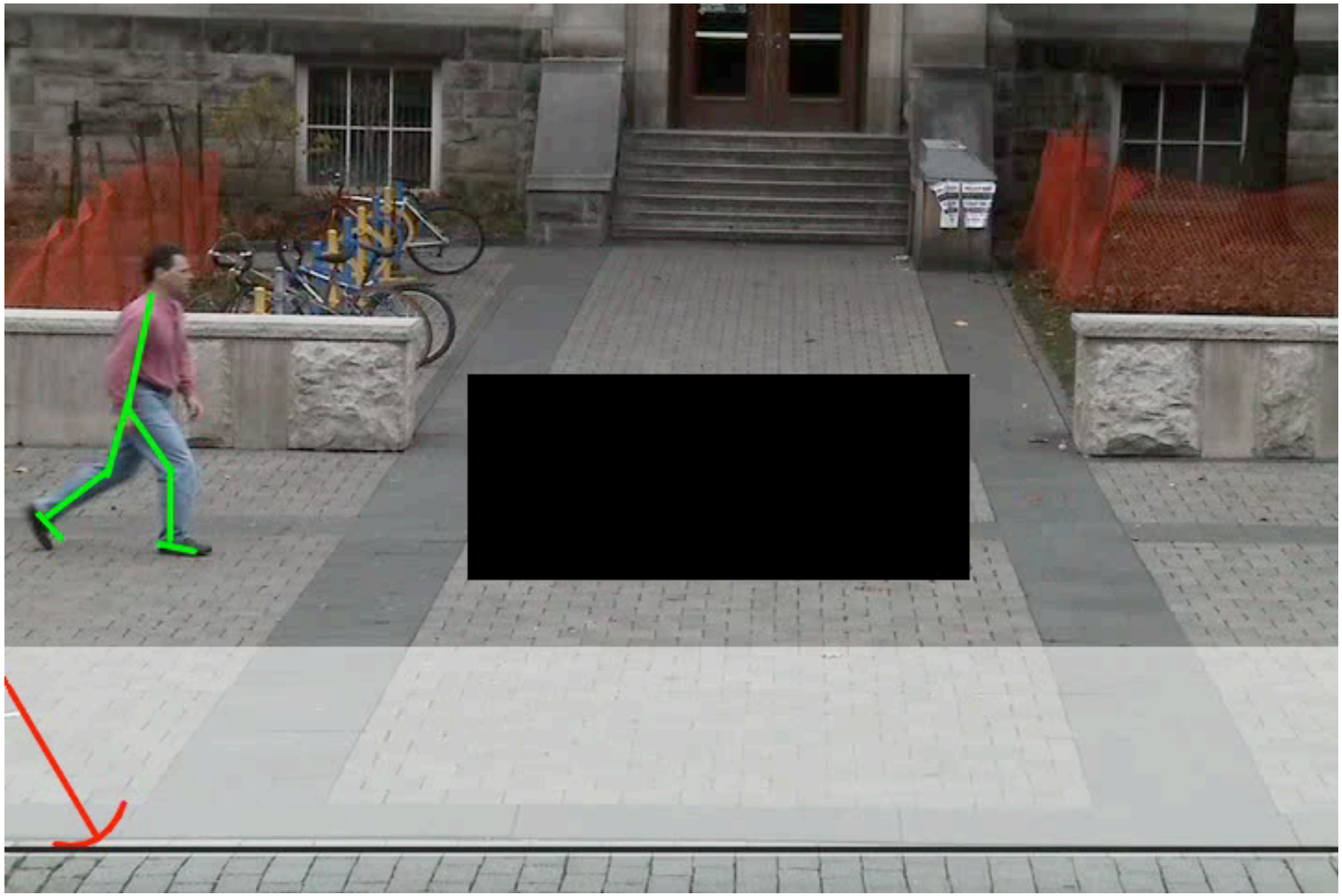
Approximate MAP trajectory in 3D

(half speed)

Experiment 2: Occlusion

Input data.

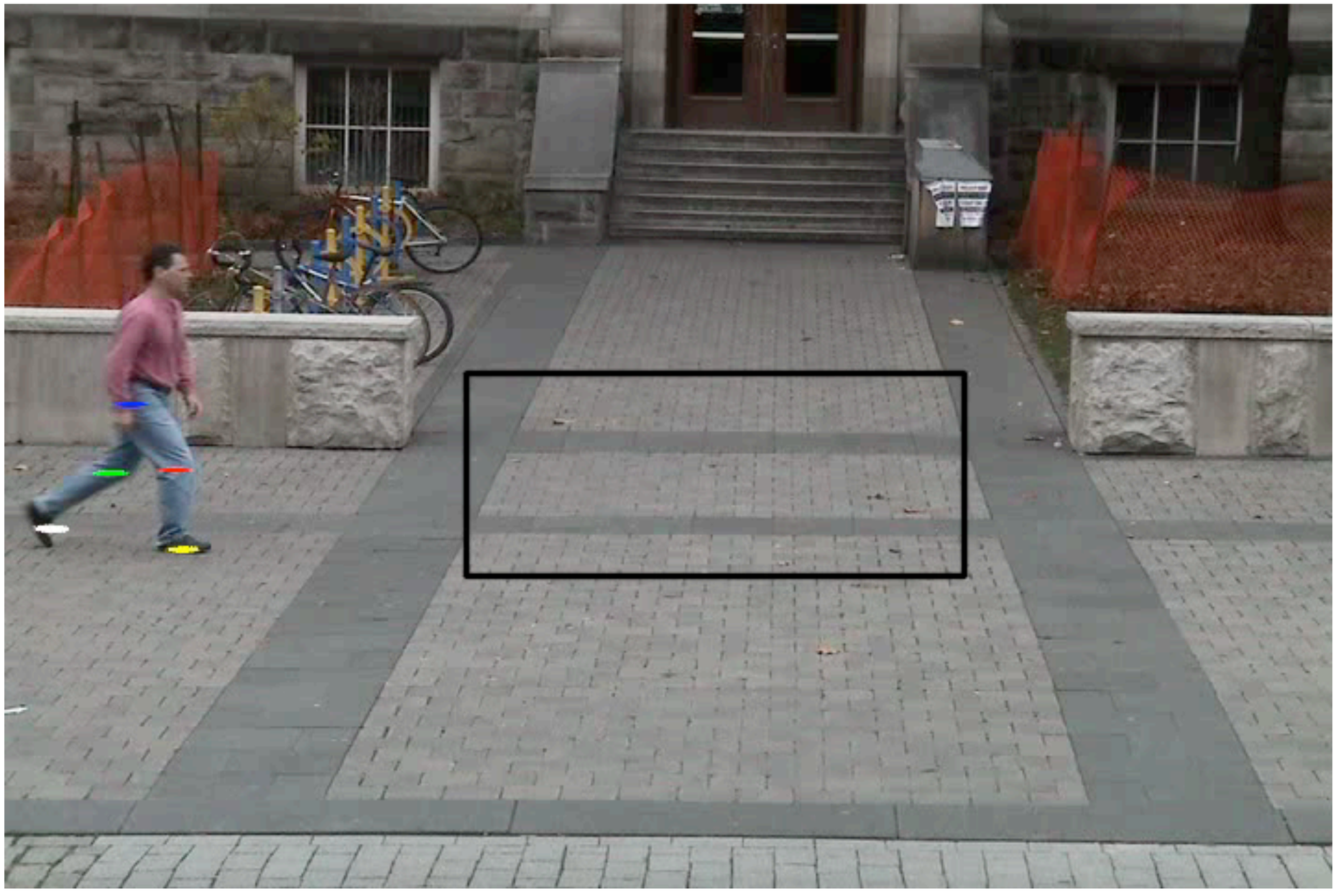
Experiment 2: Occlusion



Approximate MAP trajectory

(half speed)

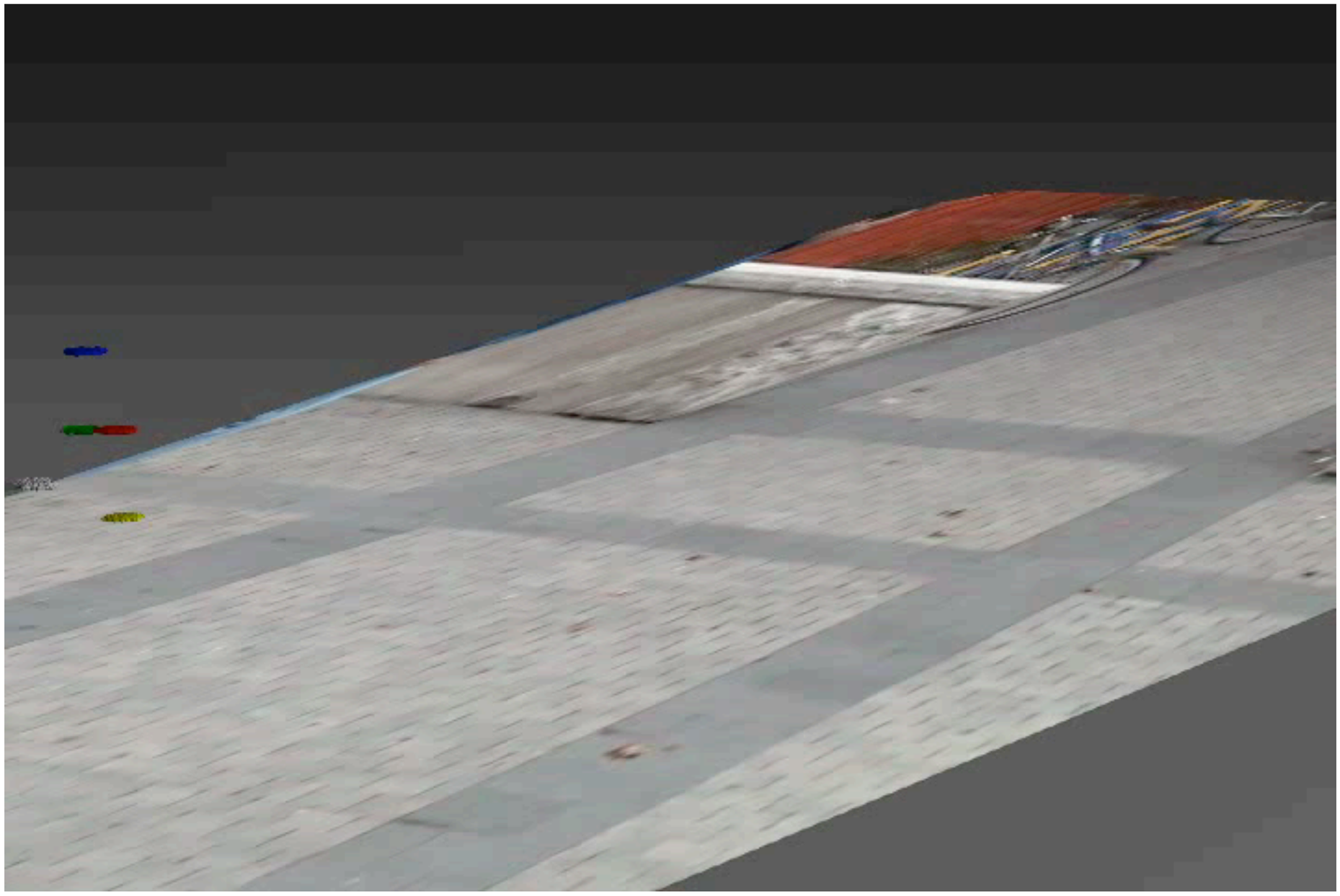
Experiment 2: Occlusion



Posterior distribution over marker locations on 3D model.

(half speed)

Experiment 2: Occlusion



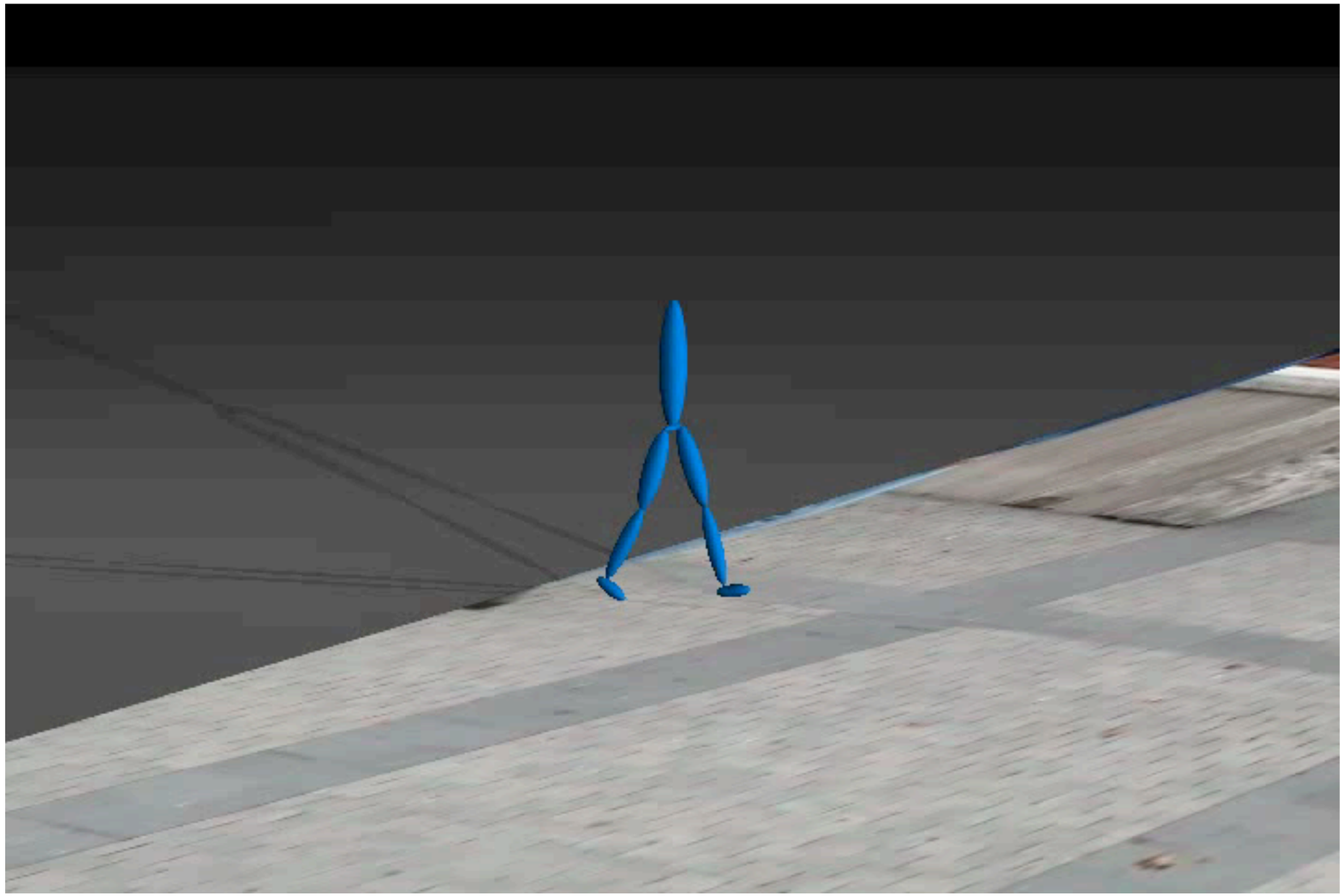
Posterior distribution over marker locations on 3D model.

(half speed)

Experiment 3: Turning with changes in speed

Approximate MAP trajectory

Experiment 3: Turning with changes in speed



Approximate MAP trajectory in 3D