Lecture 15-16

Pose Estimation – Gaussian Process

Tae-Kyun Kim

MS KINECT

http://www.youtube.com/watch?v=p2qlHoxPioM

The KINECT body pose estimation is achieved by randomised regression forest techniques.

We learn pose estimation as a regression problem, and Gaussian process as a cutting edge regression method.

We see it through the case study (slide credits to):
Semi-supervised Multi-valued Regression,

Navaratnam, Fitzgibbon, Cipolla, ICCV07,

where practical challenges addressed are
1)multi-valued regression and 2)sparsity of data.

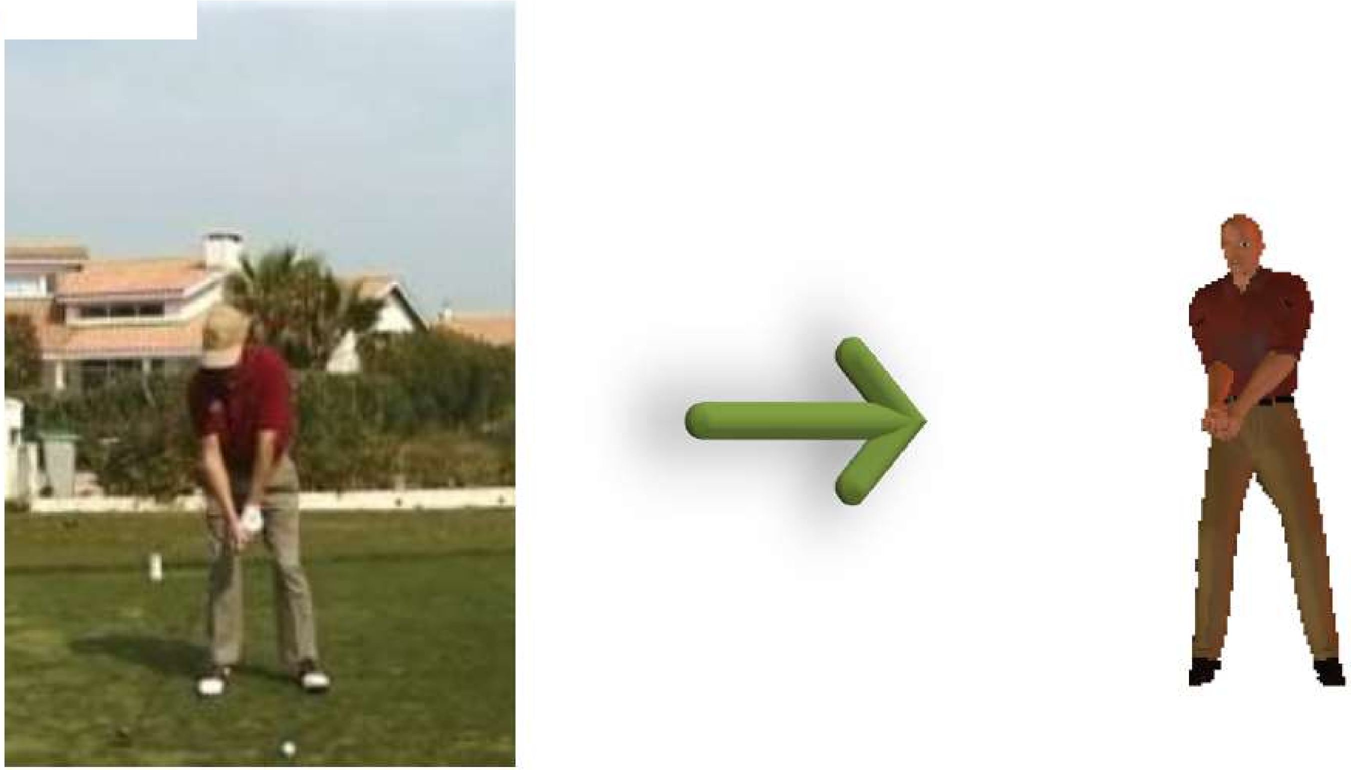
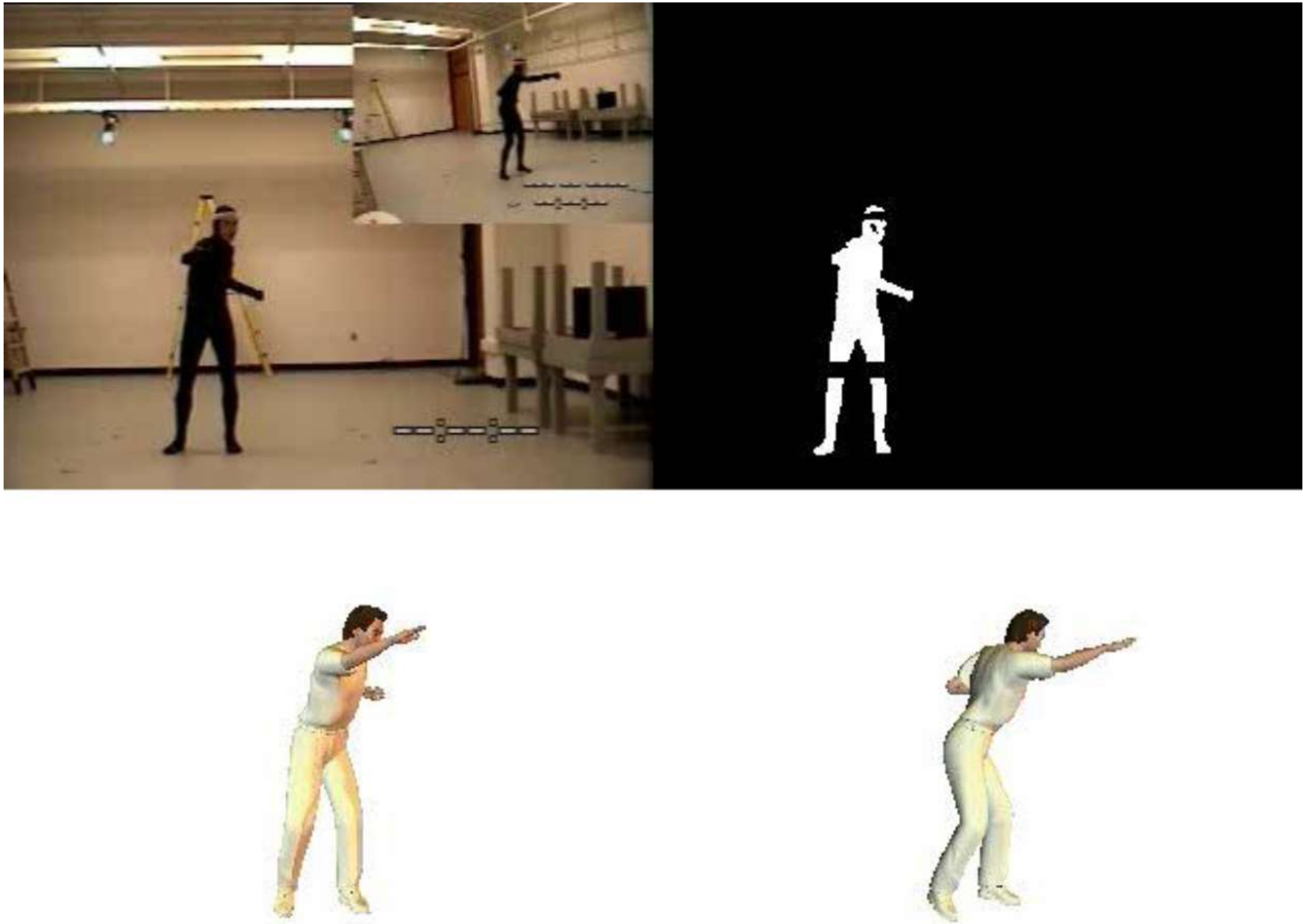


Image I Pose θ

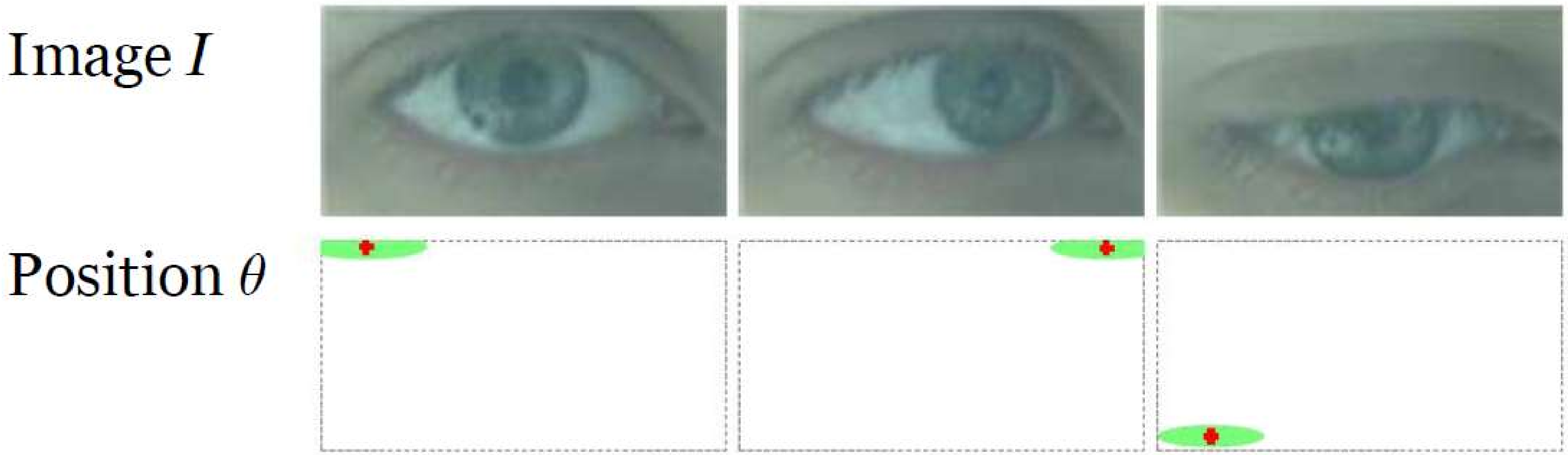
e.g. Urtasun, Fleet, Hertzmann, Fua; ICCV 2005.

A mapping function is learnt from the input image I to the pose vector θ, which is taken as a continuous variable.



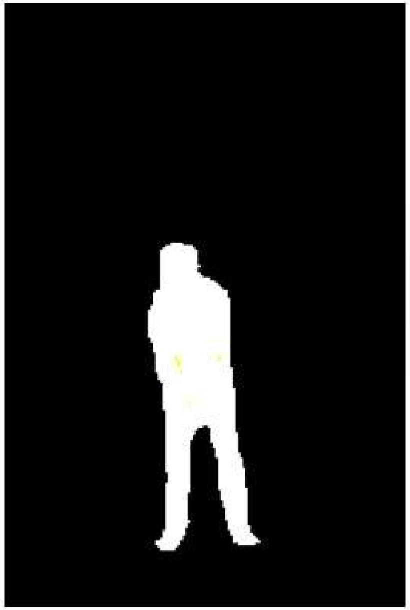
Given an image (top left) , e.g. a silhouette (top right) is obtained by background subtraction techniques (http://en.wikipedia.org/wiki/Background_subtraction). The estimated 3d pose is shown at two camera angels (bottom left and right).

We attempt to map 2D image space to 3D pose space.
There is inherent ambiguity in pose estimation (as an example in the above).



Williams, Blake, Cipolla; CVPR 2006

Eye tracking can be tackled as a regression problem, where the input is an image I and the output is a eye location.



$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

Image I

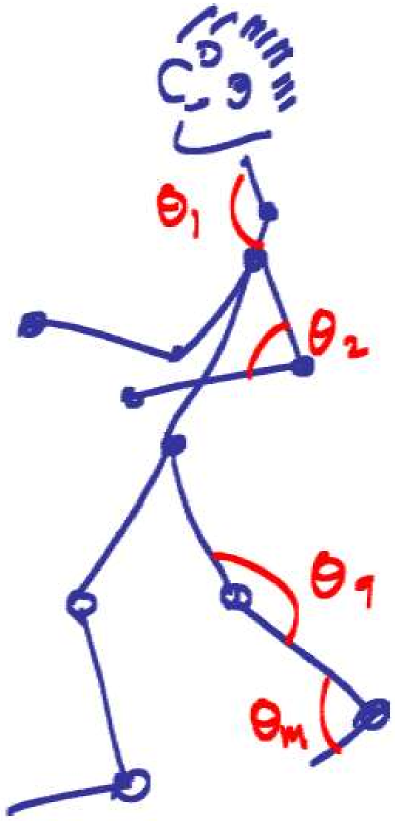
Feature vector $\mathbf{z}$

e.g. Shape contexts on silhouette, $\mathbf{z} \in \mathbb{R}^{40}$

Typical image processing steps:

Given an image, a silhouette is segmented.

A shape descriptor is applied to the silhouette to yield a finite dimensional vector. (Belongie and Malik, Matching with Shape Contexts, 2000)



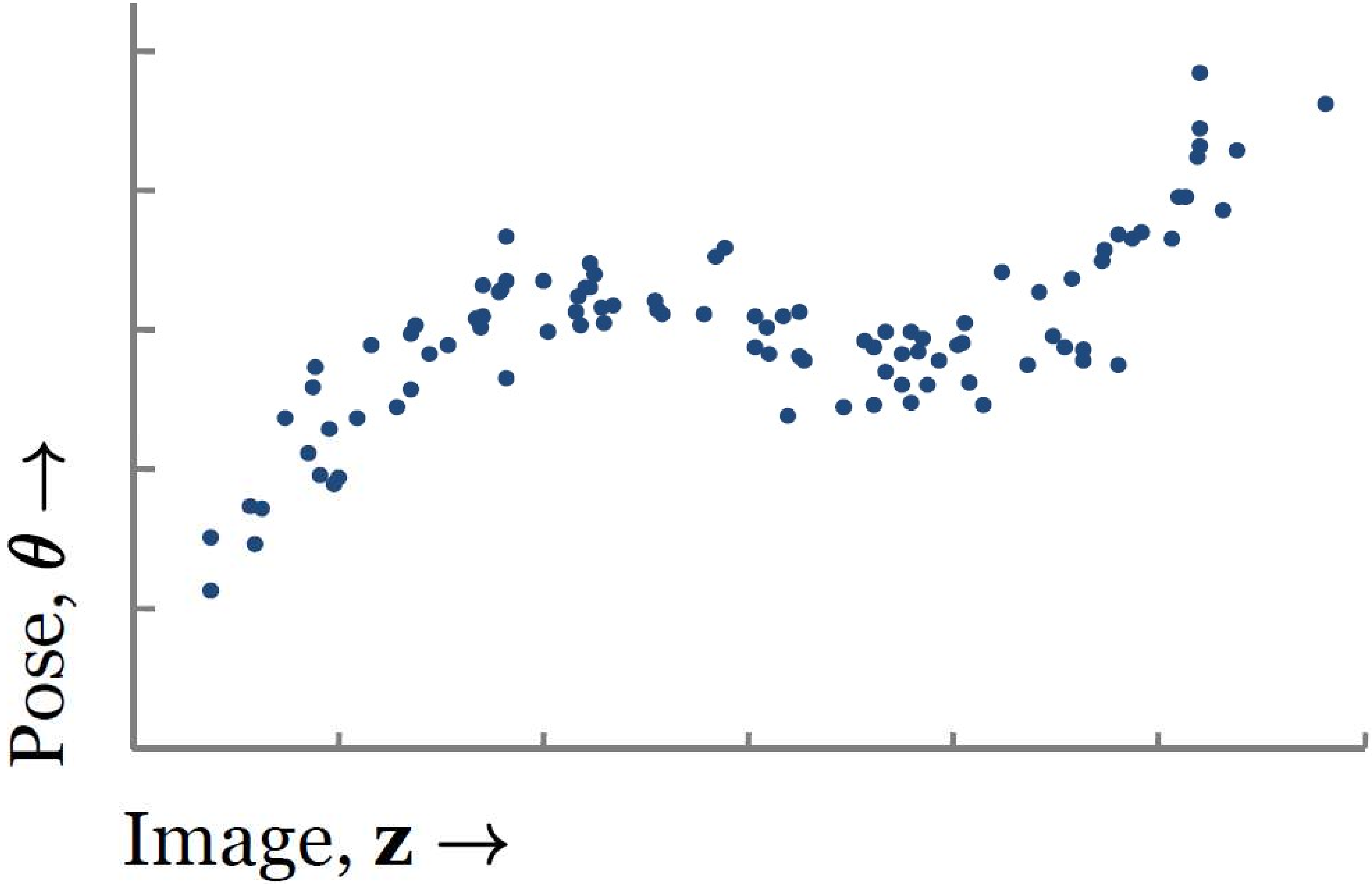
$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}$$

Pose vector $\boldsymbol{\theta}$

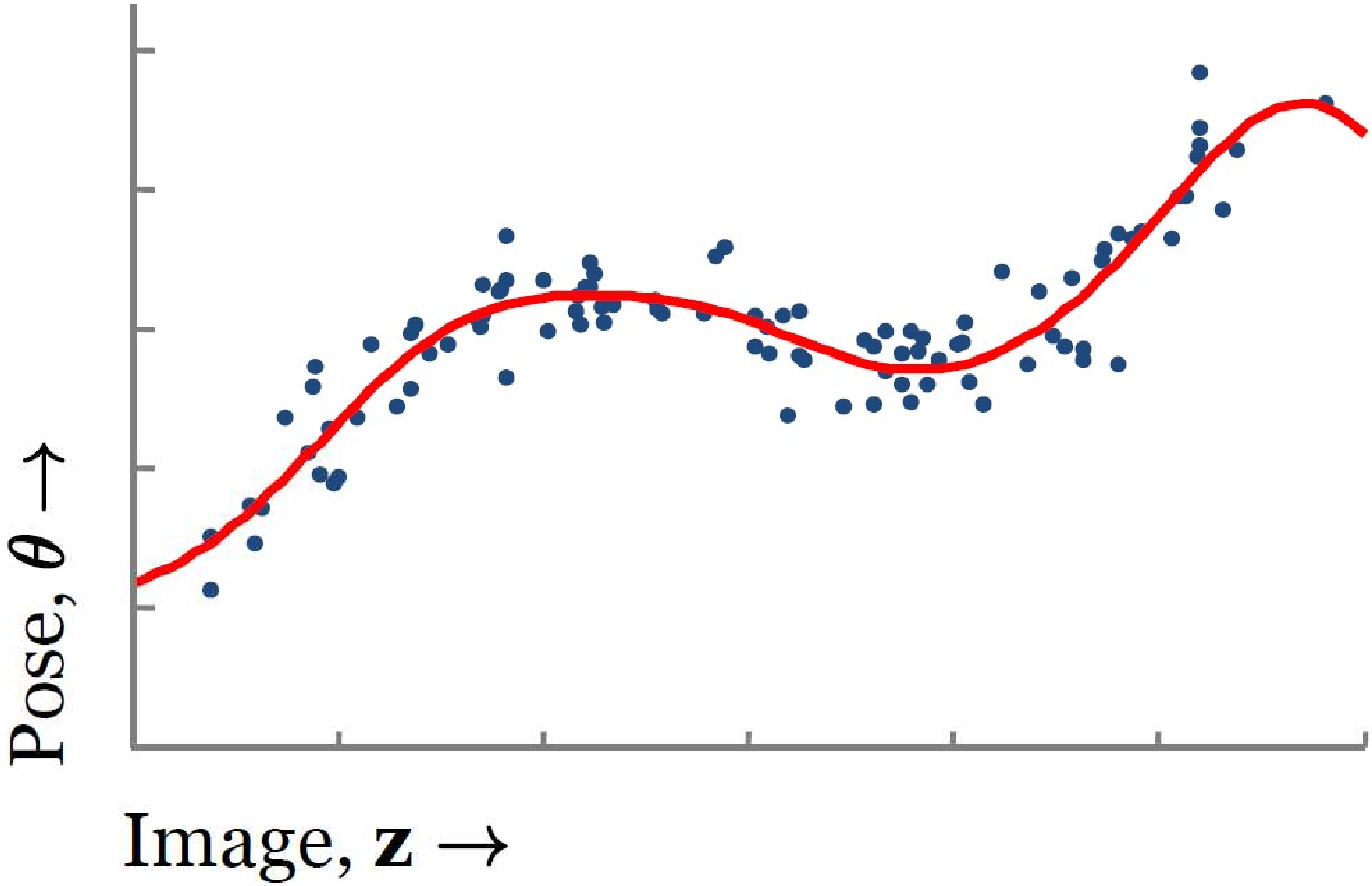
e.g. Joint angles $\boldsymbol{\theta} \in \mathbb{R}^{27}$

The output is a vector of *m* joint angles.

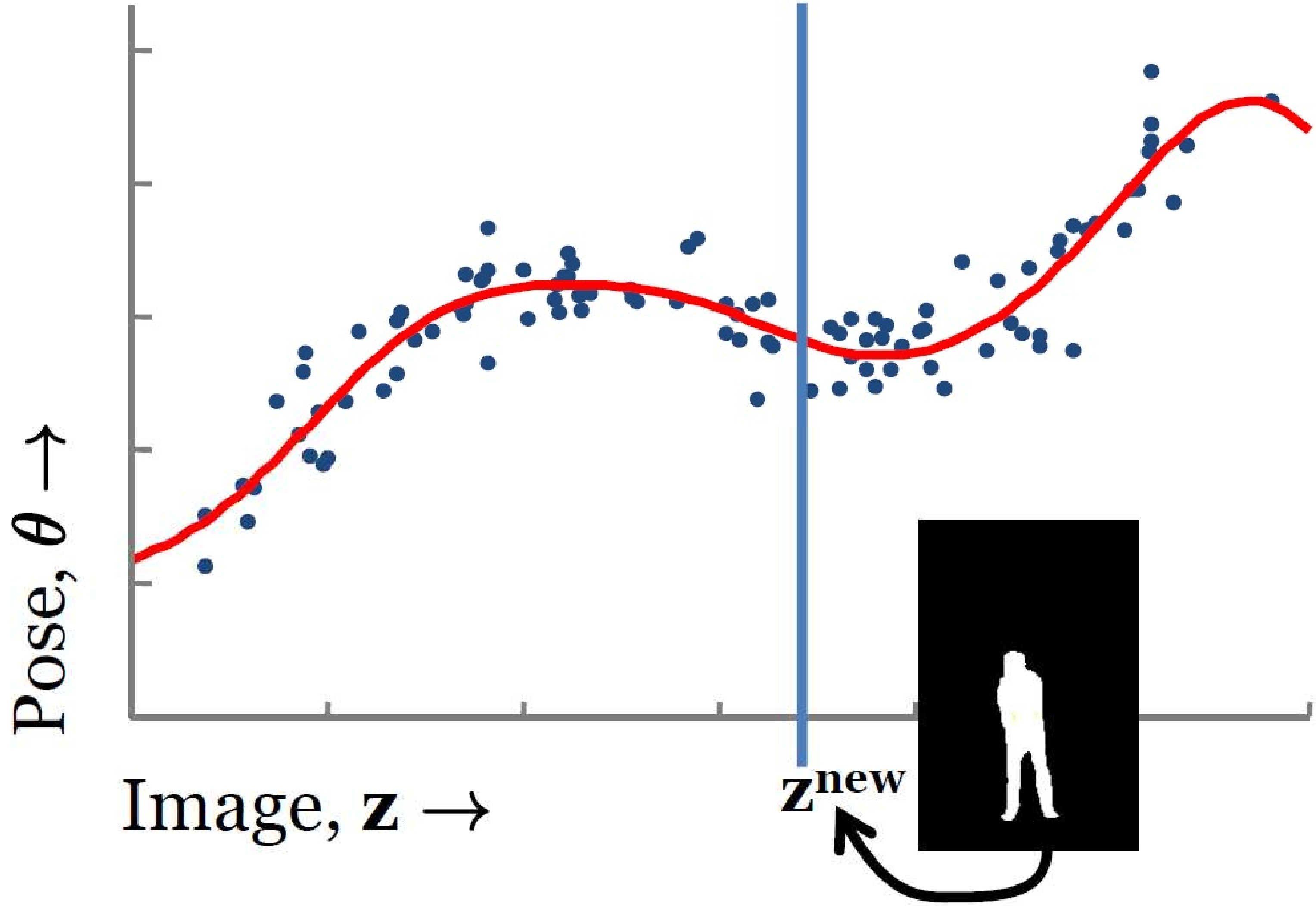
1. Obtain training samples $(\mathbf{z}_1, \boldsymbol{\theta}_1)...(\mathbf{z}_N, \boldsymbol{\theta}_N)$



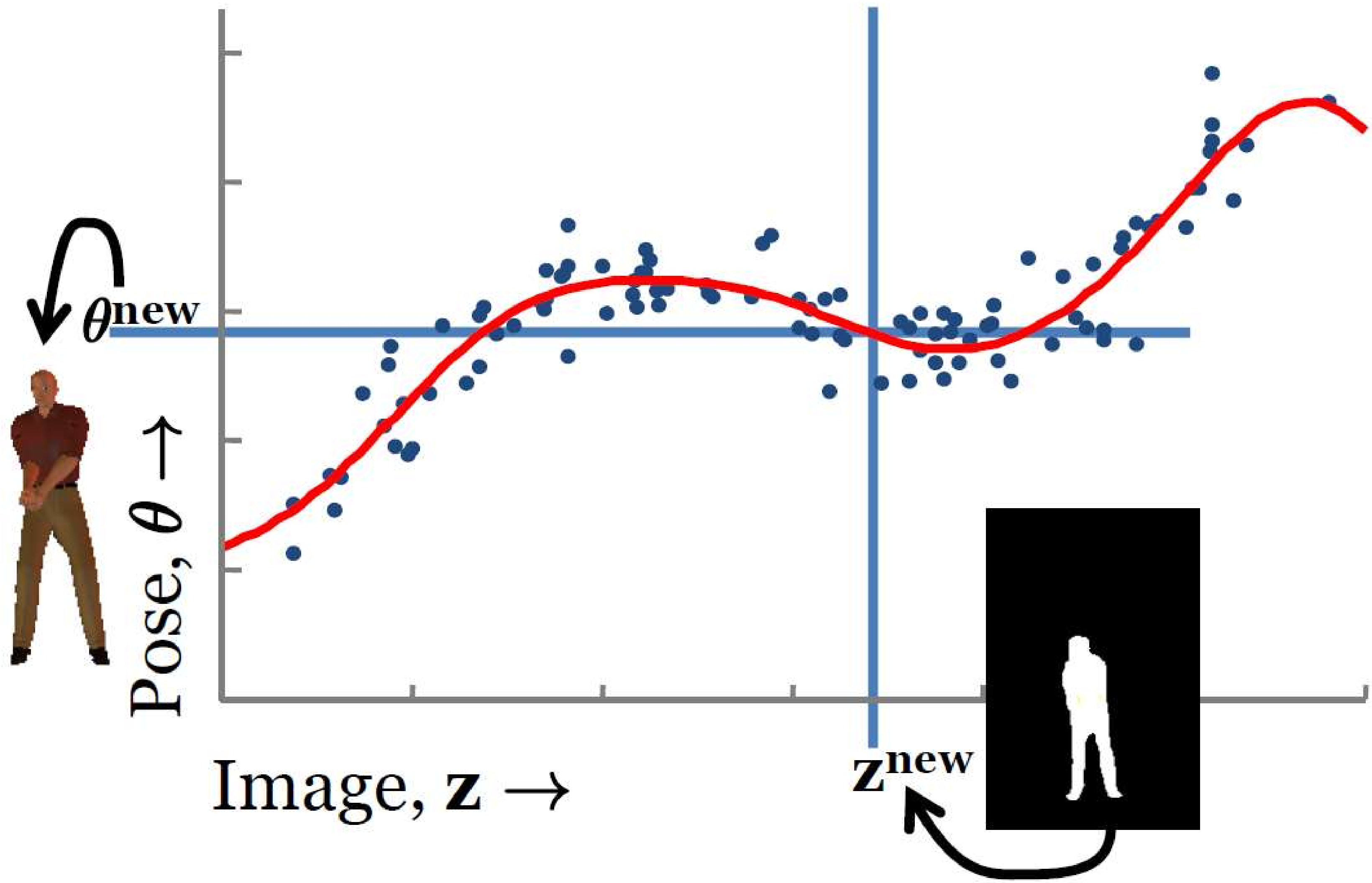
2. Training: Fit function $\boldsymbol{\theta} = f(\mathbf{z})$.



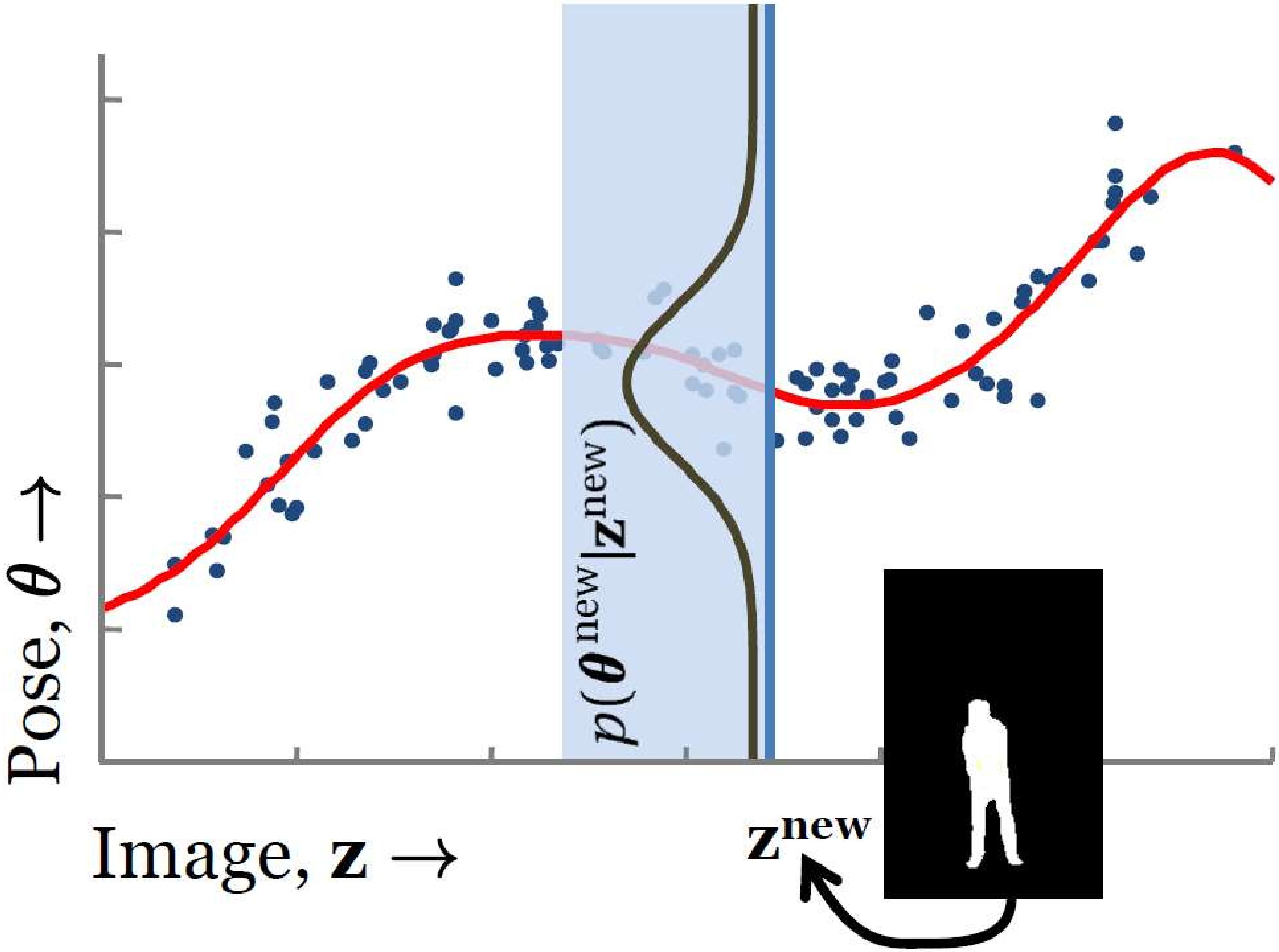
3. Given new image, $\mathbf{z}^{\text{new}}$, compute $\boldsymbol{\theta}^{\text{new}} = f(\mathbf{z}^{\text{new}})$.



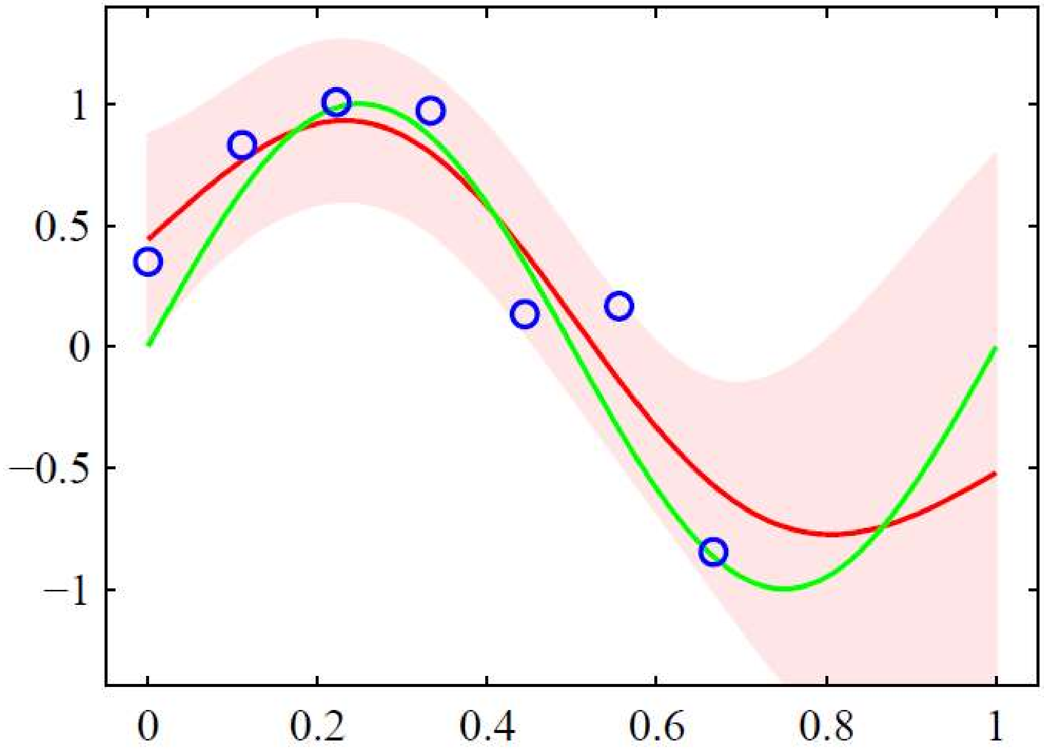
3. Given new image, $\mathbf{z}^{\text{new}}$, compute $\boldsymbol{\theta}^{\text{new}} = f(\mathbf{z}^{\text{new}})$.



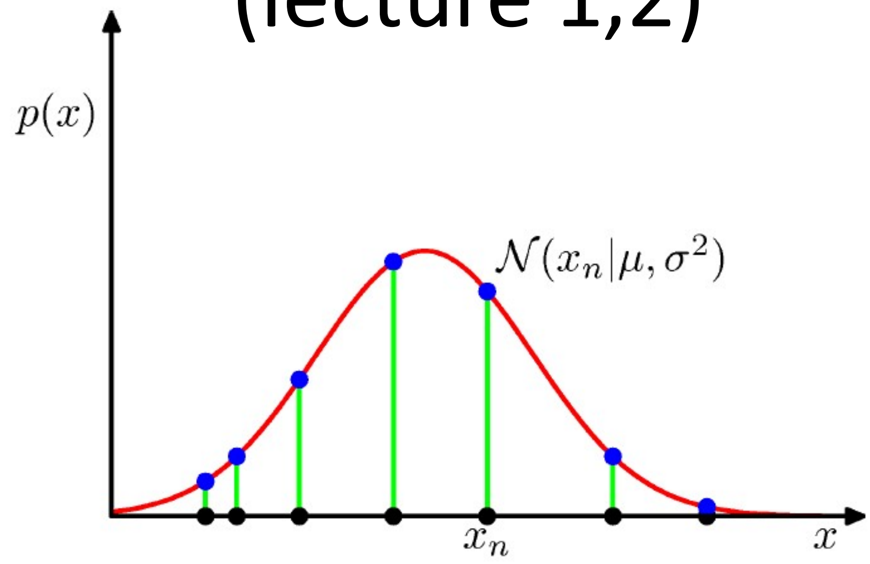
3. Or, more usefully, compute $p(\boldsymbol{\theta}^{\text{new}}|\mathbf{z}^{\text{new}})$.



Gaussian Process



Review of Gaussian density estimation (lecture 1,2)



i.i.d. (independent identical distributed)

$$\mathbf{x} = (x_1, \ldots, x_N)^T$$

We want to find the Gaussian parameters from the given data.

The problem is to find the parameters by maximising the posterior probability

$$p(\mu, \sigma^2|\mathbf{x}) = \frac{p(\mathbf{x}|\mu, \sigma^2)p(\mu, \sigma^2)}{p(\mathbf{x})}$$

Review of Gaussian density estimation (lecture 1,2)

$$p(\mu, \sigma^2|\mathbf{x}) = \frac{p(\mathbf{x}|\mu, \sigma^2)p(\mu, \sigma^2)}{p(\mathbf{x})}$$

We do not have priors on the parameters and data, thus we maximise the (log) likelihood function instead.

$$p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \sigma^2).$$

$$\ln p(\mathbf{x}|\mu, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi).$$

Maximum Likelihood (ML) vs Maximum A Posterior (MAP) solutions:

P(X|Y)

P(X|Y)P(Y)
(e.g. Gaussian Process)

Review of polynomial curve fitting (lecture 1,2)

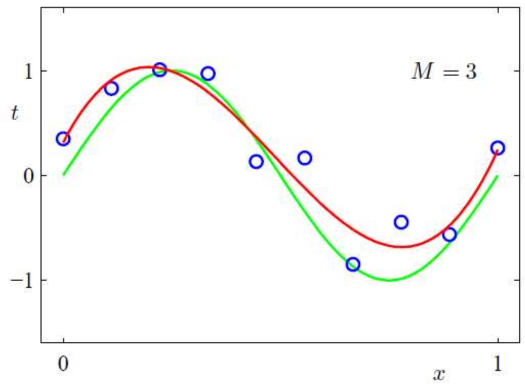
We want to fit a polynomial curve to given data pairs (x,t).

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j = \mathbf{w}^T \mathbf{x}$$

where $\mathbf{w}=\begin{bmatrix} w_0 \\ w_1 \\ \vdots \end{bmatrix}$ $\mathbf{x}=\begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \end{bmatrix}$

The objective ftn to minimise is $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}||\mathbf{w}||^2$

where $||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w} = w_0^2 + w_1^2 \ldots + w_M^2$



Gaussian Processes

• *Gaussian Processes:* extends the role of kernels to probabilistic discriminative models. We shall see how kernels arise in a Bayesian setting.

• We consider linear regression example and derive the predictive distribution by working in terms of distributions over functions $y(\mathbf{x}, \mathbf{w})$.

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

Consider a prior distribution over $\mathbf{w}$ by an isotropic Gaussian of the form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

where the hyperparameter α is the precision (inverse variance).

The probability distribution over $\mathbf{w}$ thus induces

→ a probability distribution over functions $y(\mathbf{x})$.

• From given a data set $\mathbf{x}_1, ..., \mathbf{x}_N$, we want to know the joint distribution of the function values $y(\mathbf{x}_1), ..., y(\mathbf{x}_N)$. We denote the vector $\mathbf{y}$ that has elements $y_n = y(\mathbf{x}_n)$. Then we have

$$\mathbf{y} = \Phi\mathbf{w}$$

where Φ is the matrix with elements $\Phi_{nk} = \phi_k(\mathbf{x}_n)$.

Note $\mathbf{y}$ is a linear combination of Gaussian distributed variables given by the elements of $\mathbf{w}$ and hence is itself Gaussian. It has its mean and covariance as follows.

$$\mathbb{E}[\mathbf{y}] = \Phi\mathbb{E}[\mathbf{w}] = \mathbf{0}$$

$$\mathbf{y} \sim N(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

$$\text{cov}[\mathbf{y}] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] = \Phi\mathbb{E}[\mathbf{w}\mathbf{w}^T]\Phi^T = \frac{1}{\alpha}\Phi\Phi^T = \mathbf{K}$$

where $\mathbf{K}$ is the Gram matrix that has

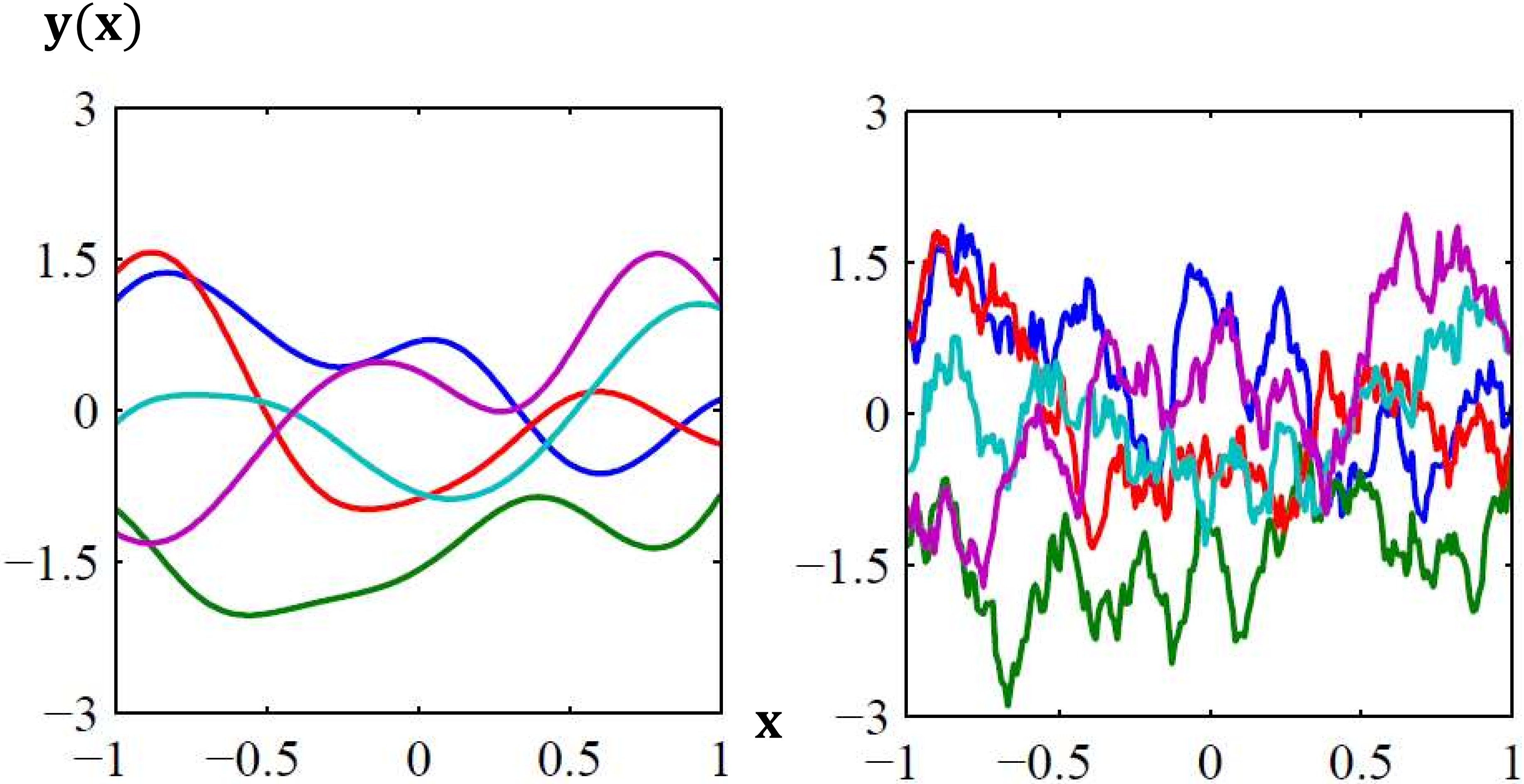
$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m).$$

- The figure below shows samples of functions drawn from Gaussian processes for the Gaussian kernel

$$k(x, x') = \exp(-||x - x'||^2/2\sigma^2)$$

(left) and the exponential kernel (right) given by

$$k(x, x') = \exp(-\theta|x - x'|)$$


$\mathbf{y}(\mathbf{x})$
3
1.5
0
−1.5
−3
−1
−0.5
0
0.5
1
$\mathbf{x}$
3
1.5
0
−1.5
−3
−1
−0.5
0
0.5
1

Gaussian Processes for Regression

- We consider the noise on the observed target values by

$$t_n = y_n + \epsilon_n$$

where $y_n = y(\mathbf{x}_n)$, and ϵ_n is a random noise variable.

We consider noise processes that have a Gaussian distribution, so that

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$$

where β is a hyperparameter representing the precision of the noise.

• The noise is independent for each data point. The joint distribution of the target values $\mathbf{t} = (t_1, ..., t_N)^T$ conditioned on $\mathbf{y} = (y_1, ..., y_N)^T$ is an isotropic Gaussian of the form

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N)$$

From the Gaussian process, the marginal distribution $p(\mathbf{y})$ is a Gaussian whose mean is zero and covariance is a Gram matrix $\mathbf{K}$ i.e.

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}).$$

• To find the marginal distribution $p(\mathbf{t})$ for given input values, we make use of the following (proof skipped).

Given

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \Lambda^{-1})$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}),$$

the marginal distribution becomes

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mu + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\Lambda^{-1}\mathbf{A}^T)$$

Thus, we have

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C})$$

where

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}.$$

Their covariances simply add, as the two Gaussian sources of randomness i.e. $y(\mathbf{x}), \epsilon$ are independent.

- One widely used kernel function for Gaussian process regression is

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\left\{-\frac{\theta_1}{2}||\mathbf{x}_n - \mathbf{x}_m||^2\right\} + \theta_2 + \theta_3 \mathbf{x}_n^T \mathbf{x}_m.$$

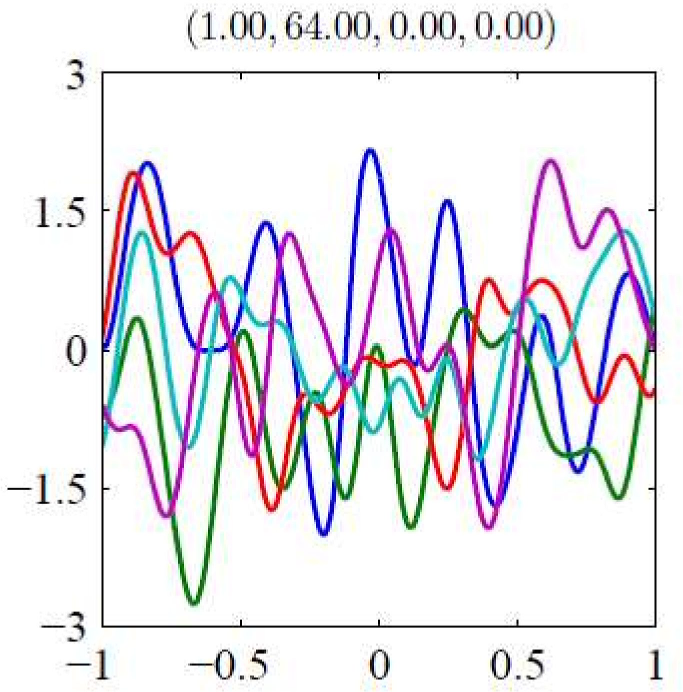


Figure 1: Samples from a Gaussian process prior defined by the kernel function with $(\theta_0, \theta_1, \theta_2, \theta3)$.

$t = y(x) + \epsilon$

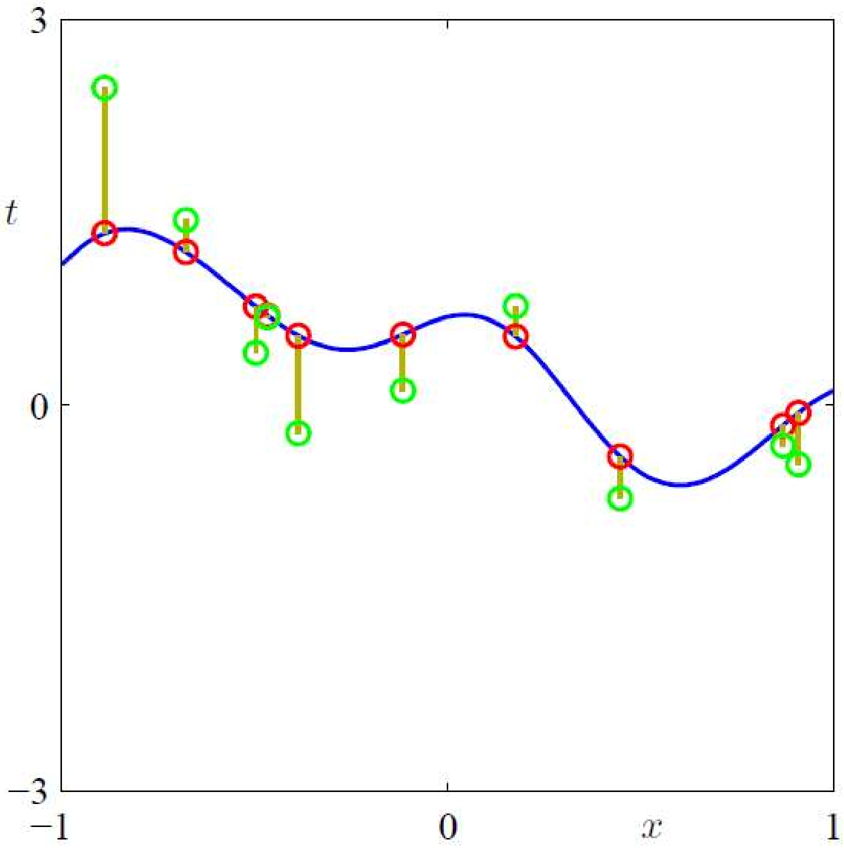


Figure 2: The blue curve shows a sample function from Gaussian process prior over functions, and the red points show the values of y_n. The greens are the values of t_n by adding Gaussian noise.

- Our goal is to predict the target value t_{N+1} for a new input $\mathbf{x}_{N+1}$, given a set of training data $\mathbf{x}_1, ..., \mathbf{x}_N$ and $\mathbf{t}_N = (t_1, ..., t_N)^T$.

This requires the evaluation of the predictive distribution $p(t_{N+1}|\mathbf{t}_N)$, where we omit the data vectors for notational simplicity.

- The joint distribution is given by

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1})$$

where $\mathbf{C}_{N+1}$ is an $(N+1) \times (N+1)$ covariance matrix. We partition the covariance matrix as

$$\mathbf{C}_{N+1} = \begin{pmatrix} \overset{N \times N}{\mathbf{C}_N} & \overset{N \times 1}{\mathbf{k}} \\ \mathbf{k}^T & c \end{pmatrix}$$

where the vector $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ for $n = 1, ..., N$, and the scalar $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$.

- We use the followings to get the conditional distribution $p(t_{N+1}|\mathbf{t})$.

Given

$$\mathbf{x} = \begin{bmatrix} x_a \\ x_b \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$$

, we have

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b)$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

Therefore, we have $p(t_{N+1}|\mathbf{t})$ a Gaussian distribution with mean and covariance given by

$$m(\mathbf{x}_{N+1}) = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}$$

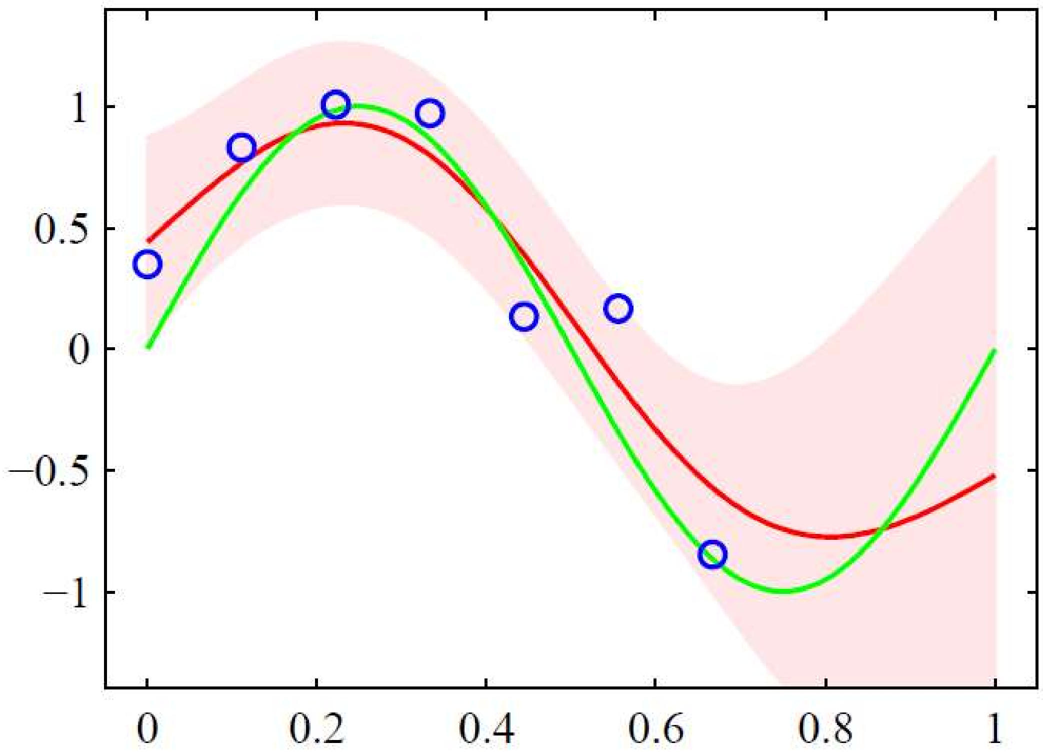
$$\sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k}.$$

• The mean of the predictive distribution can be written as a function of $\mathbf{x}_{N+1}$ as

$$m(\mathbf{x}_{N+1}) = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x}_{N+1})$$

where a_n is the n-th component of $\mathbf{C}_N^{-1}\mathbf{t}$.

These are the key results of Gaussian process regression. An example of Gaussian process regression is shown below.



Gaussian Process Matlab Toolbox

http://www.lce.hut.fi/research/mm/gpstuff/install.shtml

(try demo_regression_robust.m, demo_regression1.m)

Learning Hyperparameters

• The predictions of a Gaussian process depends on the choice of covariance i.e. kernel function. Techniques for learning the hyperparameters are based on the evaluation of the likelihood function $p(\mathbf{t}|\theta)$.

• The log likelihood function is given using the standard from of a multivariate Gaussian distribution as

$$\ln p(\mathbf{t}|\theta) = -\frac{1}{2}\ln|\mathbf{C}_N| - \frac{1}{2}\mathbf{t}^T\mathbf{C}_N^{-1}\mathbf{t} - \frac{N}{2}\ln(2\pi).$$

• Maximisation of the log likelihood can be done using the gradient-based optimisation. We can take the derivative w.r.t. θ_i as

$$\frac{\partial}{\partial \theta_i} \ln p(\mathbf{t}|\theta) = -\frac{1}{2}\mathrm{Tr}\left(\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{t}^T\mathbf{C}_N^{-1}\frac{\partial \mathbf{C}_N}{\partial \theta_i}\mathbf{C}_N^{-1}\mathbf{t}$$

which is obtained by $\frac{\partial}{\partial x}(\mathbf{A}^{-1}) = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1}$ and $\frac{\partial}{\partial x}\ln|\mathbf{A}| = \mathrm{Tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right)$ (proofs skipped).

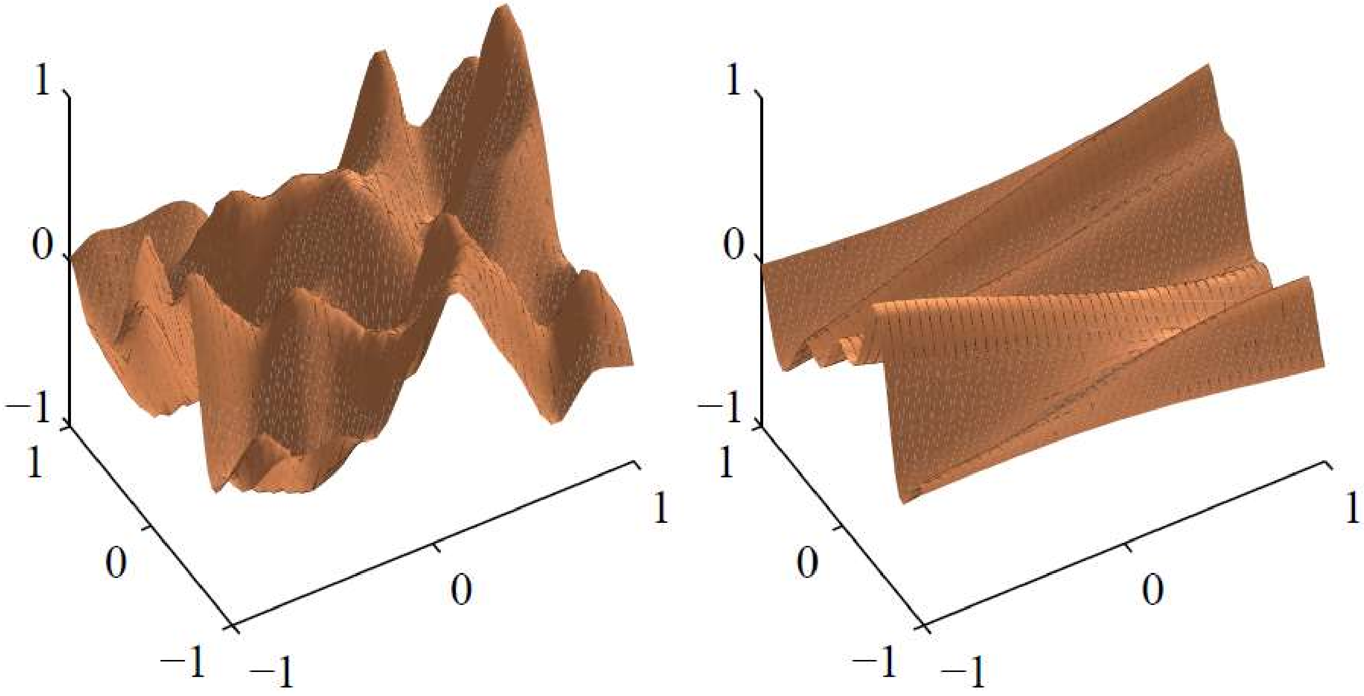
- Because $\ln p(\mathbf{t}|\theta)$ is in general a nonconvex function, thus having multiple maxima.

Automatic Relevance Determination

- Consider a Gaussian process with a two-dimensional input $\mathbf{x} = (x_1, x_2)$, with a kernel function

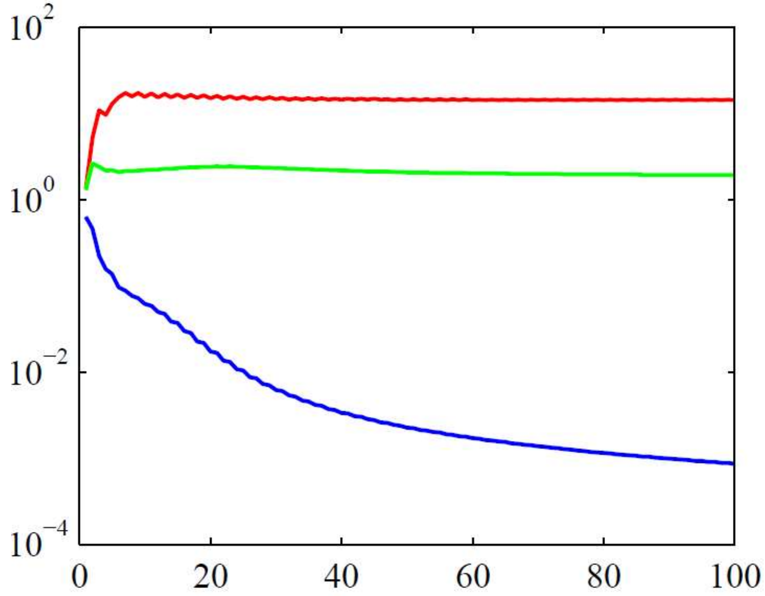
$$k(\mathbf{x}, \mathbf{x}') = \theta_0 \exp\left\{-\frac{1}{2}\sum_{i=1}^{2}\eta_i(x_i - x_i')^2\right\}$$

- Samples from the prior over functions $y(\mathbf{x})$ are shown for two different settings of the precision parameters η_i. As a particular parameter η_i becomes small, the function becomes relatively insensitive to the corresponding input variable x_i.


1
0
−1
1
0
−1
−1
0
1
1
0
−1
1
0
−1
−1
0
1

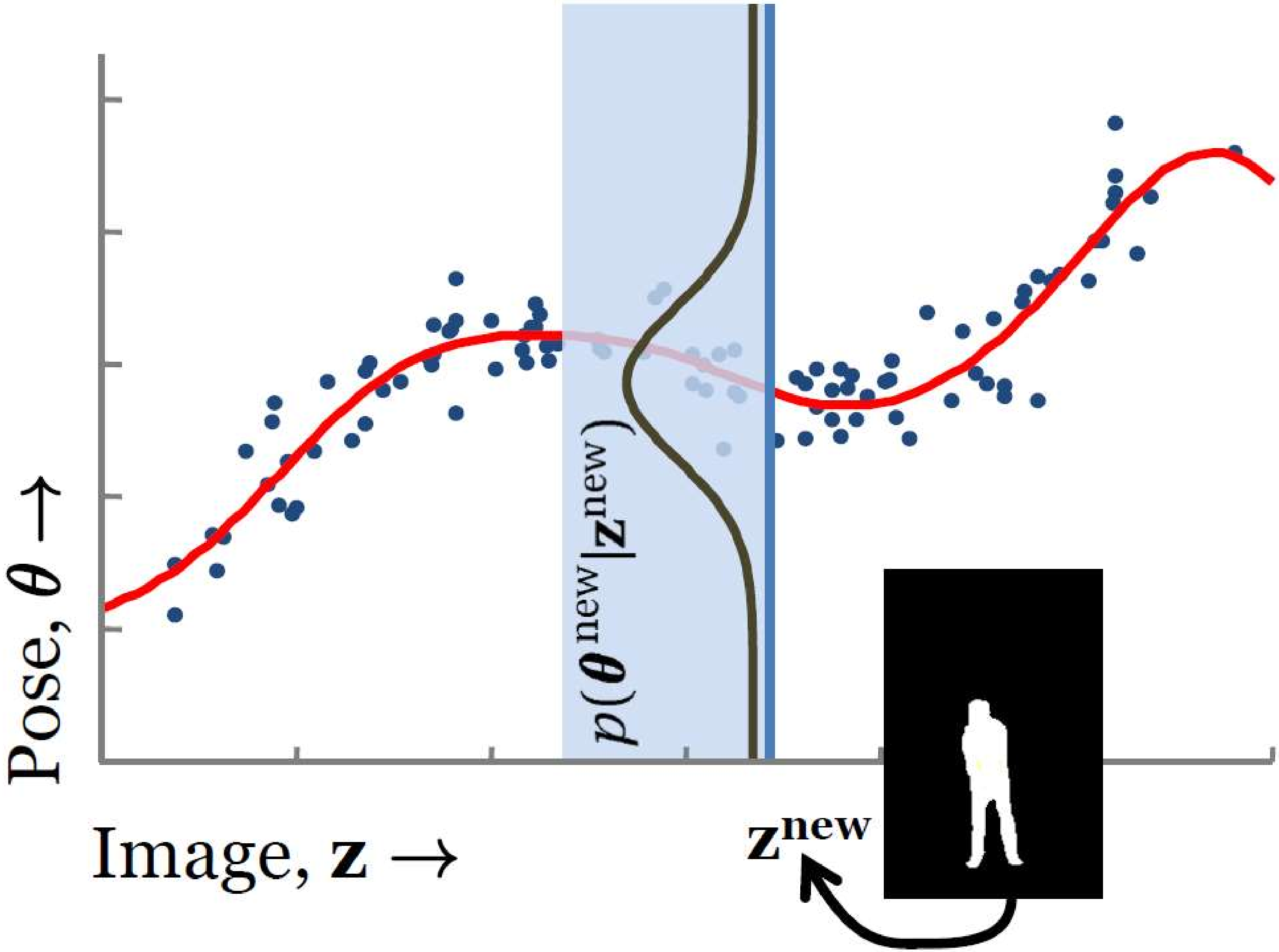
• The target variable t is generated by sampling 100 values of x_1 from a Gaussian, evaluating the function $\sin(2\pi x_1)$, and then adding Gaussian noise. Values of x_2 are given by copying the values of x_1 and adding noise, and values of x_3 are sampled from an independent Gaussian distribution.

The marginal likelihood is opimised w.r.t. η_1, η_2, η_3 by the gradient algorithm. The figure shows that x_1 is a good predictor of t, x_2 is a more noisy predictor, and x_3 is irrelevant for predicting t.



Back to the pose estimation problem

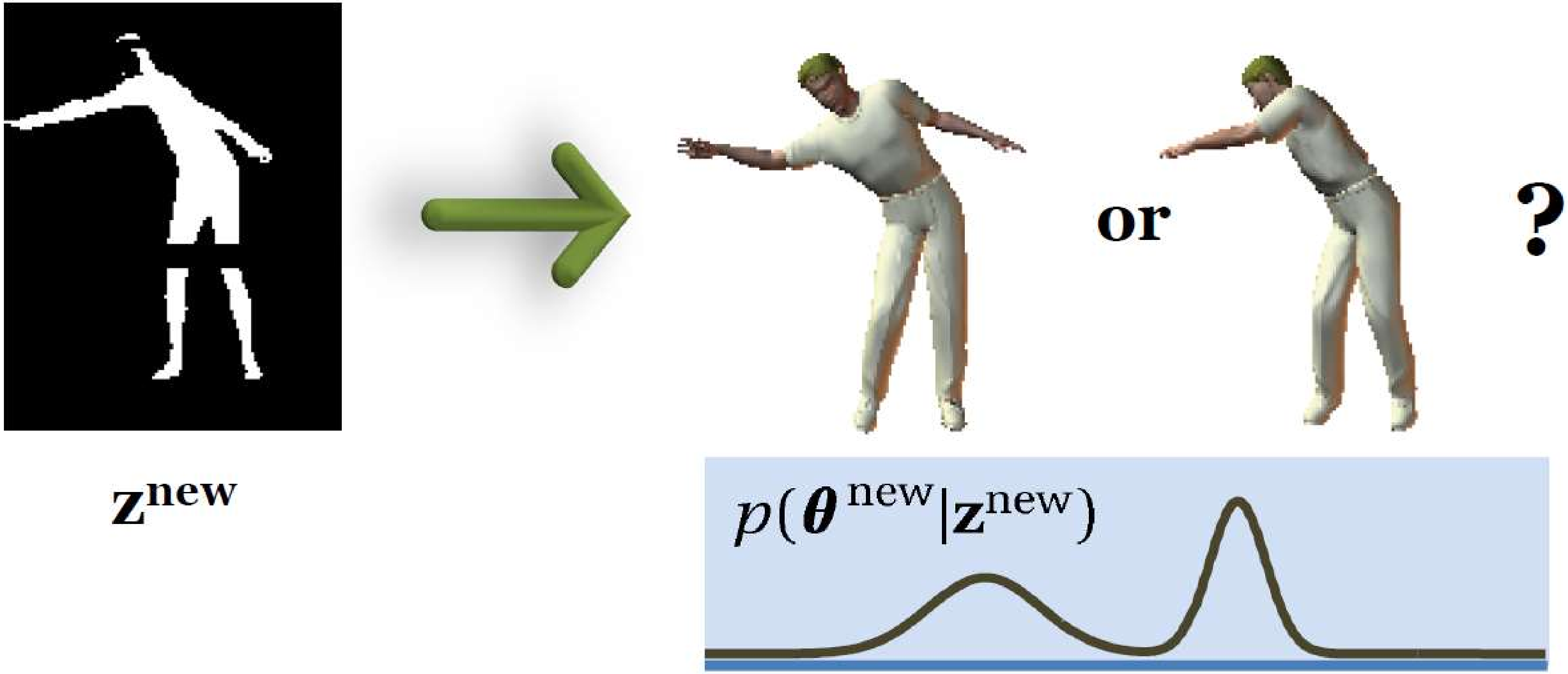
3. Or, more usefully, compute $p(\boldsymbol{\theta}^{\text{new}}|\mathbf{z}^{\text{new}})$.



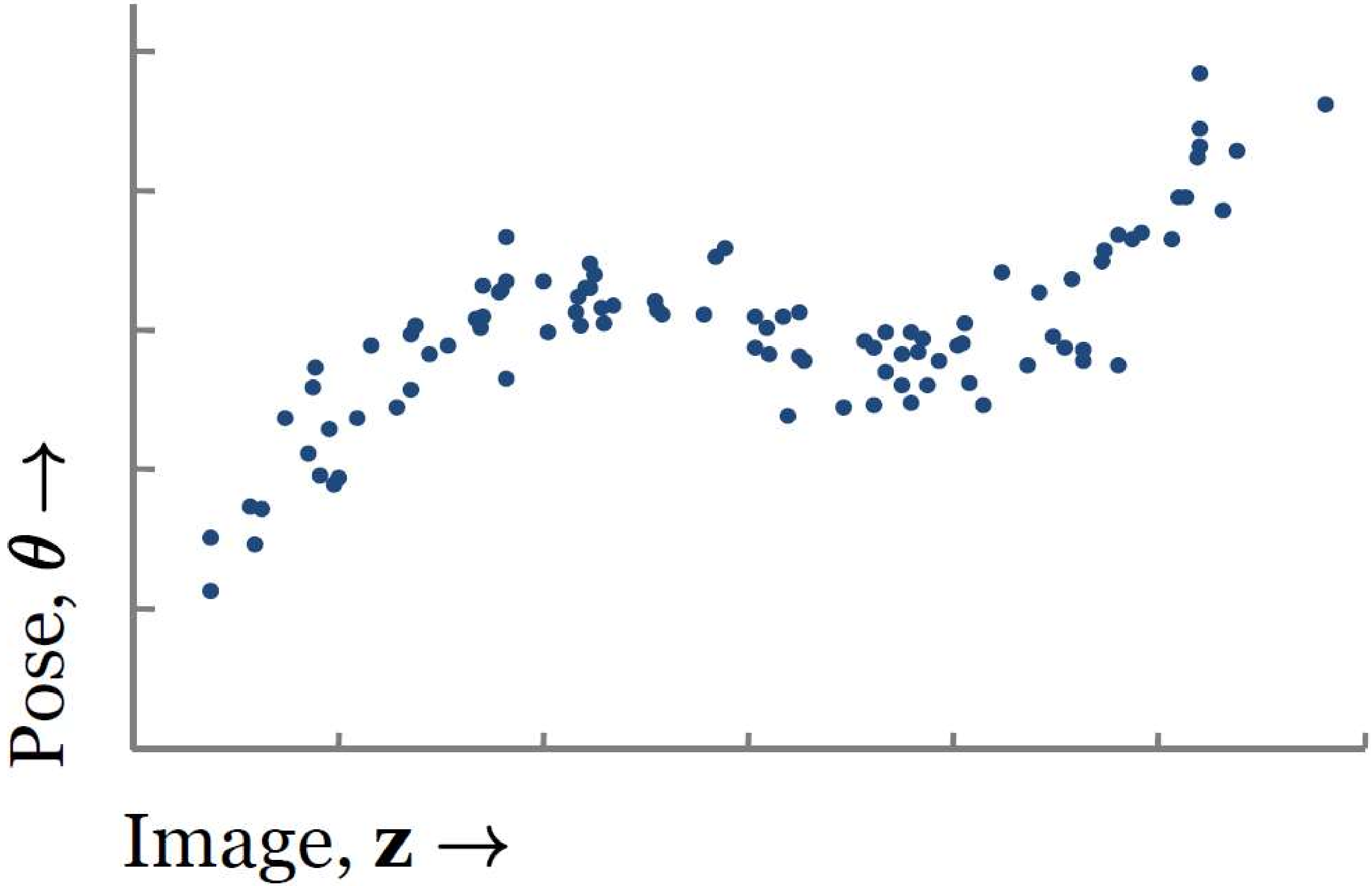
- It’ll never work…
  - f is multivalued
  - $\mathbf{z}$ and θ live in high dimensions

Multivalued f:

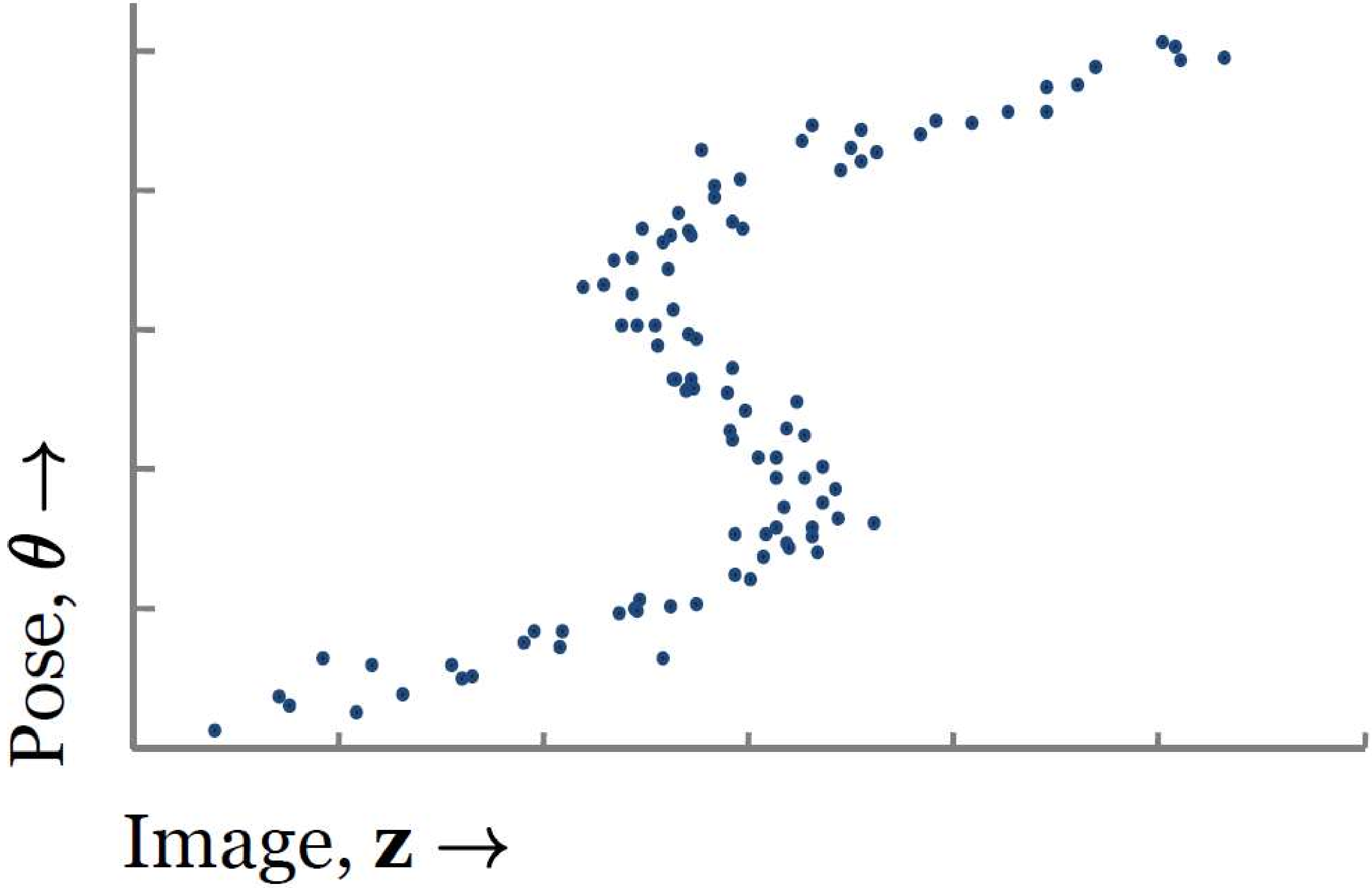
Multivalued f:



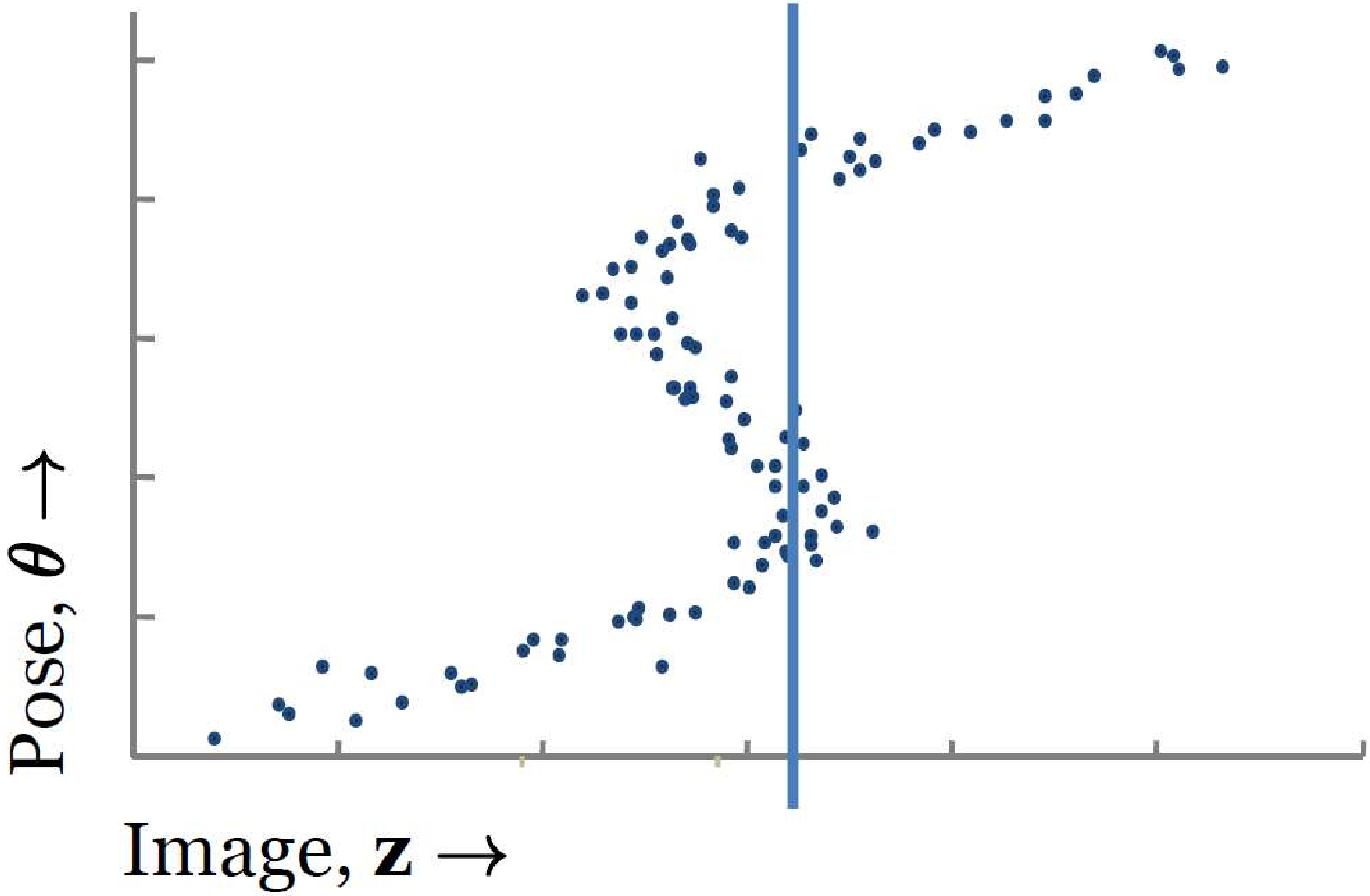
Instead of this:



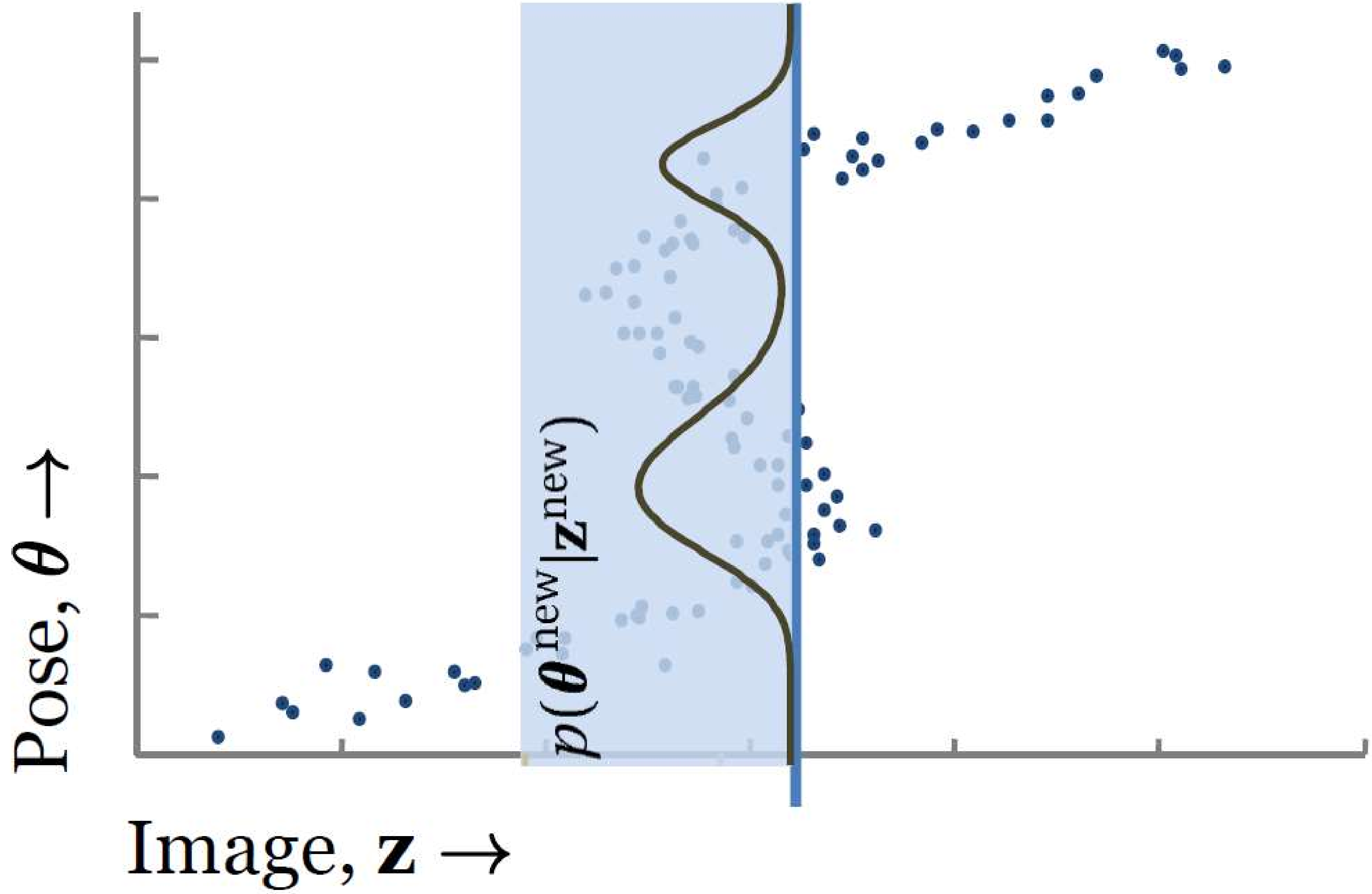
We have this:



We have this:

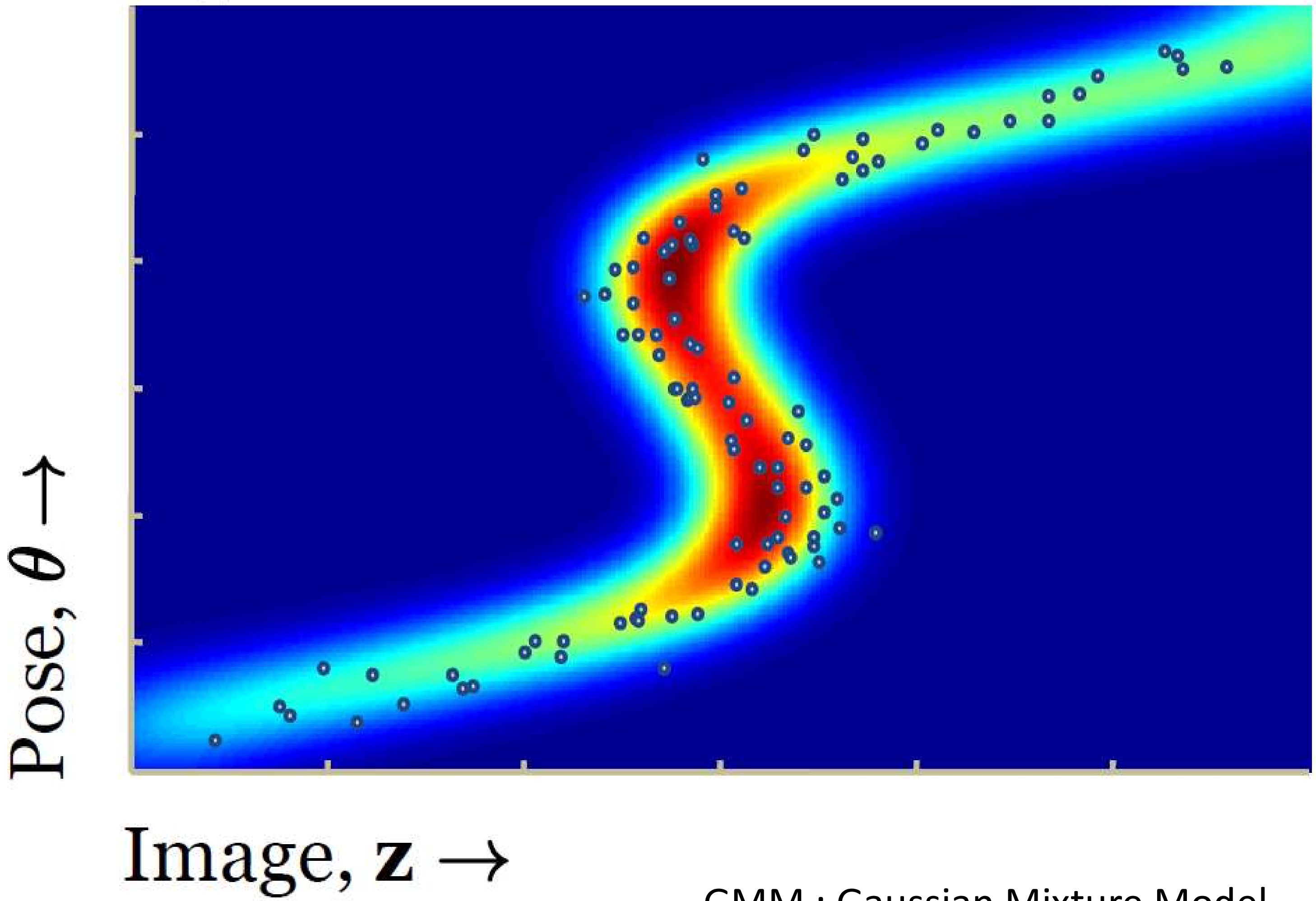


We have this:



Pose, $\boldsymbol{\theta} \rightarrow$
$p(\boldsymbol{\theta}^{\text{new}}|\mathbf{z}^{\text{new}})$
Image, $\mathbf{z} \rightarrow$

Instead of fitting $p(\boldsymbol{\theta}|\mathbf{z})$, fit $p(\boldsymbol{\theta},\mathbf{z})$, e.g. with GMM, Parzen, or GPLVM.

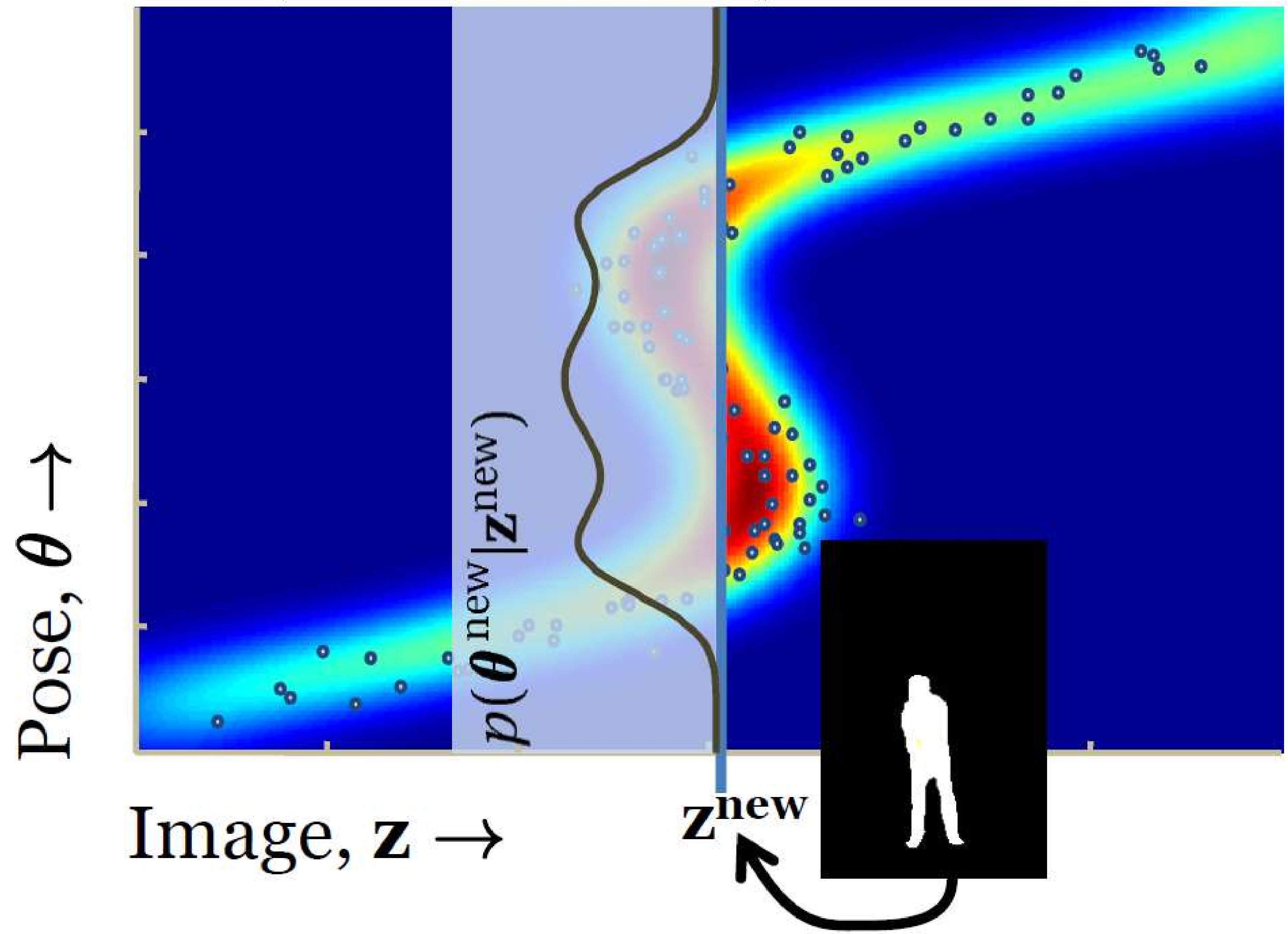


GMM : Gaussian Mixture Model
GPLVM : Gaussian Process Latent Variable Model

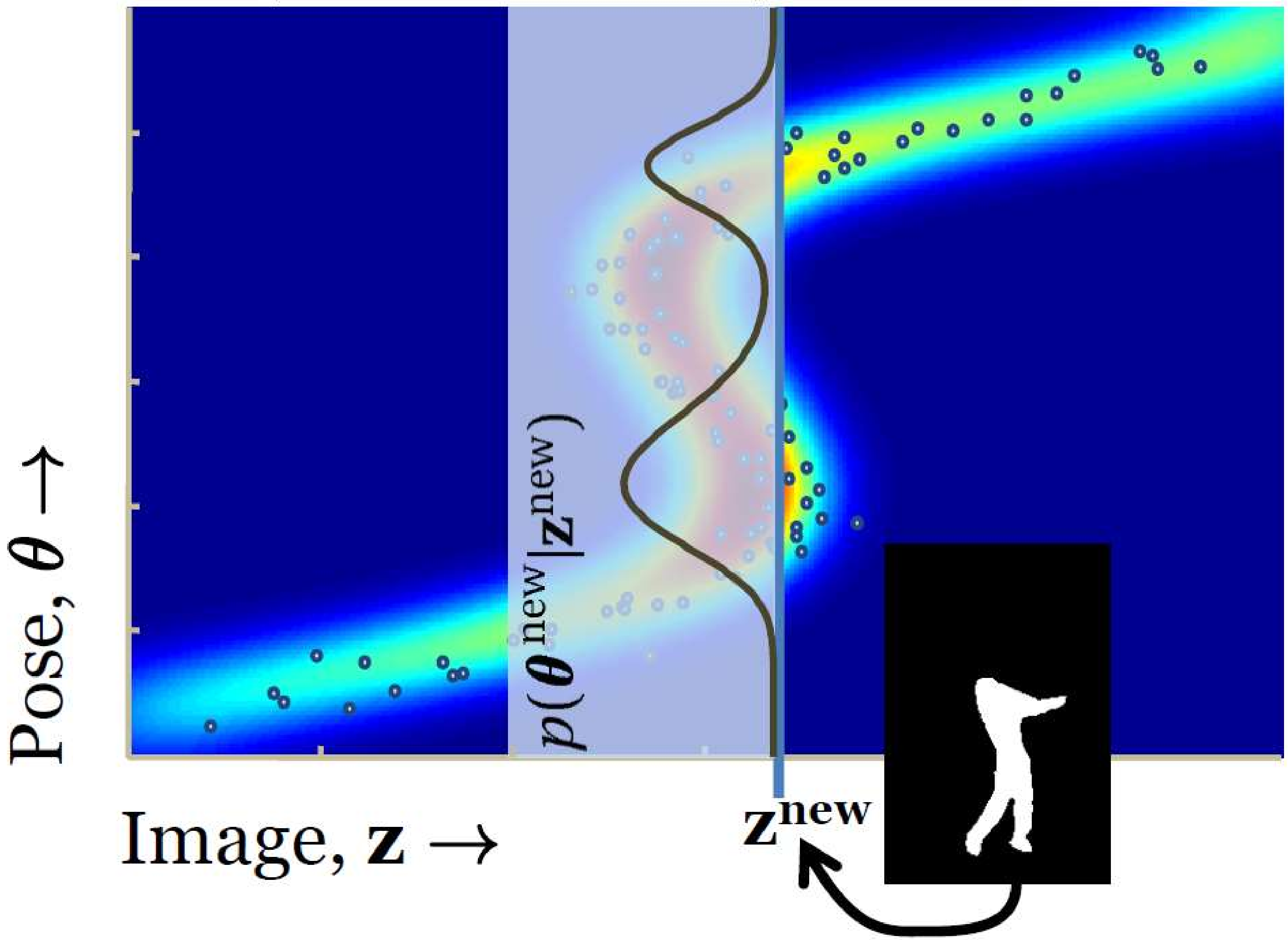
Given new image $\mathbf{z}^{\text{new}}$, conditional $p(\boldsymbol{\theta}|\mathbf{z}^{\text{new}})$ is computed from the joint.

$$p(\theta|z) = \frac{p(\theta,z)}{p(z)}$$

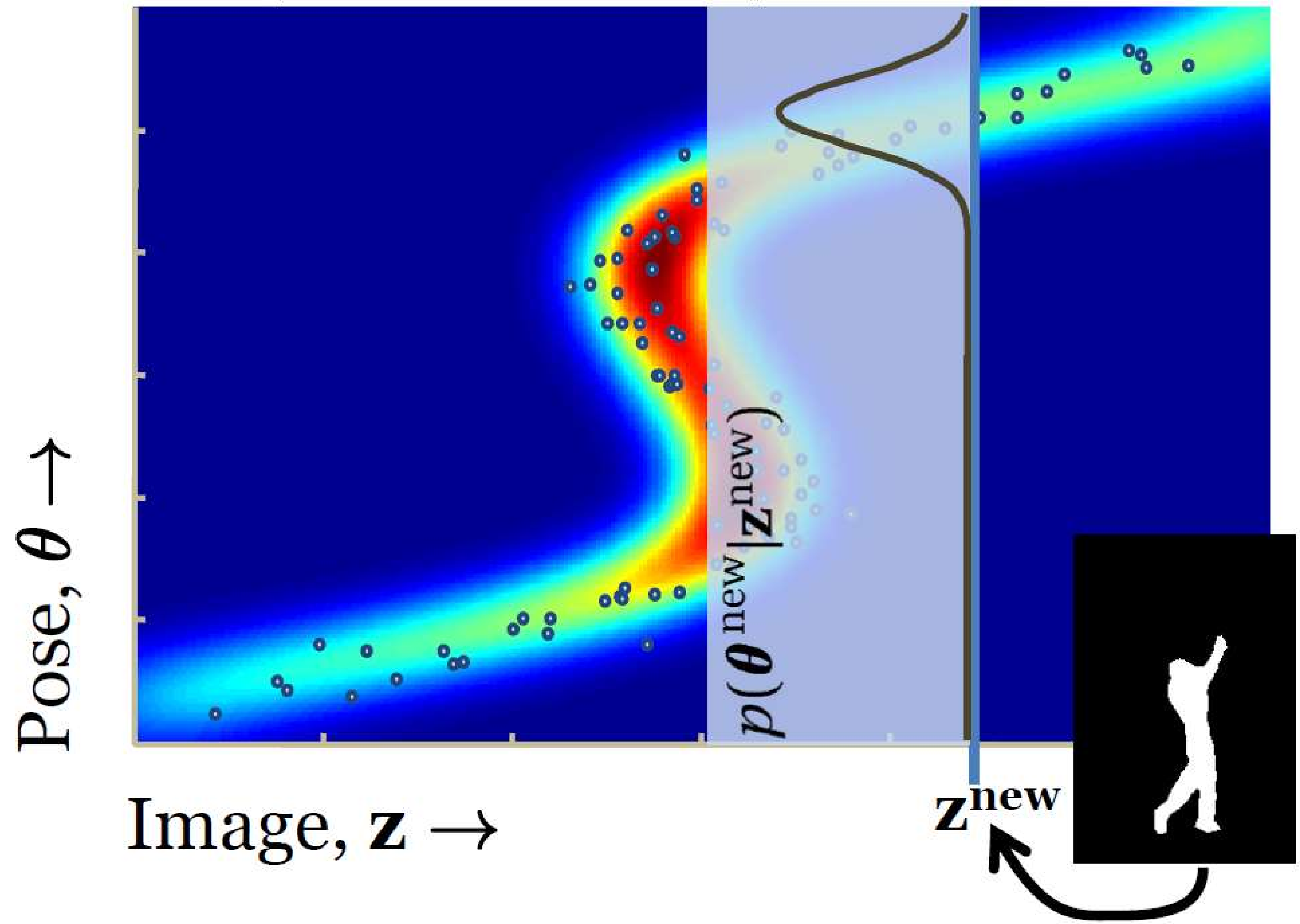
$$p(z) = \int p(\theta,z)d\theta$$

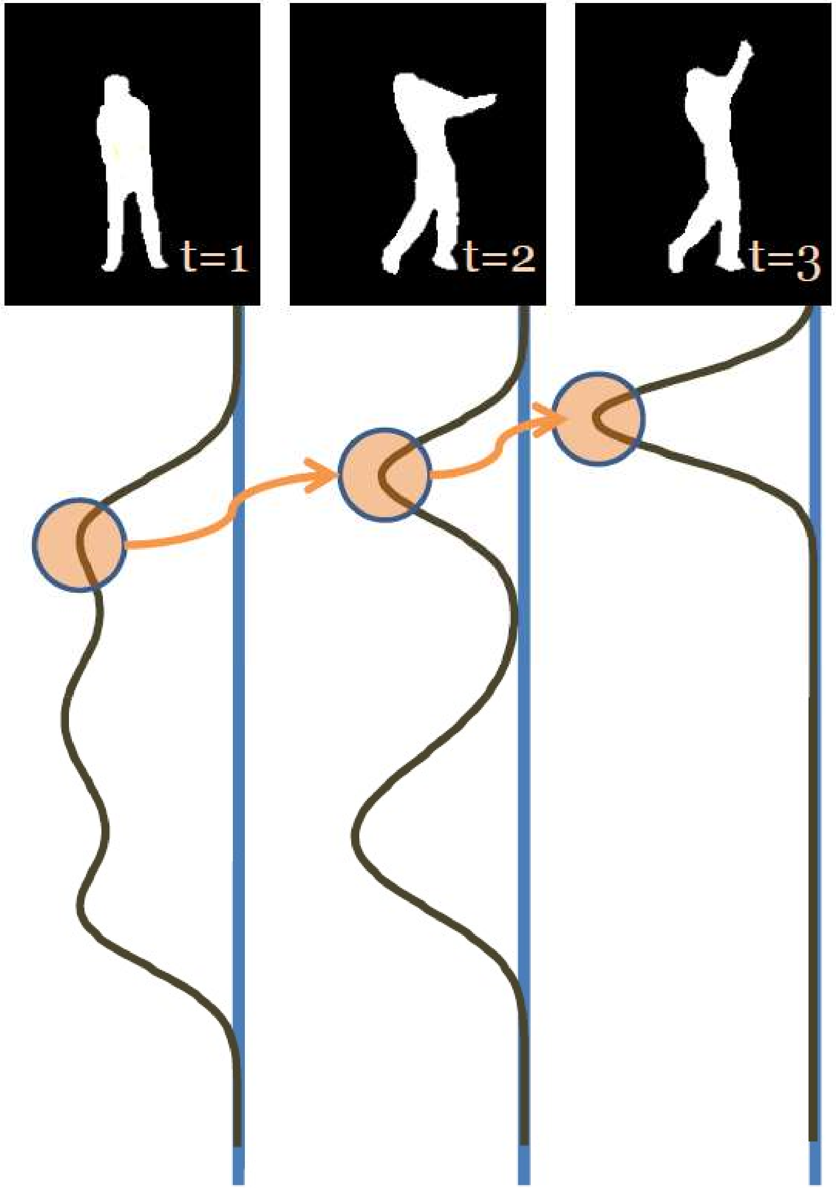


Given new image $\mathbf{z}^{\text{new}}$, conditional $p(\boldsymbol{\theta}|\mathbf{z}^{\text{new}})$ is computed from the joint.



Given new image $\mathbf{z}^{\text{new}}$, conditional $p(\boldsymbol{\theta}|\mathbf{z}^{\text{new}})$ is computed from the joint.



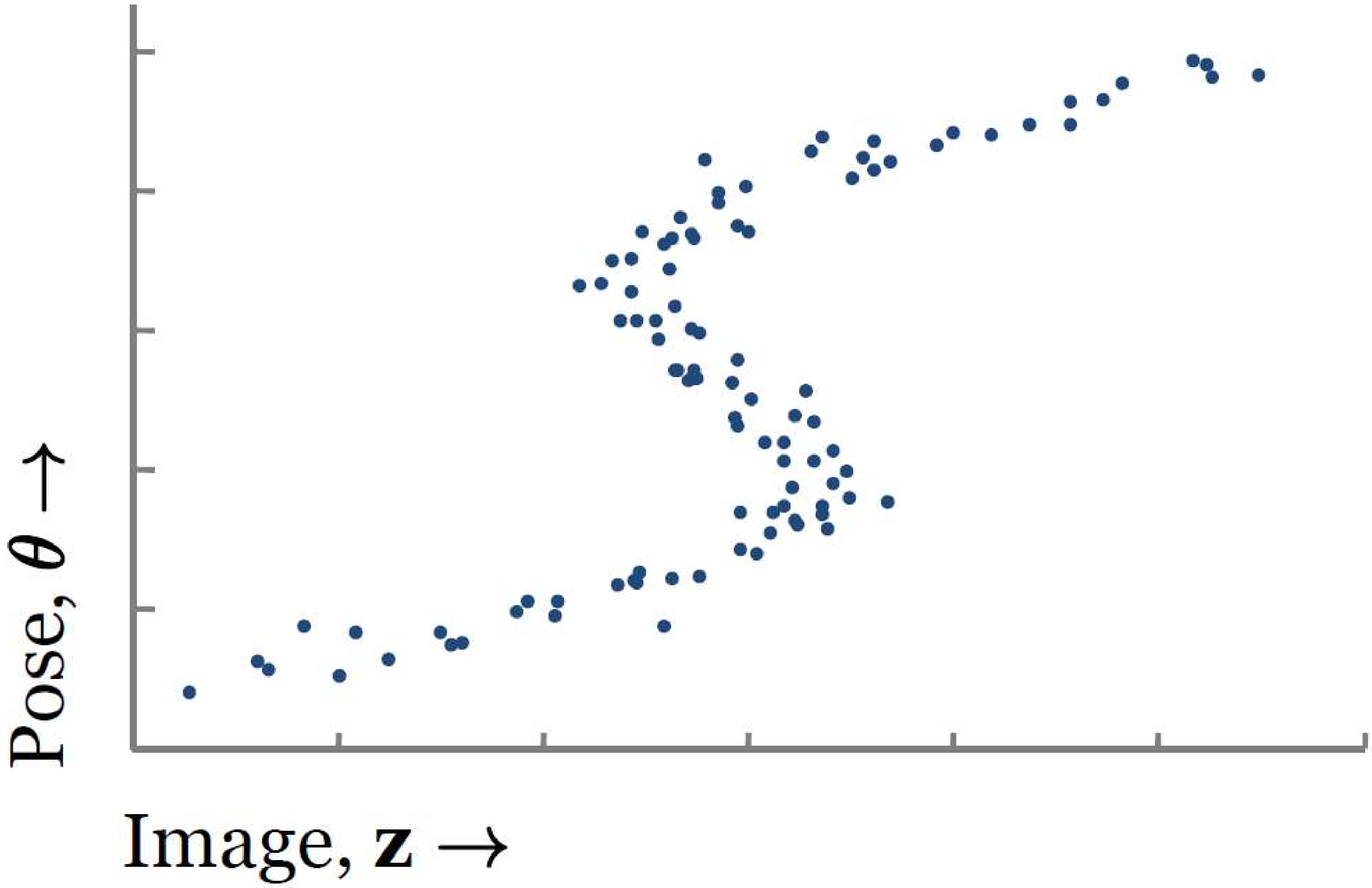


For a video sequence:

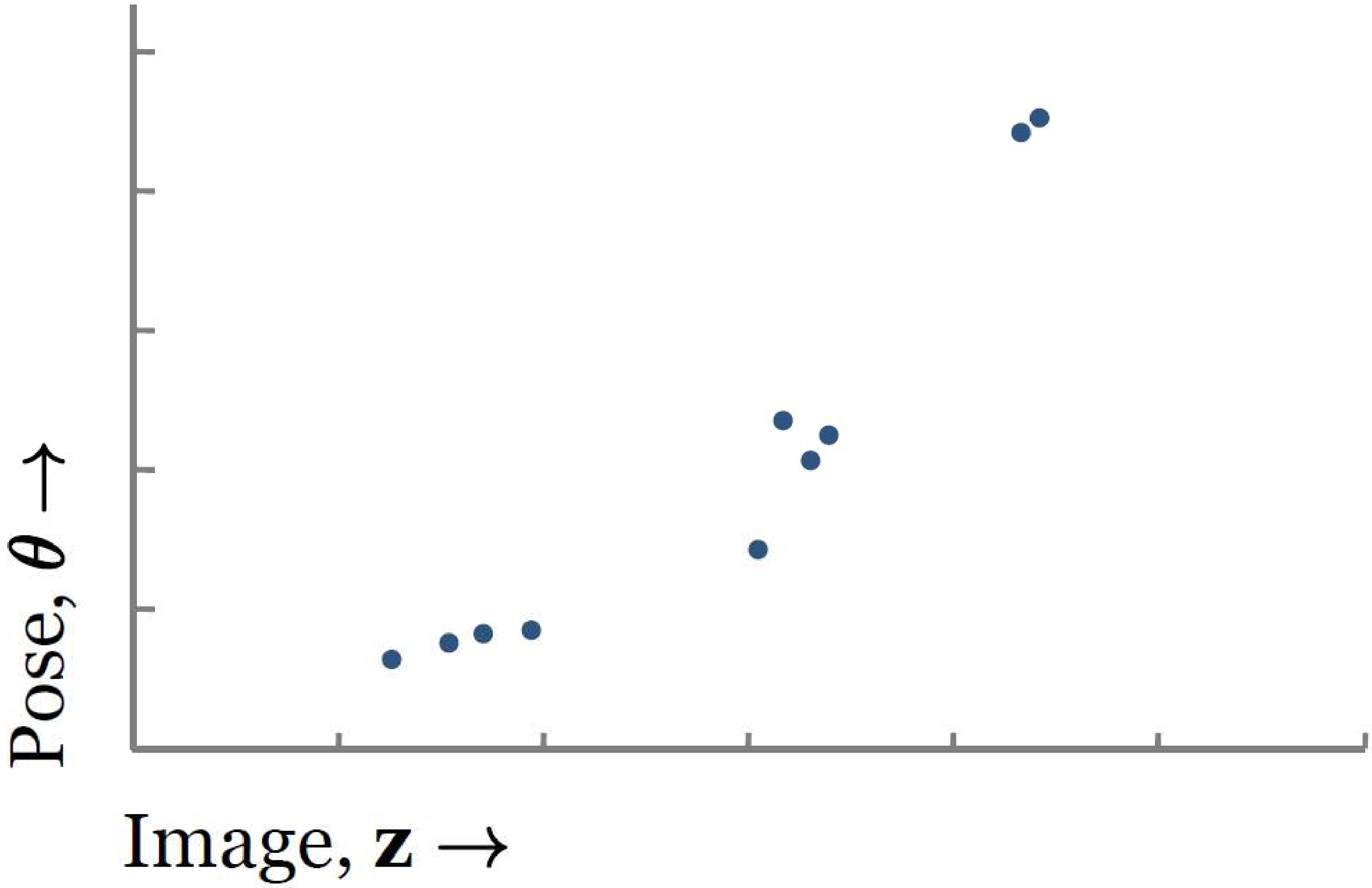
- Compute modes of conditional at every frame
- Choose sequence of modes to maximize product of likelihood and temporal smoothness using Viterbi

But…

Instead of this:

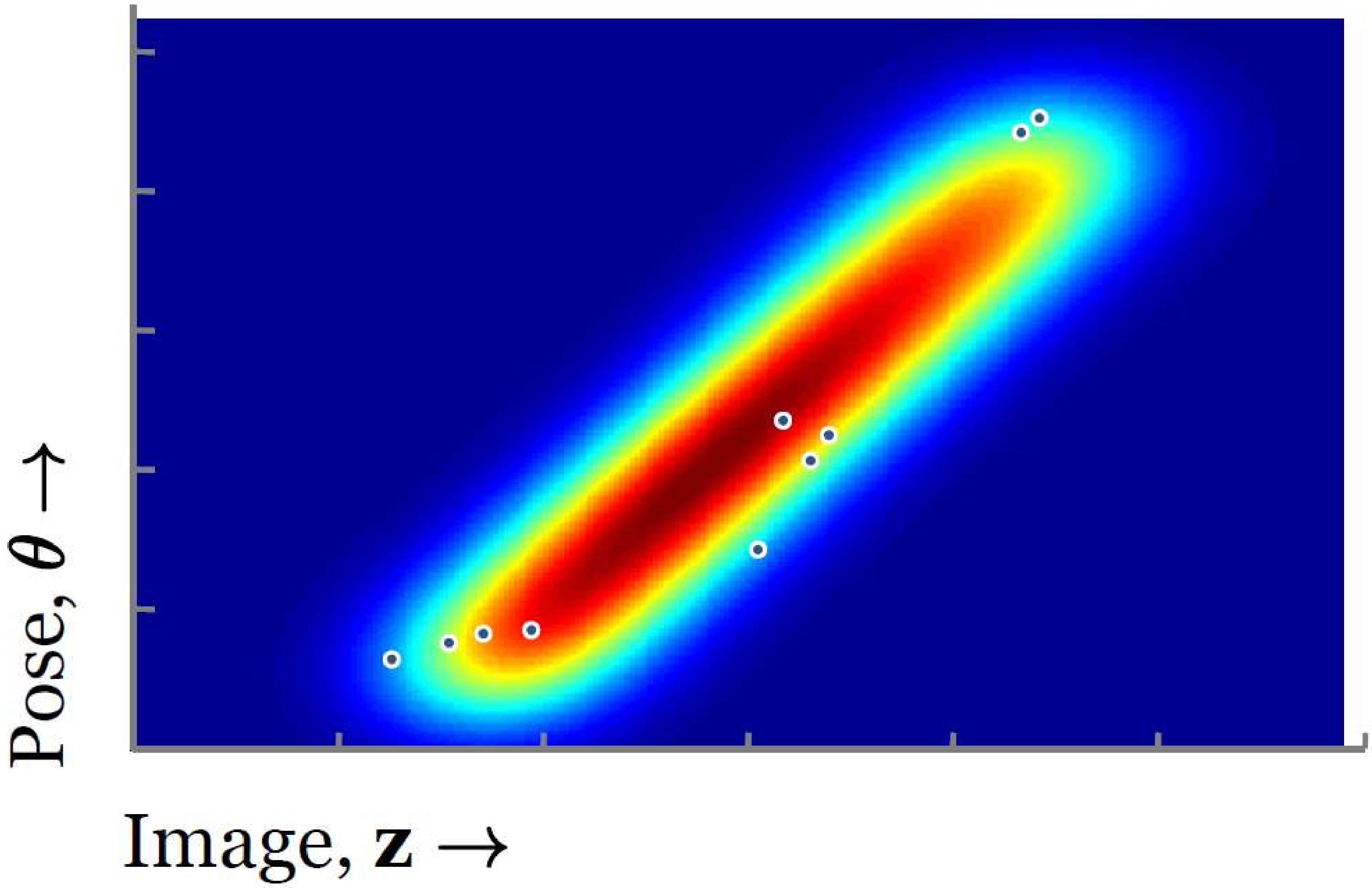


We have this:

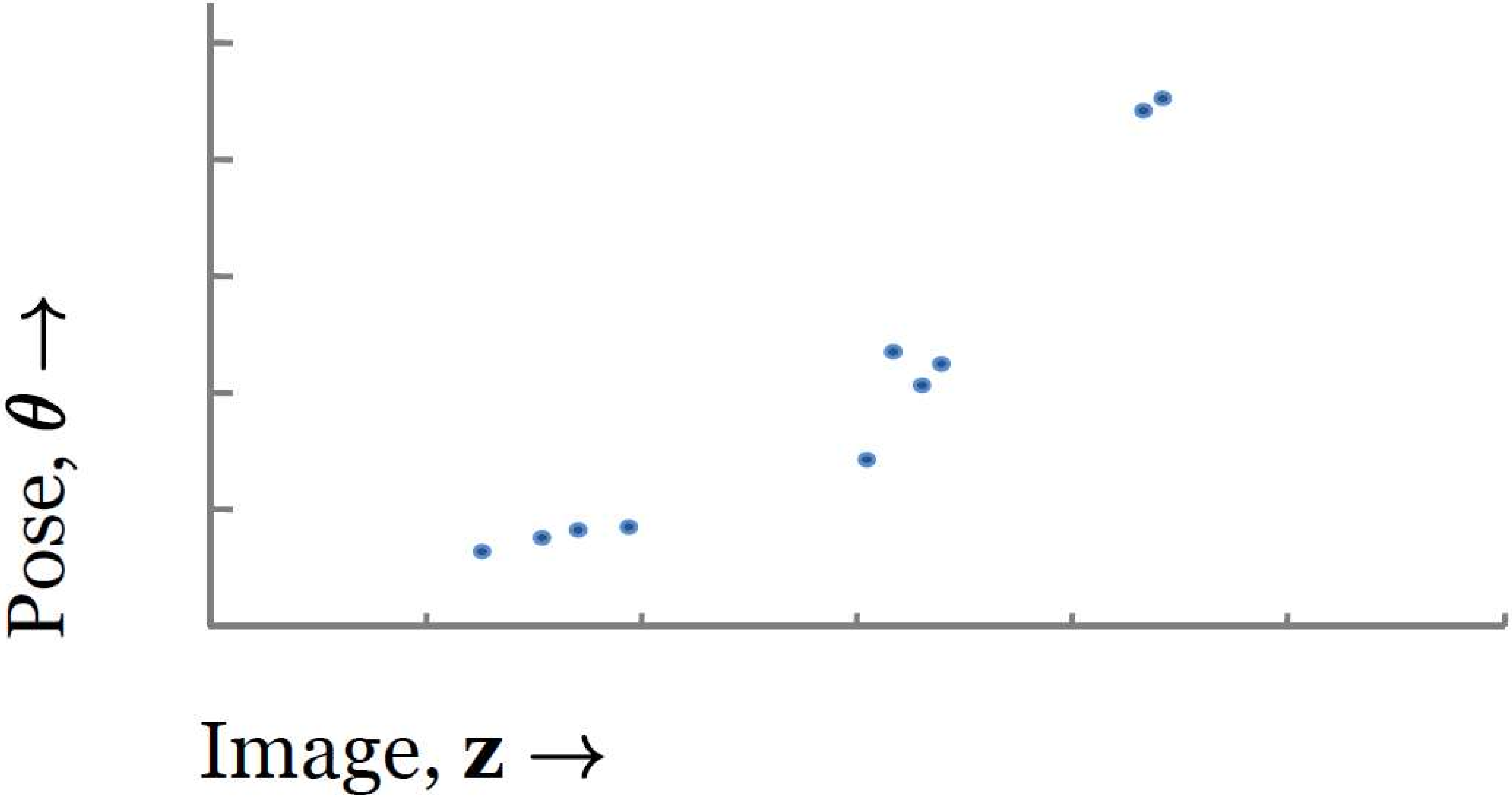


Pose, $\boldsymbol{\theta} \rightarrow$
Image, $\mathbf{z} \rightarrow$

Of which a not unreasonable density estimate is:

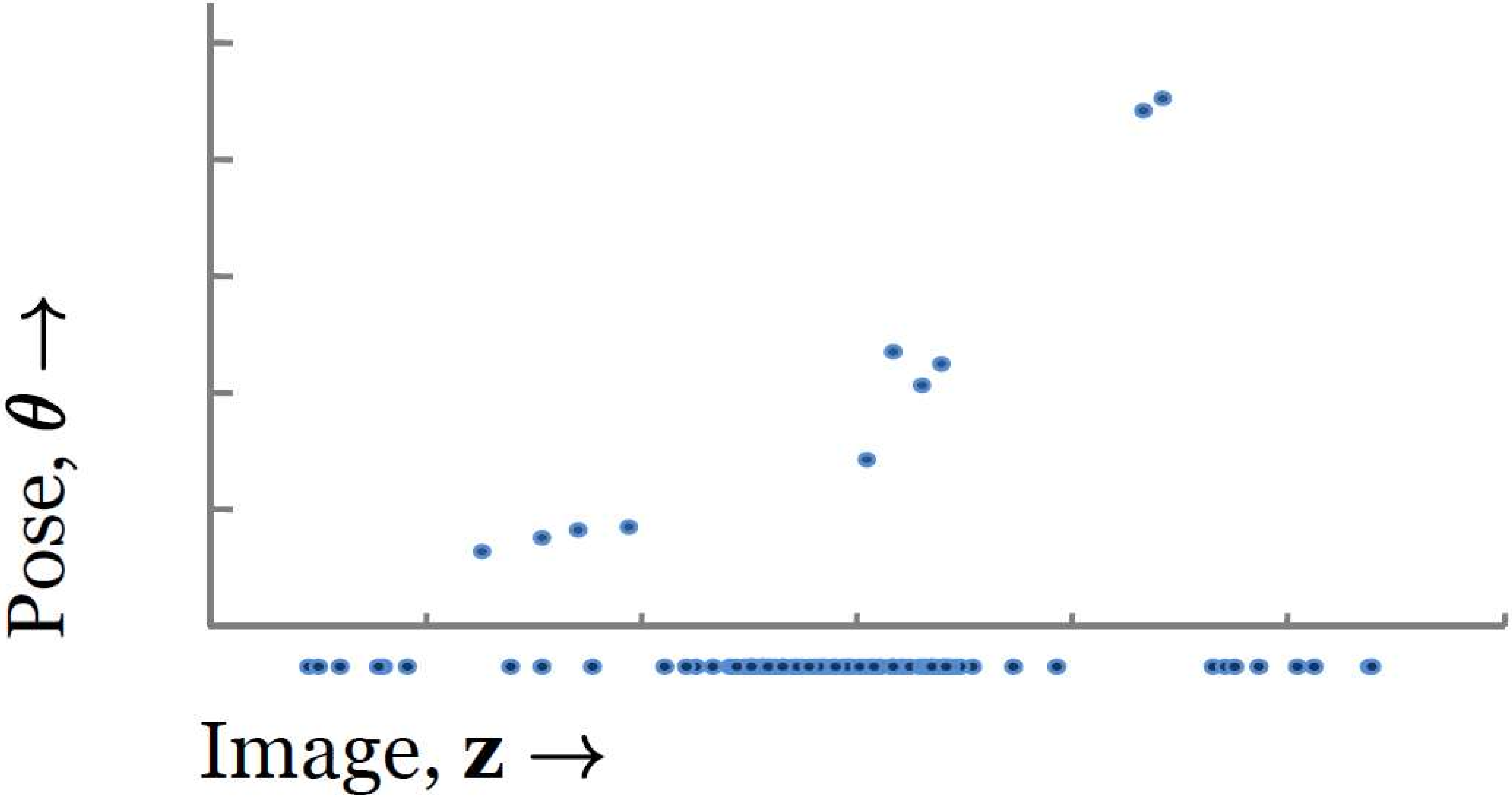


We have too little training data, i.e. too few labelled $(\mathbf{z},\theta)$ pairs

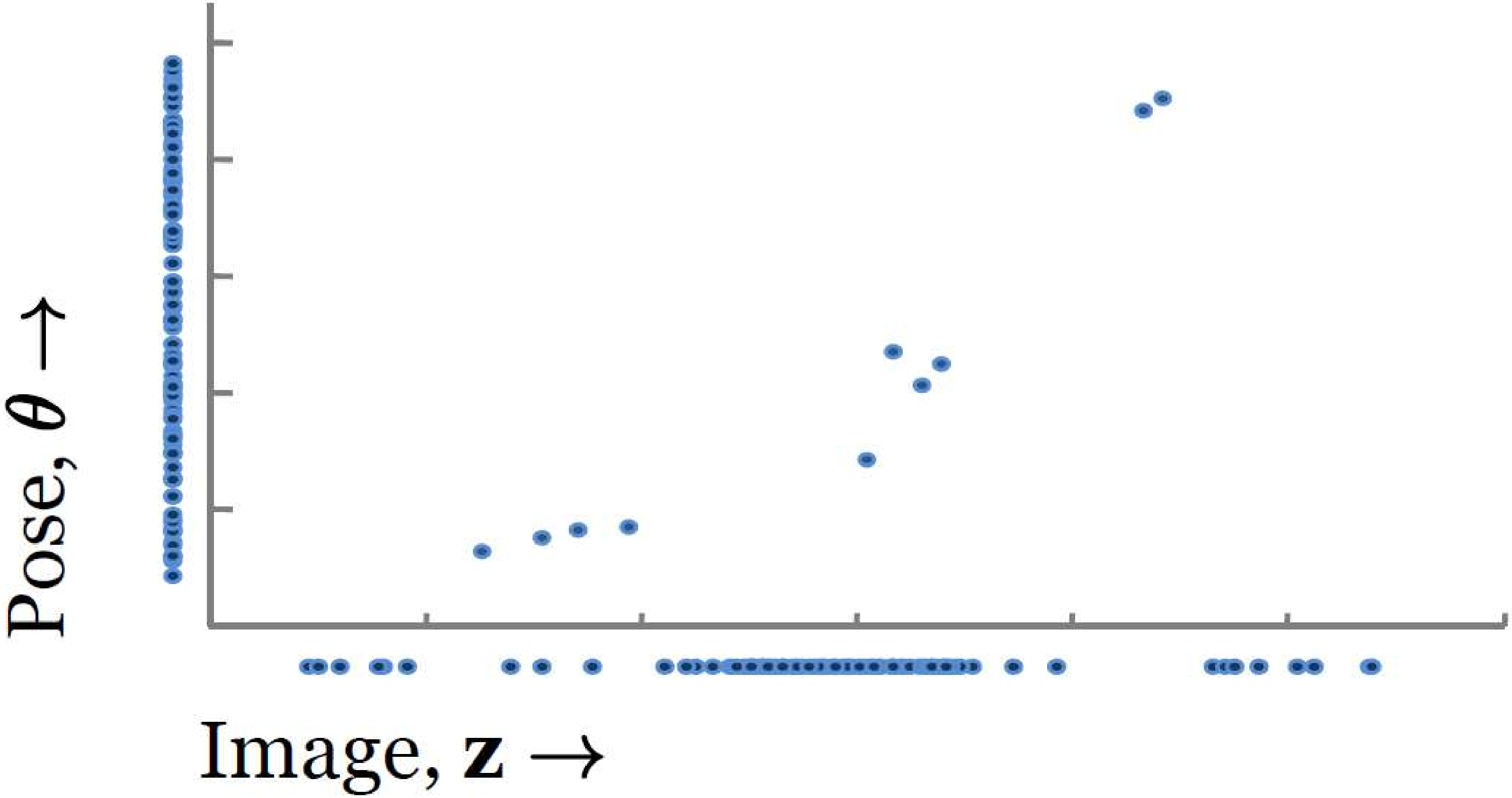


We can't get more because labelling images is expensive...

But we can easily capture more ***unlabelled*** images, i.e. (z,*) pairs

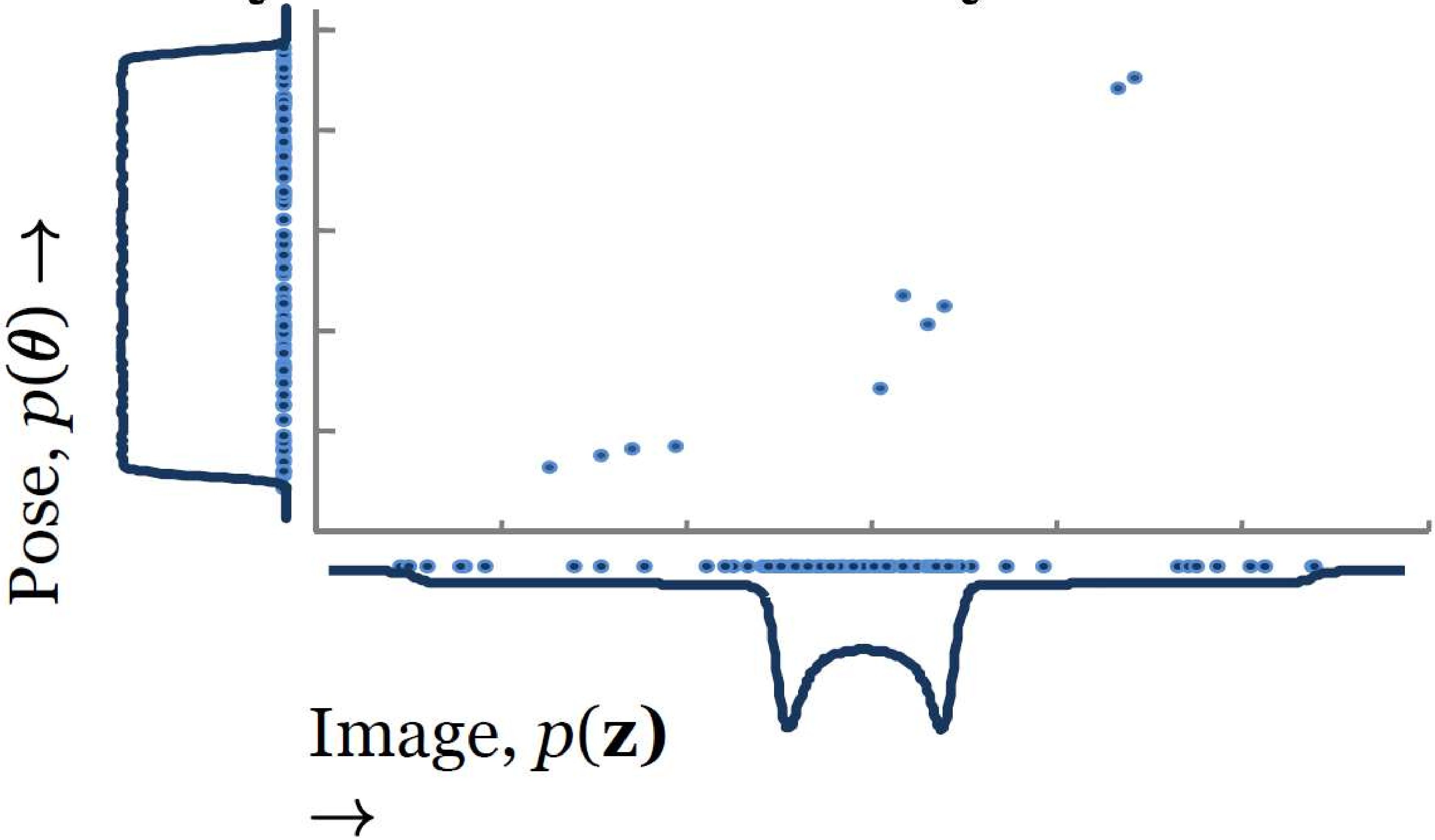


And we can easily download more mocap data without images, i.e. more (*,θ) pairs

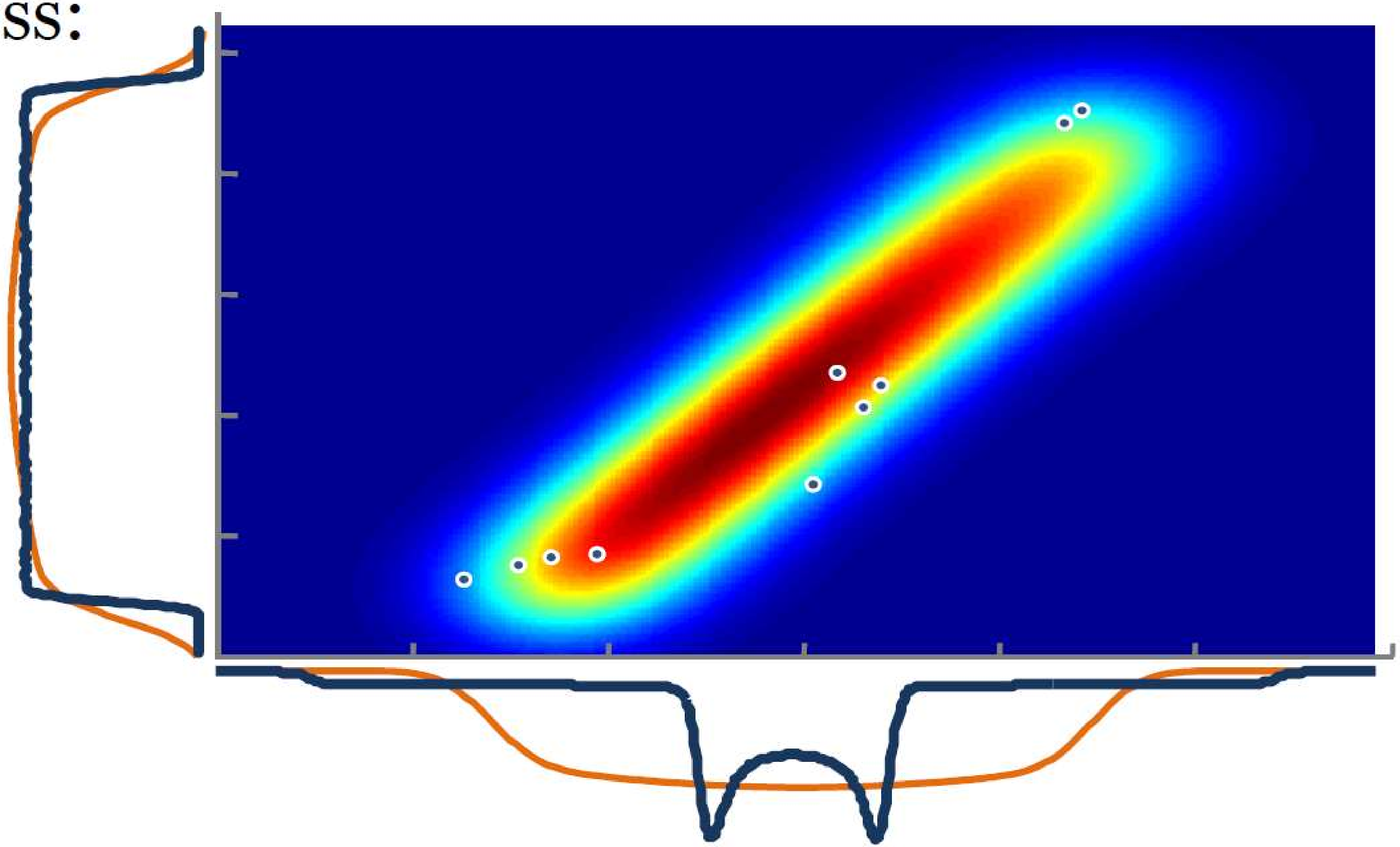


In fact, it's as if we know the **marginals**

$p(\boldsymbol{\theta}) = \int p(\mathbf{z}, \boldsymbol{\theta}) d\mathbf{z}$ and $p(\mathbf{z}) = \int p(\mathbf{z}, \boldsymbol{\theta}) d\boldsymbol{\theta}$

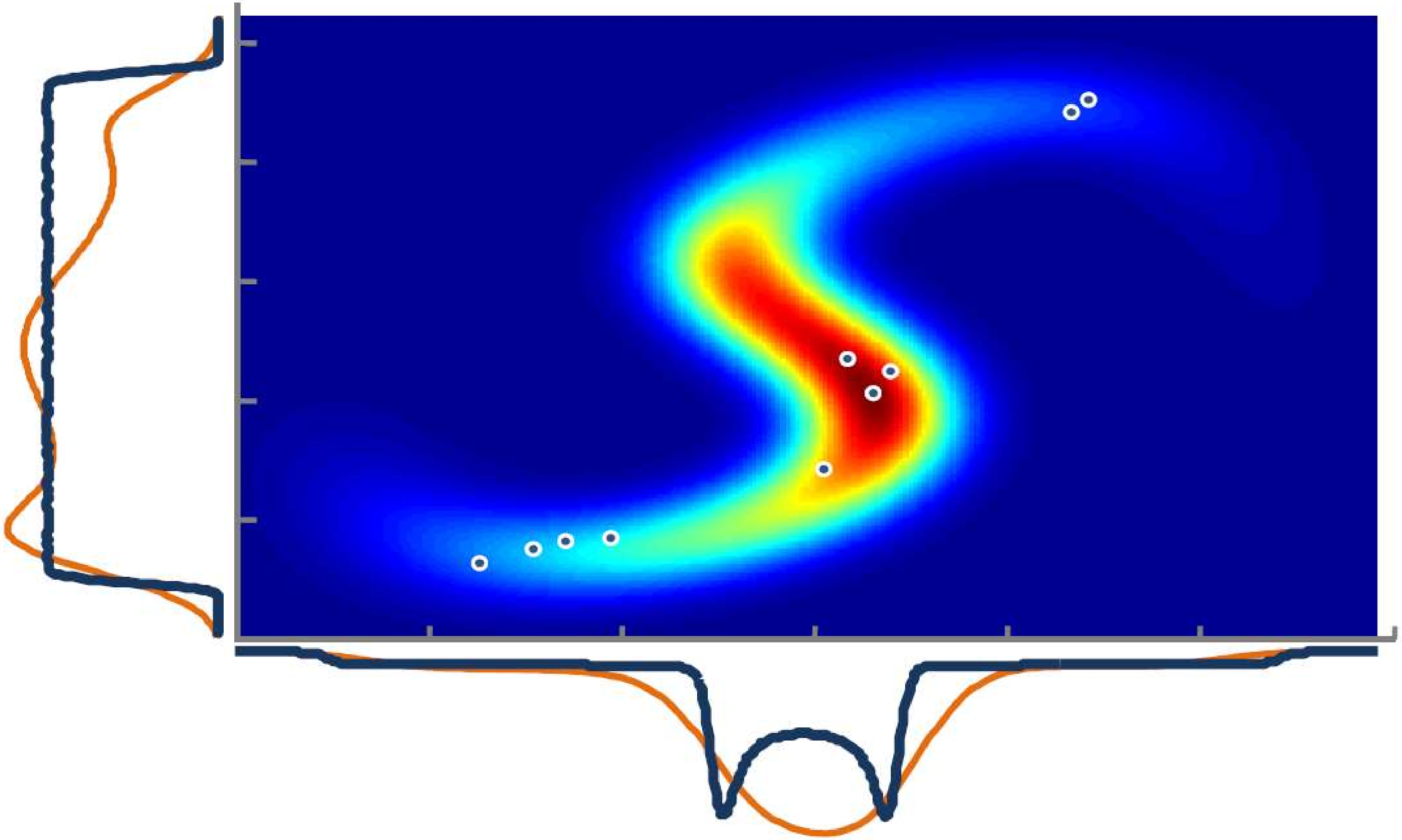


Which contradict the marginals of our earlier guess:



Imperial College London

[ffwd] Using the marginal samples gives this:



Joint manifold model

t: a continuous latent variable

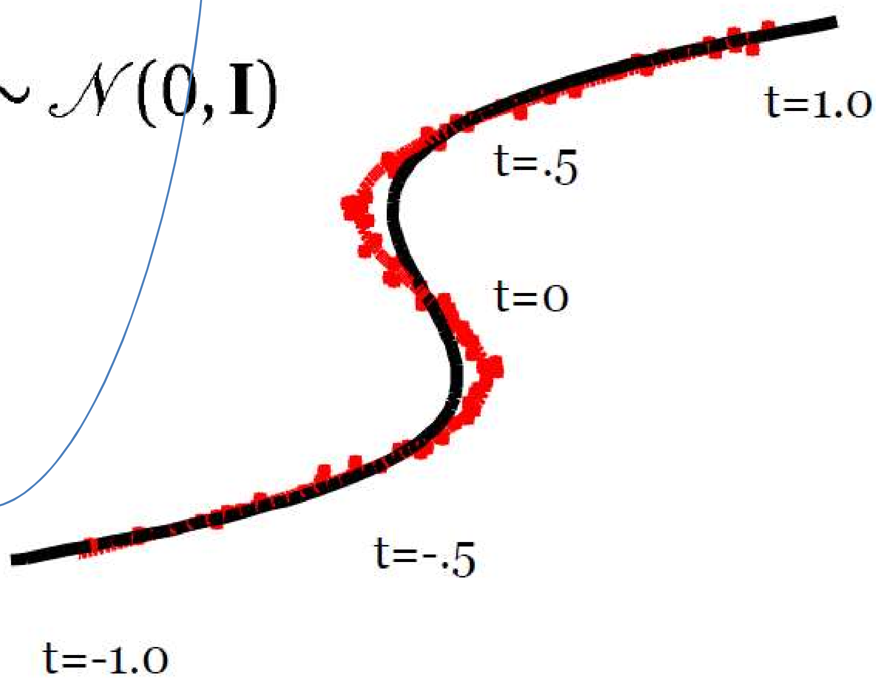
Joint density $p(\mathbf{z},\boldsymbol{\theta}) = \int p(\boldsymbol{\theta}|\mathbf{t})p(\mathbf{z}|\mathbf{t})p(\mathbf{t})d\mathbf{t}$

Or, loosely, the "spine" of the joint density is a manifold

$$\begin{pmatrix} \boldsymbol{\theta}(\mathbf{t}) \\ \mathbf{z}(\mathbf{t}) \end{pmatrix} \qquad \mathbf{t} \sim \mathcal{N}(0,\mathbf{I})$$

plus noise.

$$p(z,\theta) = \int p(z,\theta|t)p(t)dt$$



Joint manifold model

Find latent variables $\mathbf{t}$ to maximize
the posterior of training data $\mathbf{D} = \{(\boldsymbol{\theta}_l, \mathbf{z}_l)\}_{l=1}^{L}$

$$\begin{aligned} p(\mathbf{t}_{1..L}|\mathbf{D}) &\propto p(\mathbf{D}|\mathbf{t}_{1..L})p(\mathbf{t}_{1..L}) \\ &= p(\boldsymbol{\theta}_{1..L}, \mathbf{z}_{1..L}|\mathbf{t}_{1..L})p(\mathbf{t}_{1..L}) \\ &= p(\boldsymbol{\theta}_{1..L}|\mathbf{t}_{1..L})p(\mathbf{z}_{1..L}|\mathbf{t}_{1..L})p(\mathbf{t}_{1..L}). \end{aligned}$$

Adding the marginal samples…

Maximize the posterior of

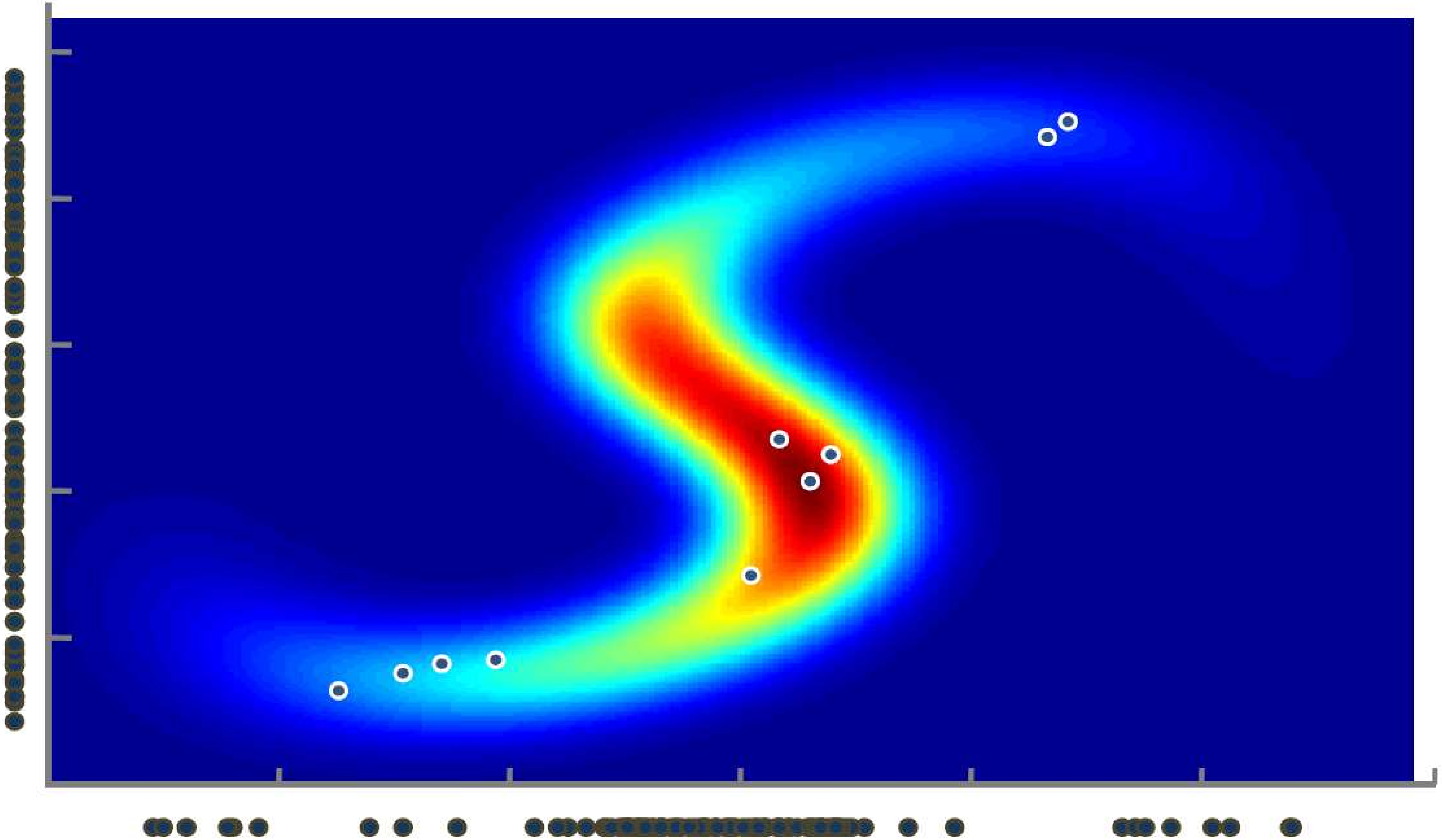
- joint training data $\mathbf{D} = \{(\boldsymbol{\theta}_l, \mathbf{z}_l)\}_{l=1}^{L}$
- marginal $\boldsymbol{\theta}$ data $\mathbf{M} = \{(\boldsymbol{\theta}_l^*, *)\}_{l=1}^{L}$
- marginal $\mathbf{z}$ data $\mathbf{N} = \{(*, \mathbf{z}_l^*)\}_{l=1}^{L}$

$$p(\boldsymbol{\theta}_{1..L} | \mathbf{t}_{1..L}) p(\mathbf{z}_{1..L} | \mathbf{t}_{1..L}) \times$$

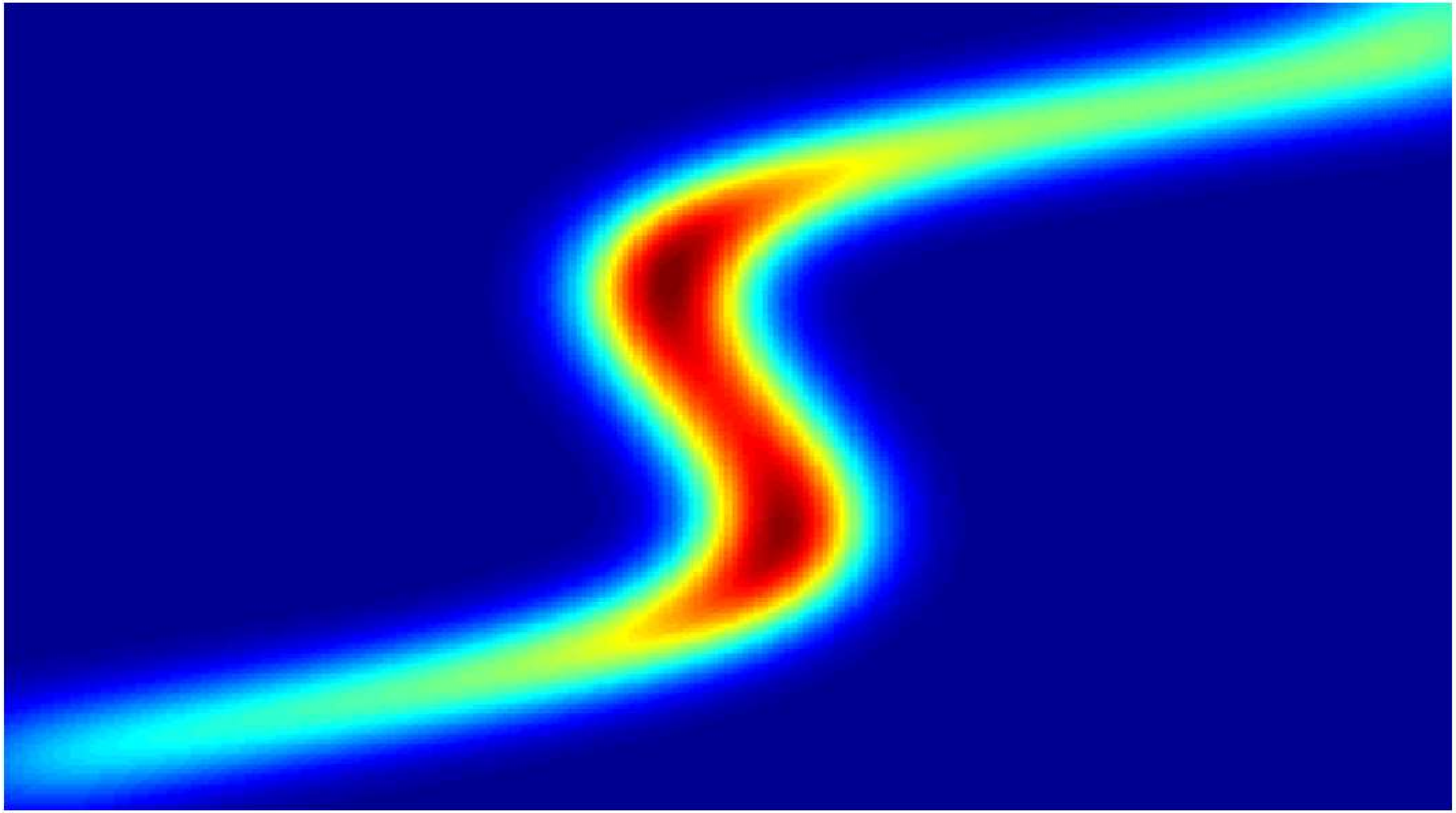
$$p(\boldsymbol{\theta}^*_{1..L} | \mathbf{t}^{\boldsymbol{\theta}}_{1..L}) \times p(\mathbf{z}^*_{1..L} | \mathbf{t}^{\mathbf{z}}_{1..L}) \times$$

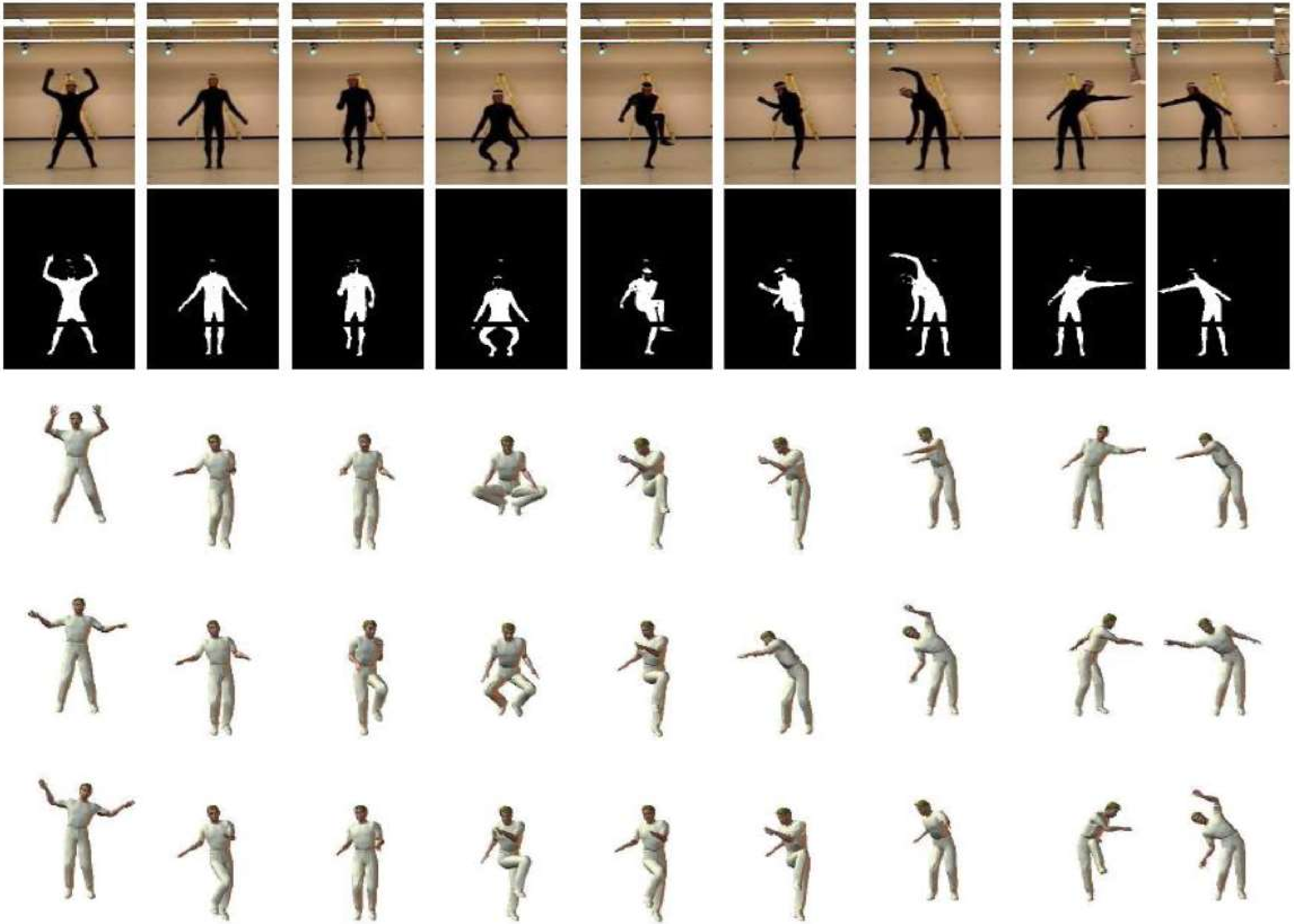
$$p(\mathbf{t}_{1..L}, \mathbf{t}^{\boldsymbol{\theta}}_{1..L}, \mathbf{t}^{\mathbf{z}}_{1..L})$$

Optimizing using scaled CG gives:

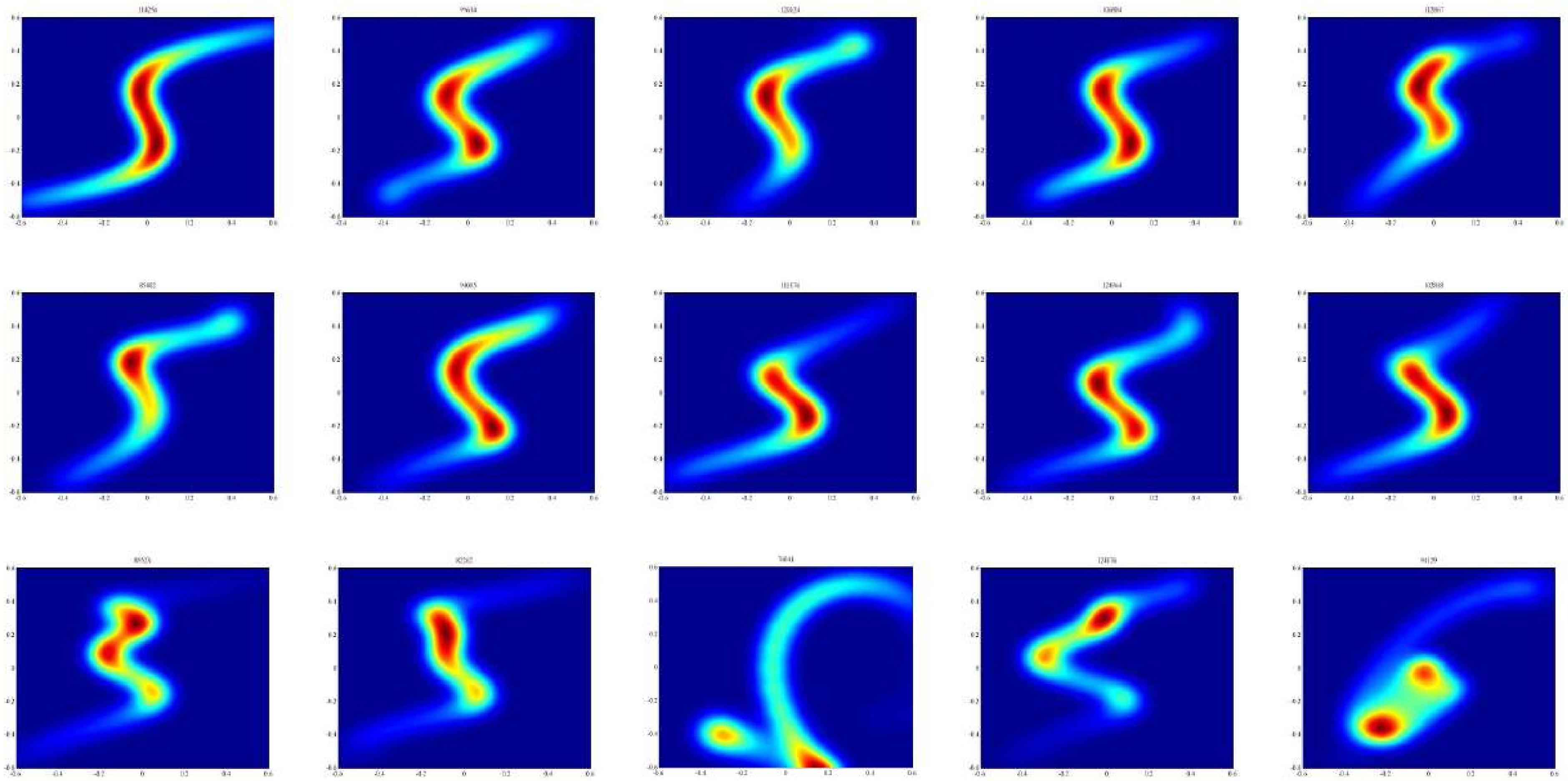


The estimate from 100 training pairs:

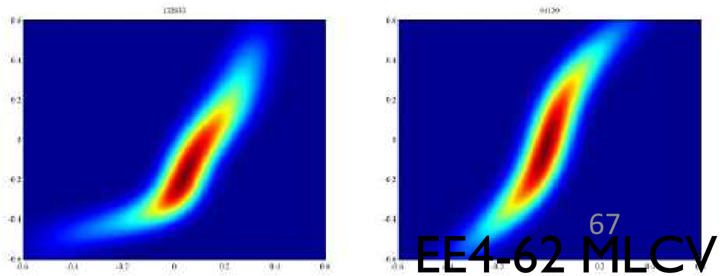




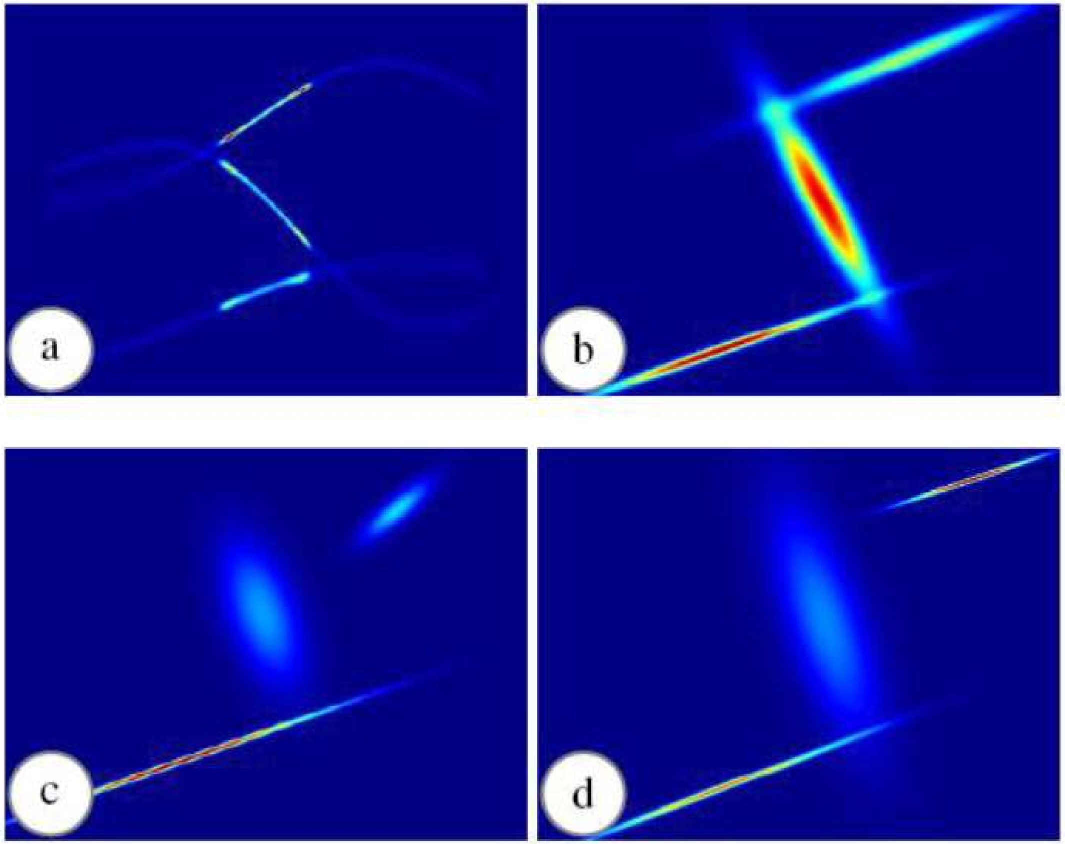
Applied to real-world example



Multiple runs...

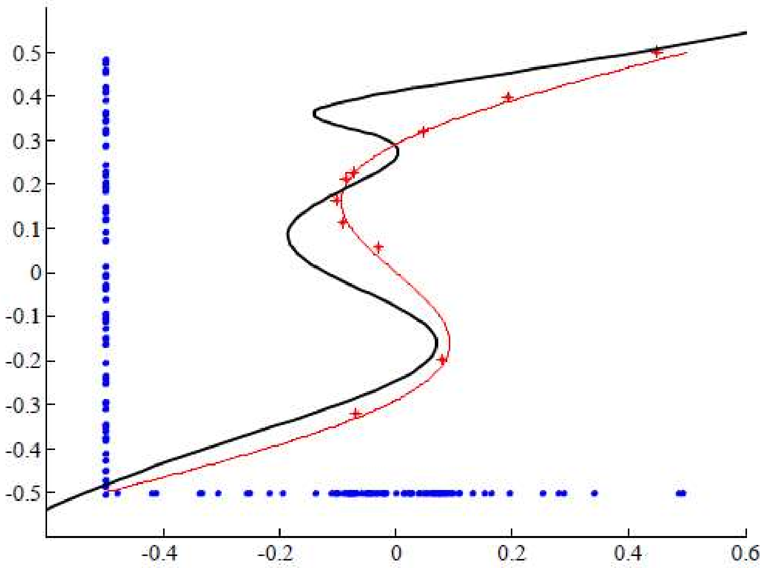


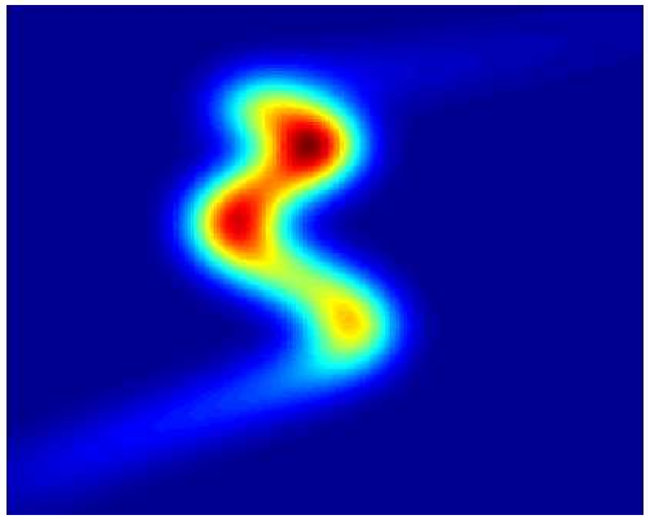
Other methods



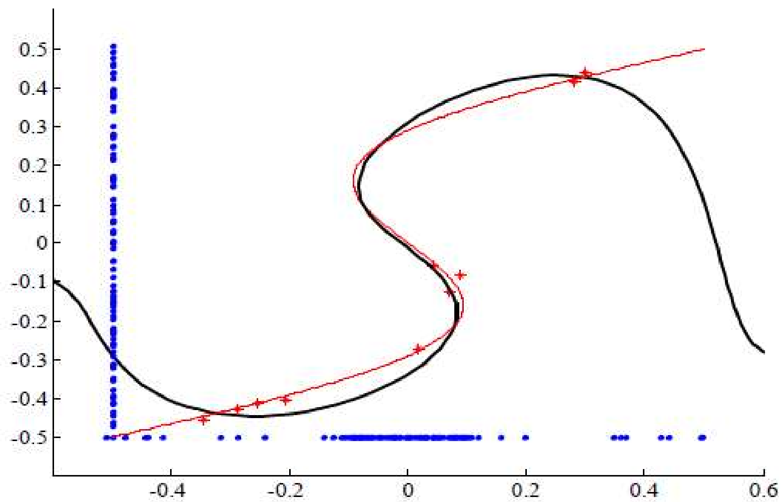
a. Mixture of experts, 15J
b. GMM, 15J
c. GMM, 8J
d. GMM, 8J, 104M

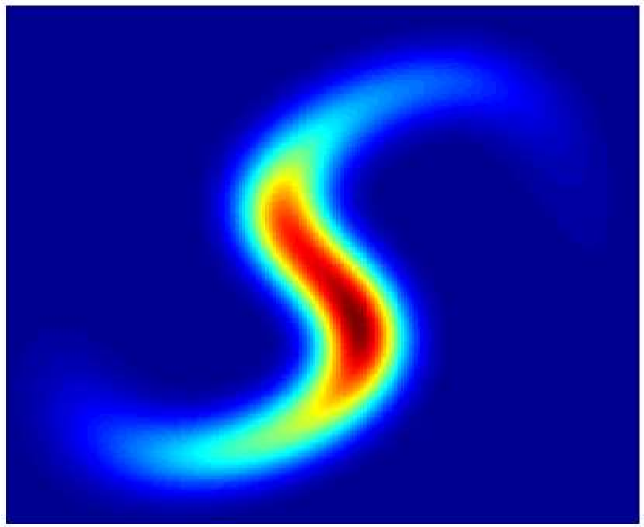
Failure modes



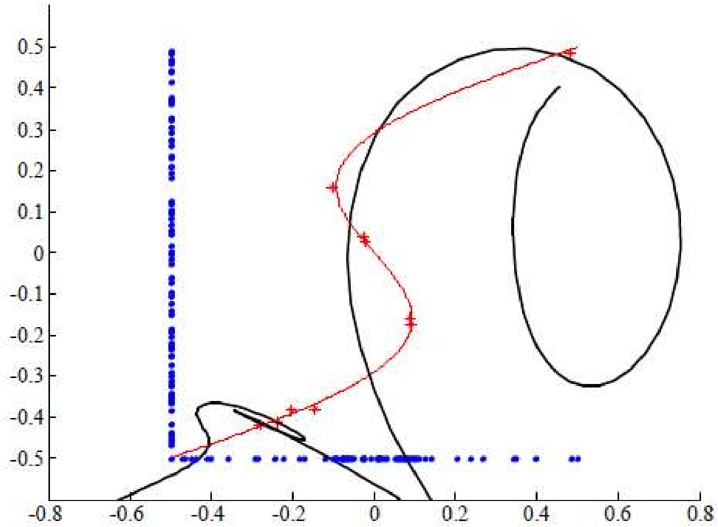


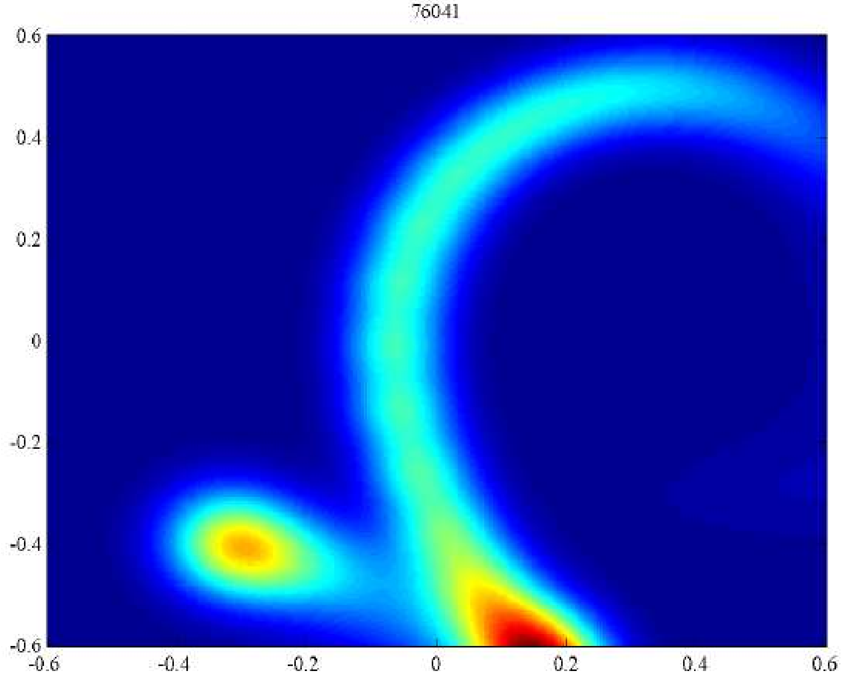
Bad convergence



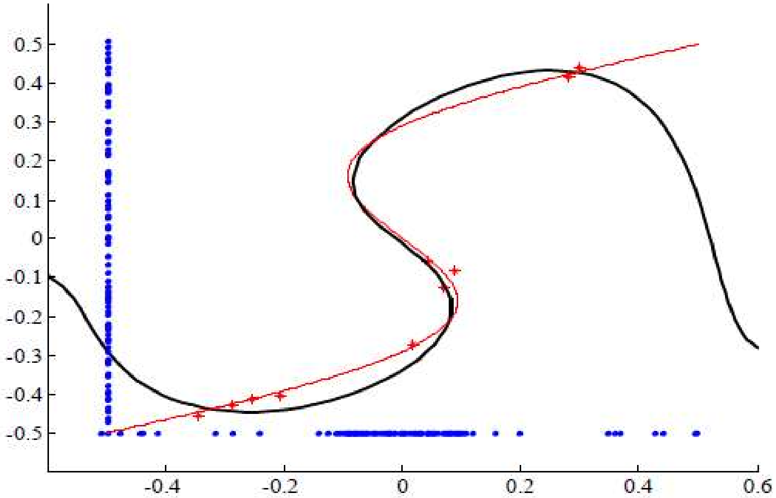


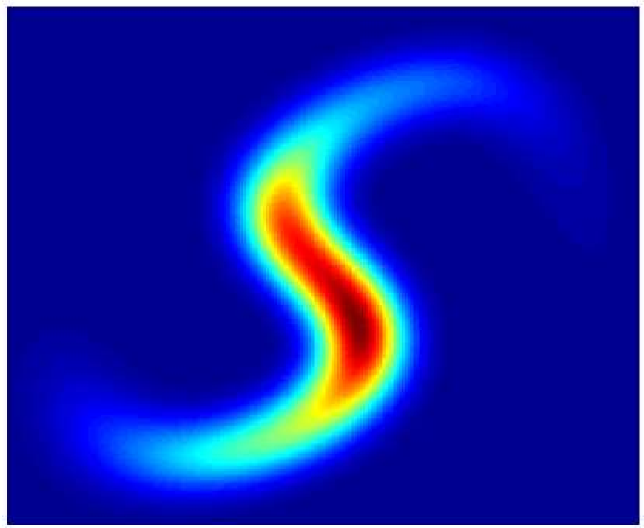
Failure modes





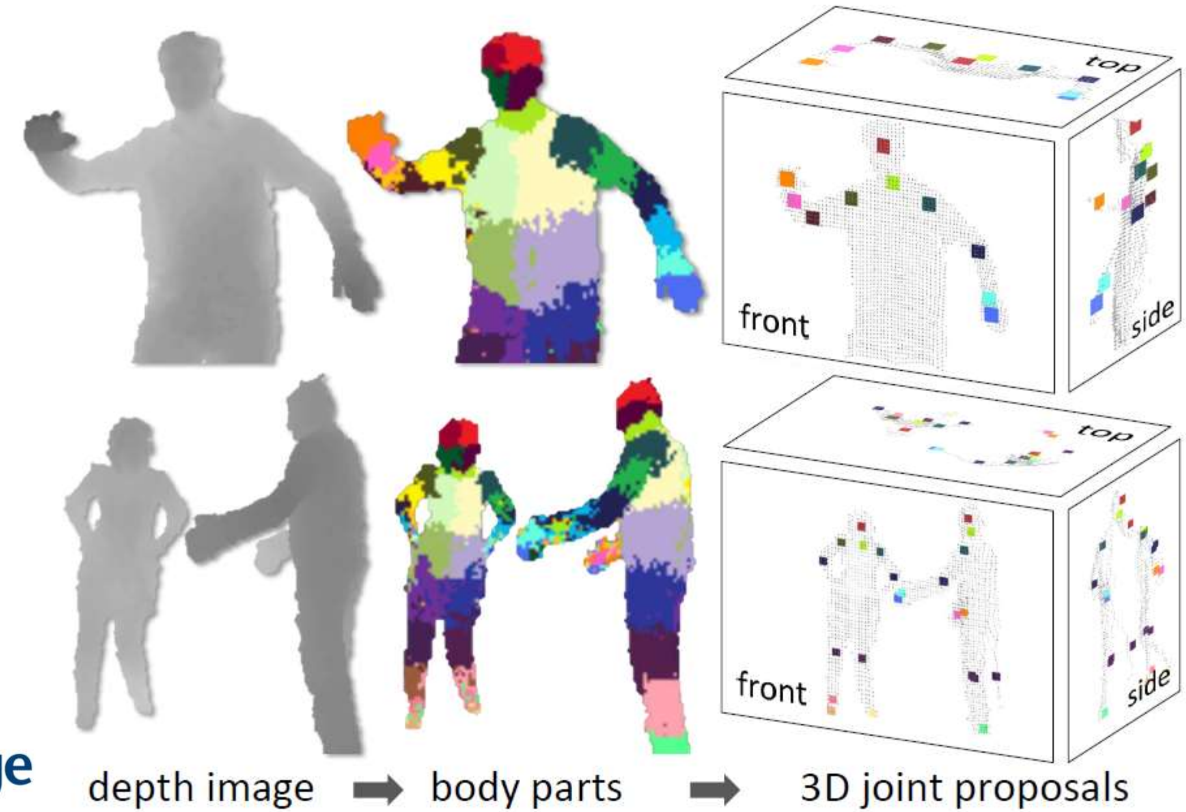
Bad initialization





Real-Time Human Pose Recognition in Parts from Single Depth Images (J. Shotton et al, 2011)

- Key features
  - Depth image as input
  - Real-time by Random Forest, and Part-based



Progressive Search Space Reduction for Human Pose Estimation (Ferrari et al, 2008)