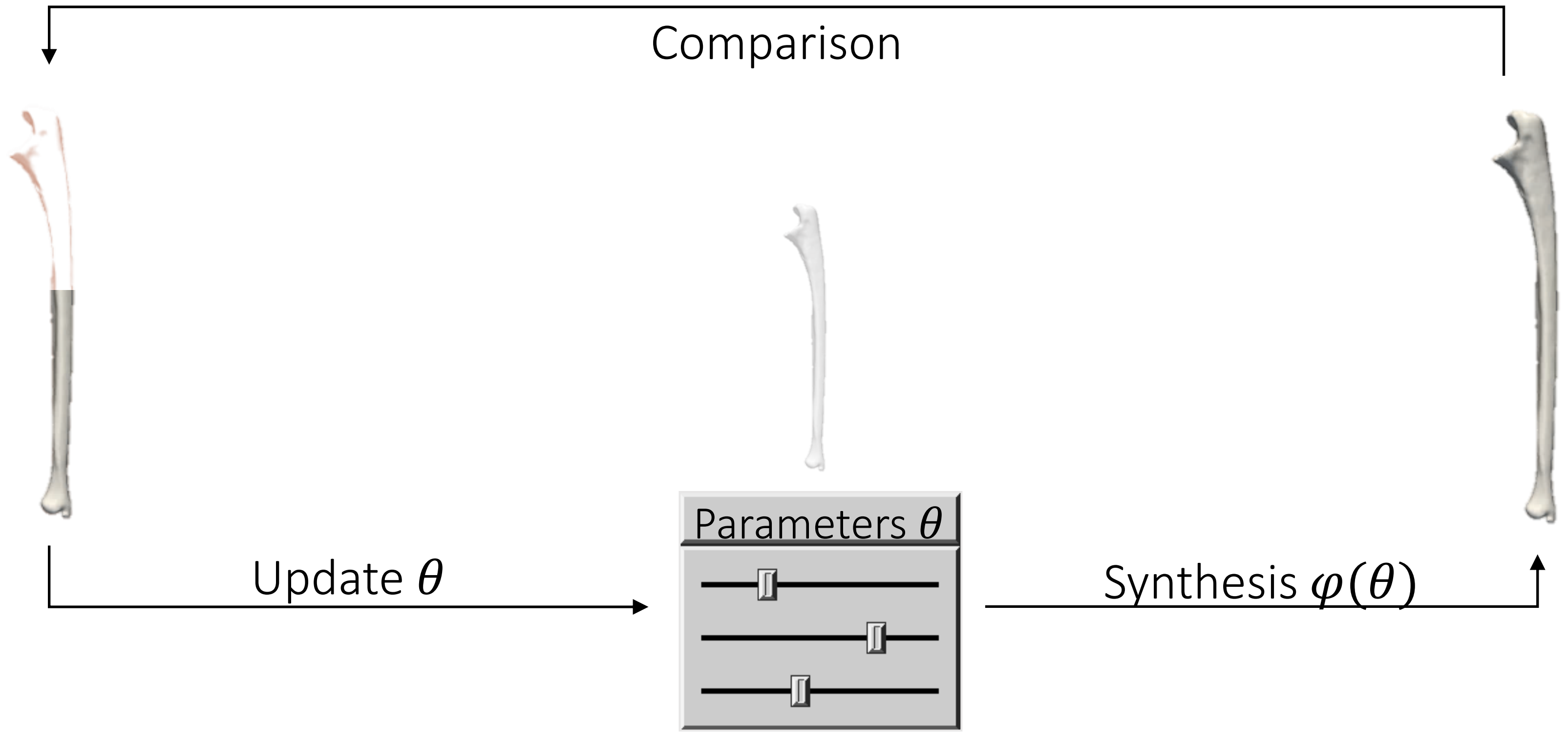
Probabilistic Fitting

Marcel Lüthi,

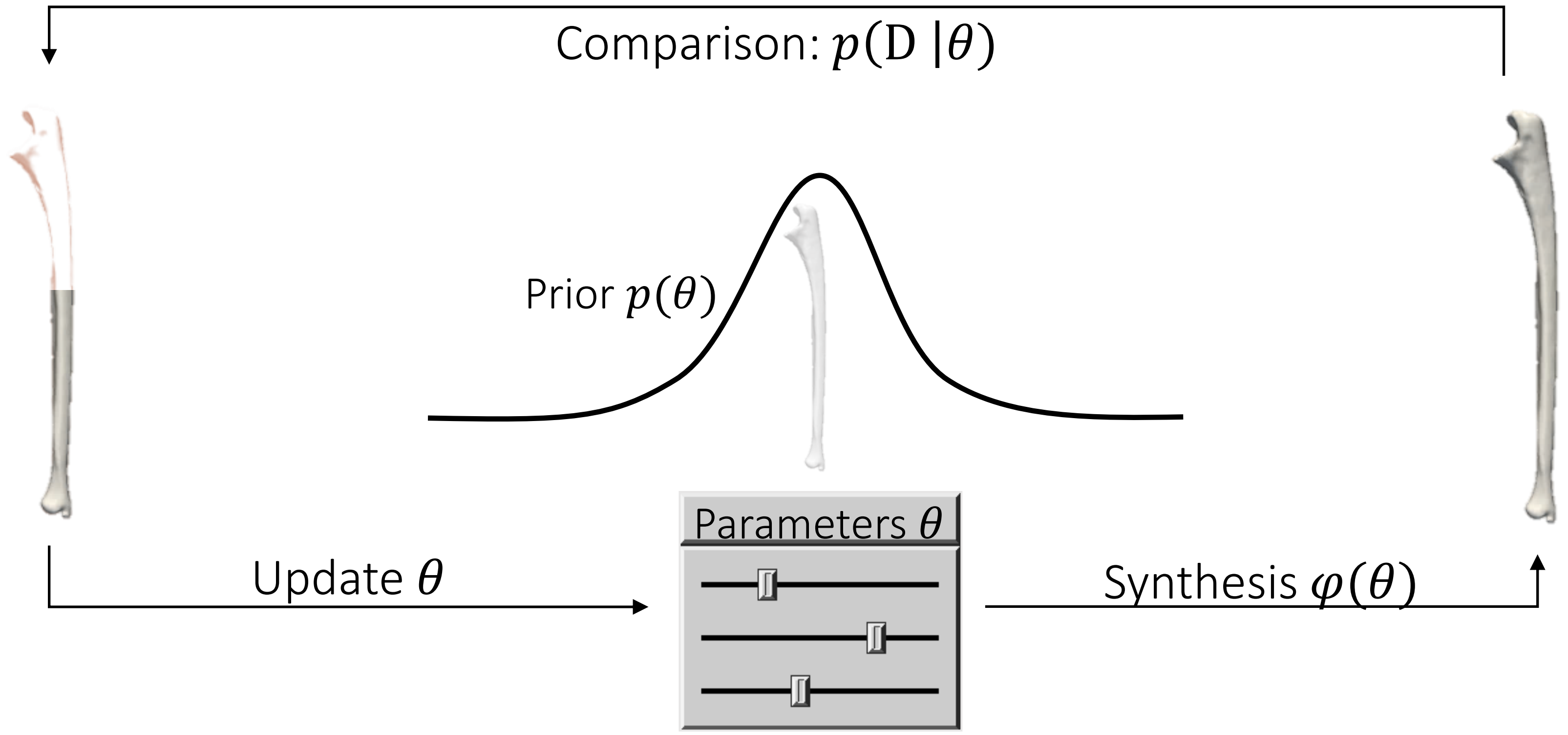
University of Basel

Analysis by Synthesis - Idea



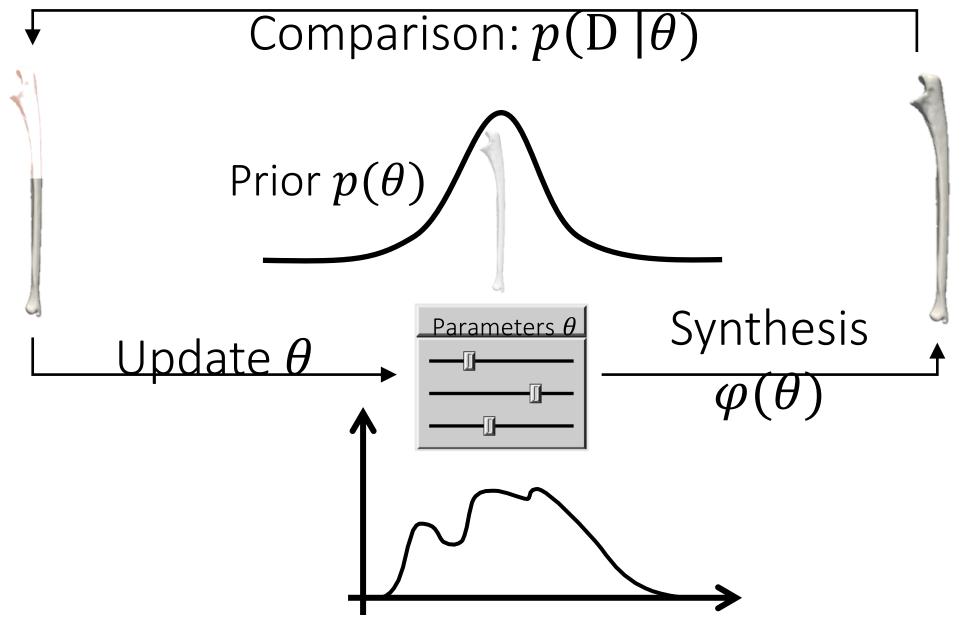
Belief: Understanding means being able to synthesize it

Analysis by Synthesis – Modelling problem


Comparison: $p(\mathrm{D}\ |\theta)$
Prior $p(\theta)$
Parameters θ
Update θ
Synthesis $\varphi(\theta)$

Modelling problem: What are $p(\theta)$ and $p(D\ |\theta)$

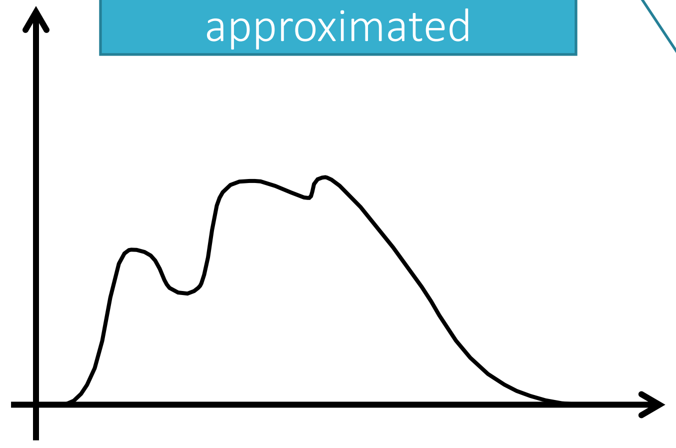
Analysis by synthesis – Conceptual problem



Updating beliefs through Bayesian inference

$$p(\theta|\mathrm{D}) = \frac{p(\theta)p(\mathrm{D}|\theta)}{\int p(\theta)p(D|\theta)d\theta}$$

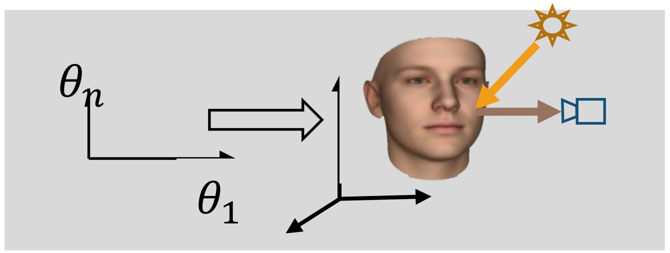
Analysis by synthesis – Computational problem



Can only be approximated

$$p(\theta|\mathrm{D}) = \frac{p(\theta)p(\mathrm{D}|\theta)}{\int p(\theta)p(D|\theta)d\theta}$$

Usually non-linear and expensive to evaluate



High-Dimensional integral

$$\int\int \dots \int p(\theta_1, \dots, \theta_n)\, p(D|\theta_1, \dots, \theta_n) d\theta_1 \dots \theta_n$$

Outline

- Basic idea: Sampling methods and MCMC

- The Metropolis-Hastings algorithm
  - The Metropolis algorithm
  - Implementing the Metropolis algorithm
  - The Metropolis-Hastings algorithm

- Example: 3D Landmark fitting

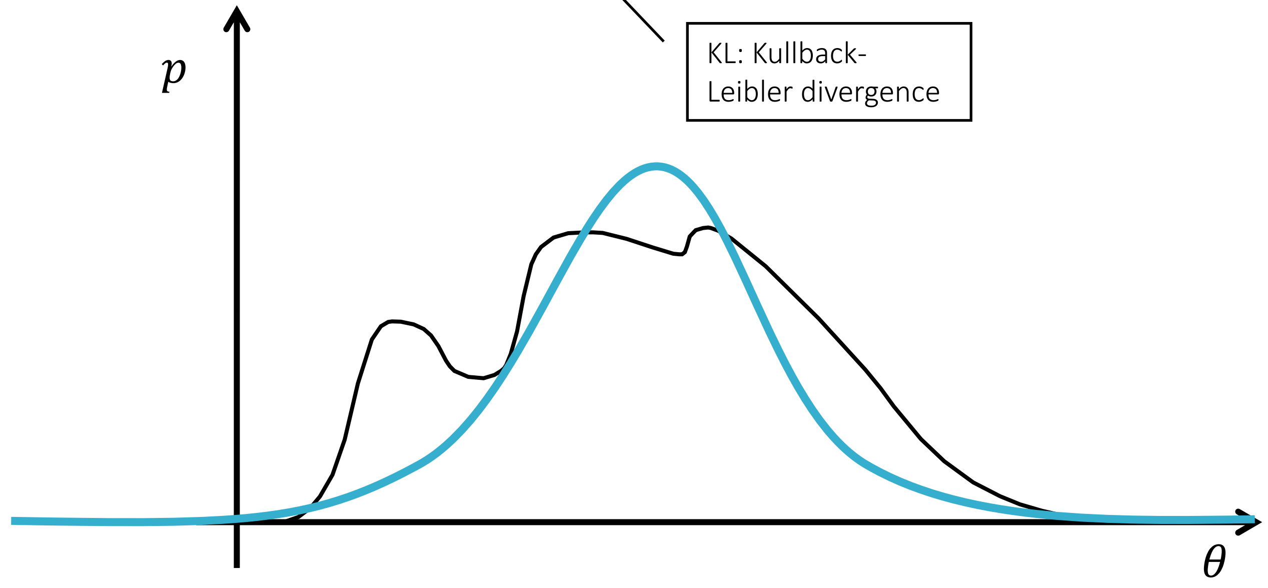
Approximate Bayesian Inference

Variational methods

- Function approximation $q(\theta)$

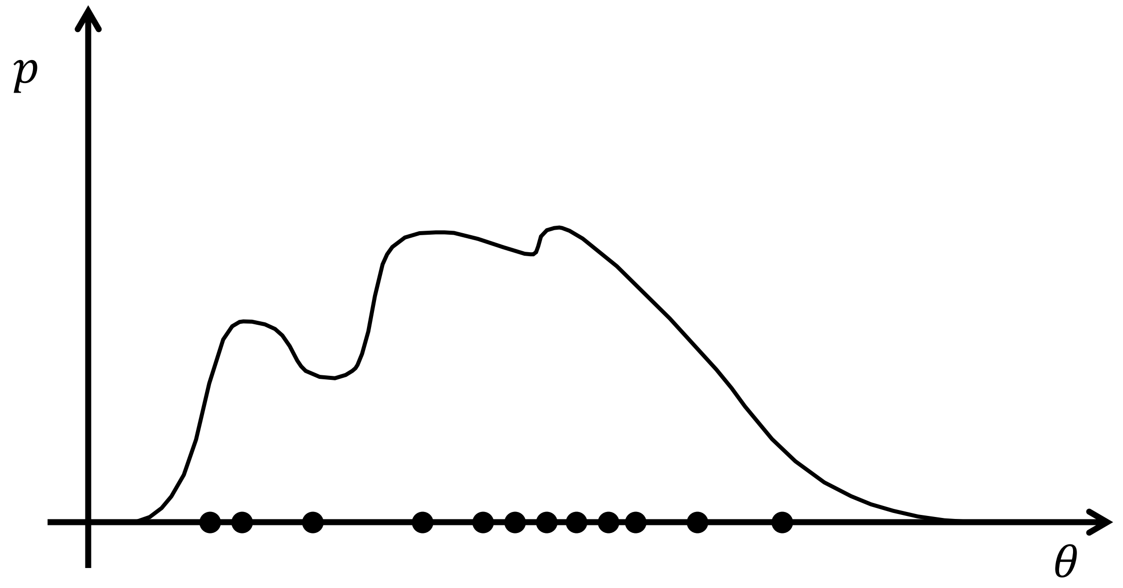
$$\arg\max_q \mathrm{KL}(q(\theta)|p(\theta|D))$$

KL: Kullback-Leibler divergence



Sampling methods

- Numeric approximations through simulation



Sampling Methods

- Simulate a distribution p through random samples x_i
- Evaluate expectation (of some function f of random variable X)

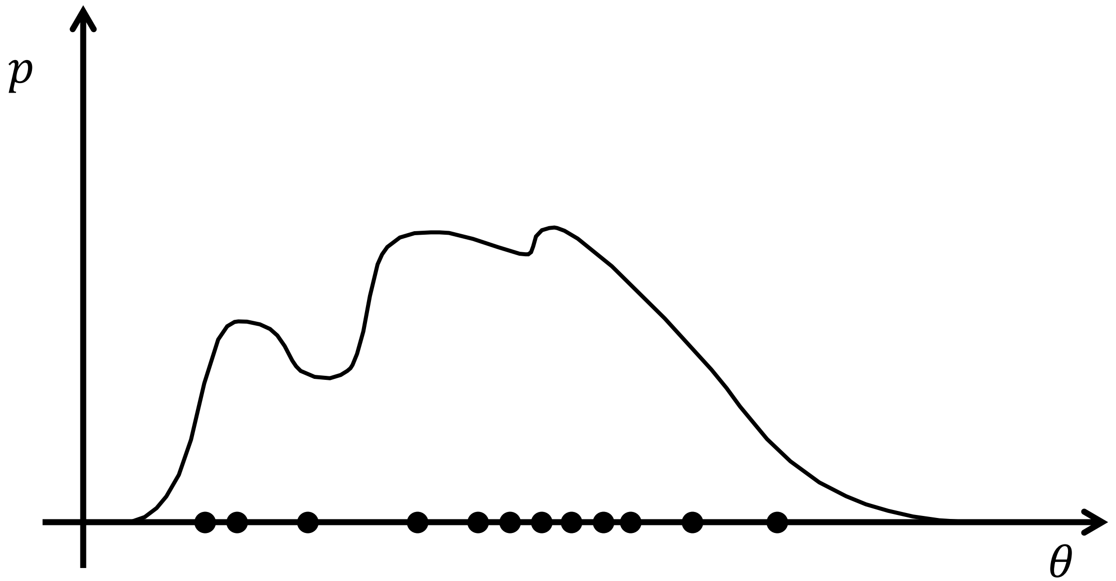
$$E[f(X)] = \int f(x)p(x)dx$$

$$E[f(X)] \approx \hat{f} = \frac{1}{N}\sum_{i}^{N} f(x_i), \qquad x_i \sim p(x)$$

$$V[\hat{f}(X)] \sim O\left(\frac{1}{N}\right)$$

This is difficult!

- *"Independent" of dimensionality of X*
- *More samples increase accuracy*



Sampling from a Distribution

```
Random.nextDouble()
Random.nextGaussian()
```

- Easy for standard distributions … is it?
  - Uniform
  - Gaussian
- How to sample from more complex distributions?
  - Beta, Exponential, Chi square, Gamma, …
  - Posteriors are very often not in a “nice” standard text book form

- *We need to sample from an unknown posterior with only unnormalized, expensive point-wise evaluation* ☹

Markov Chain Monte Carlo

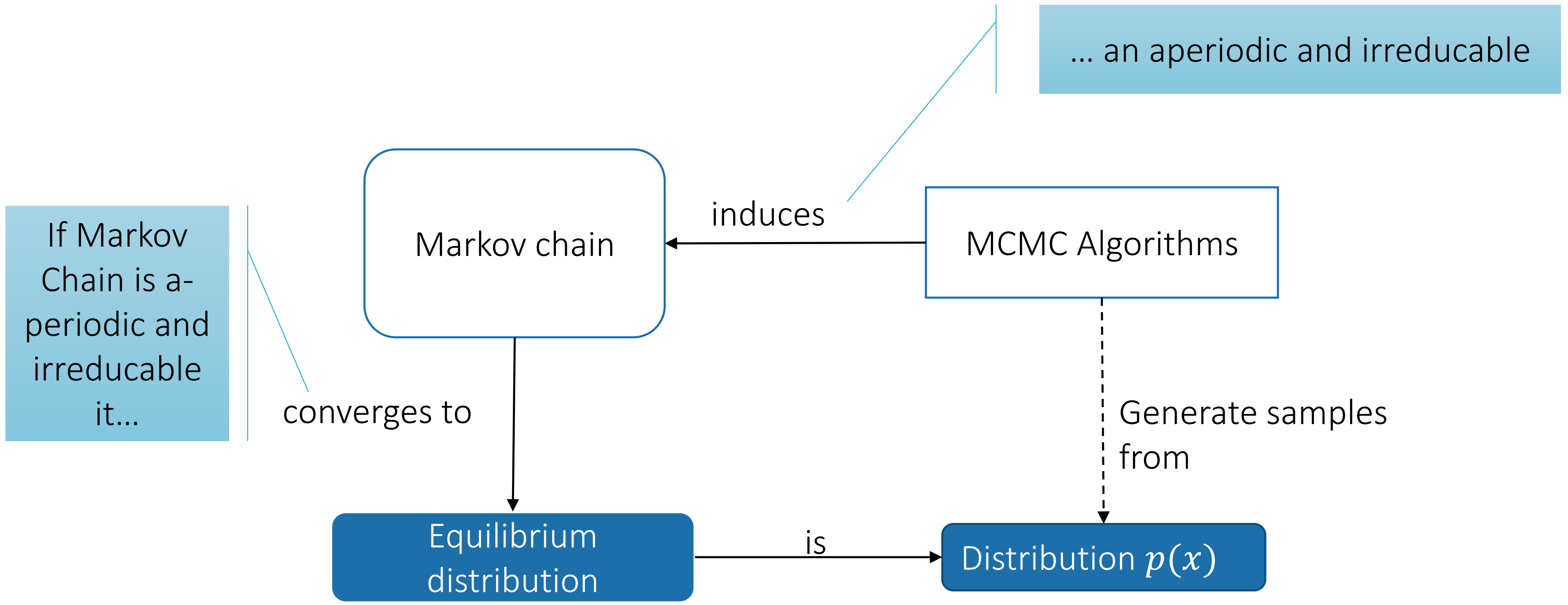
Markov Chain Monte Carlo Methods (MCMC)

Idea: Design a *Markov Chain* such that samples x obey the target distribution p

Concept: *"Use an already existing sample to produce the next one"*

- Many successful practical applications
  - Proven: developed in the 1950/1970ies (Metropolis/Hastings)
- Direct mapping of computing power to approximation accuracy

MCMC: An ingenious mathematical construction



No need to understand this now: more details follow!

The Metropolis Algorithm

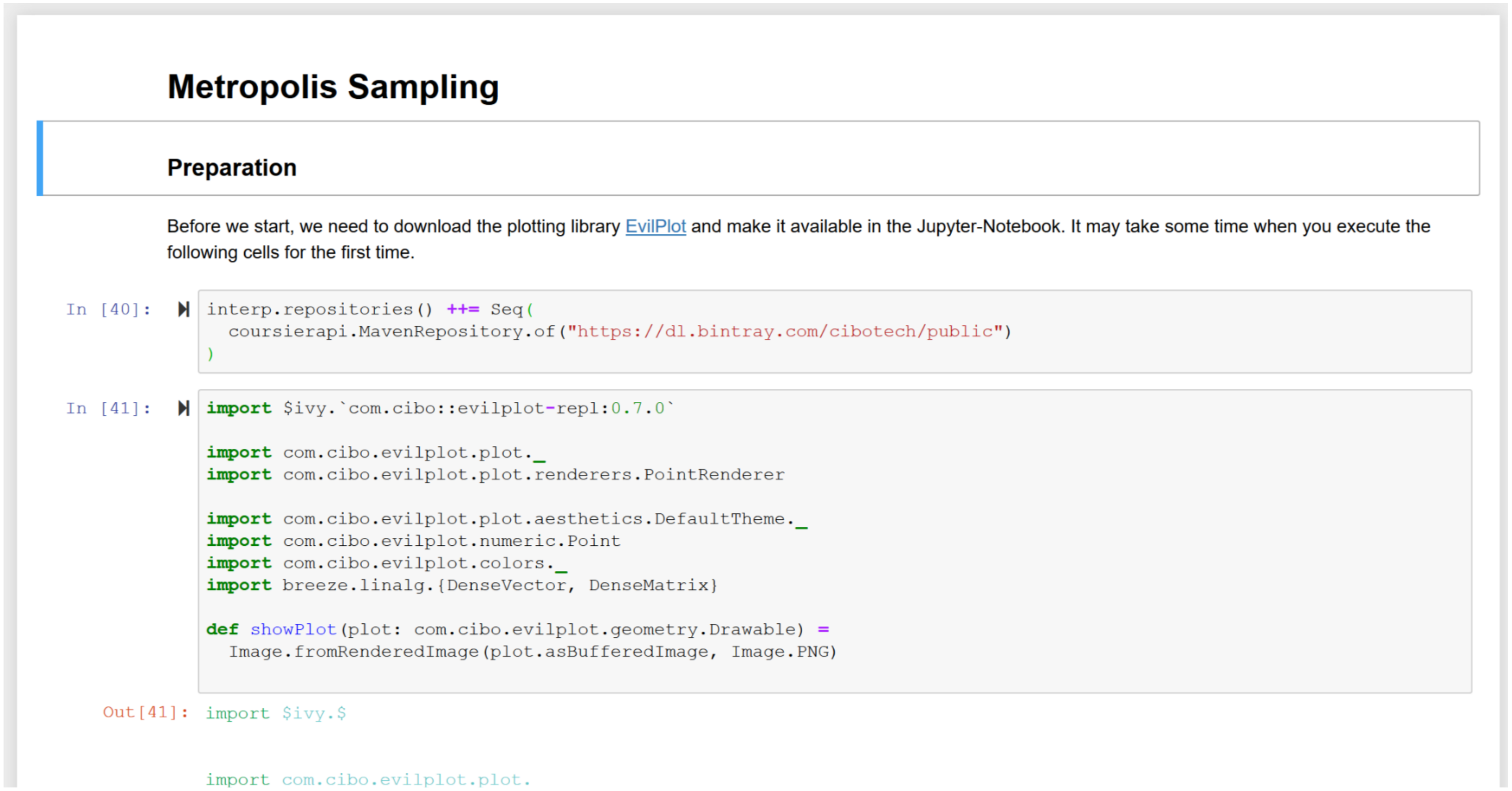
Requirements:

- Proposal distribution $Q(\boldsymbol{x}'|\boldsymbol{x})$ – *must generate samples, symmetric*
- Target distribution $P(\boldsymbol{x})$ – *with point-wise evaluation*

Result:

- Stream of samples approximately from $P(\boldsymbol{x})$

- Initialize with sample $\boldsymbol{x}$
- Generate next sample, with current sample $\boldsymbol{x}$
  1. Draw a sample $\boldsymbol{x}'$ from $Q(\boldsymbol{x}'|\boldsymbol{x})$ ("proposal")
  2. With *probability* $\alpha = \min\left\{\frac{P(\boldsymbol{x}')}{P(\boldsymbol{x})}, 1\right\}$ accept $\boldsymbol{x}'$ as new state $\boldsymbol{x}$
  3. Emit current state $\boldsymbol{x}$ as sample



Metropolis Sampling

Preparation

Before we start, we need to download the plotting library EvilPlot and make it available in the Jupyter-Notebook. It may take some time when you execute the following cells for the first time.

```
In [40]:  interp.repositories() ++= Seq(
            coursierapi.MavenRepository.of("https://dl.bintray.com/cibotech/public")
          )
```

```
In [41]:  import $ivy.`com.cibo::evilplot-repl:0.7.0`

          import com.cibo.evilplot.plot._
          import com.cibo.evilplot.plot.renderers.PointRenderer

          import com.cibo.evilplot.plot.aesthetics.DefaultTheme._
          import com.cibo.evilplot.numeric.Point
          import com.cibo.evilplot.colors._
          import breeze.linalg.{DenseVector, DenseMatrix}

          def showPlot(plot: com.cibo.evilplot.geometry.Drawable) =
            Image.fromRenderedImage(plot.asBufferedImage, Image.PNG)
```

```
Out[41]:  import $ivy.$


          import com.cibo.evilplot.plot.
```

Jupyter-Notebook – Metropolis-Hastings.ipynb

The Metropolis-Hastings Algorithm

- Initialize with sample $\boldsymbol{x}$
- Generate next sample, with current sample $\boldsymbol{x}$
  1. Draw a sample $\boldsymbol{x}'$ from $Q(\boldsymbol{x}'|\boldsymbol{x})$ ("proposal")
  2. With *probability* $\alpha = \min\left\{\frac{P(x')}{P(x)}\frac{Q(x|x')}{Q(x'|x)}, 1\right\}$ accept $\boldsymbol{x}'$ as new state $\boldsymbol{x}$
  3. Emit current state $\boldsymbol{x}$ as sample

- Generalization of Metropolis algorithm to asymmetric Proposal distribution

$$Q(\boldsymbol{x}'|\boldsymbol{x}) \neq Q(\boldsymbol{x}|\boldsymbol{x}')$$
$$Q(\boldsymbol{x}'|\boldsymbol{x}) > 0 \Leftrightarrow Q(\boldsymbol{x}|\boldsymbol{x}') > 0$$

Properties

- **Approximation:** Samples $x_1, x_2, \ldots$ approximate $P(x)$

  Unbiased but correlated (not *i.i.d.*)

- **Normalization:** $P(x)$ does not need to be normalized

  Algorithm only considers ratios $P(x')/P(x)$

- **Dependent Proposals:** $Q(x'|x)$ depends on current sample x

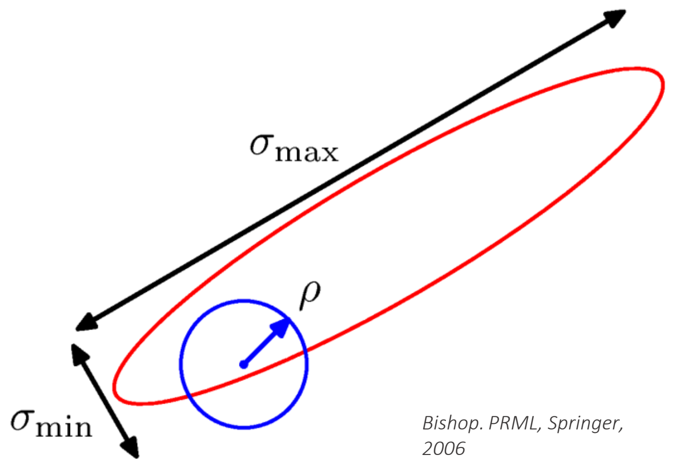
  Algorithm adapts to target with simple 1-step memory

Metropolis - Hastings: Limitations

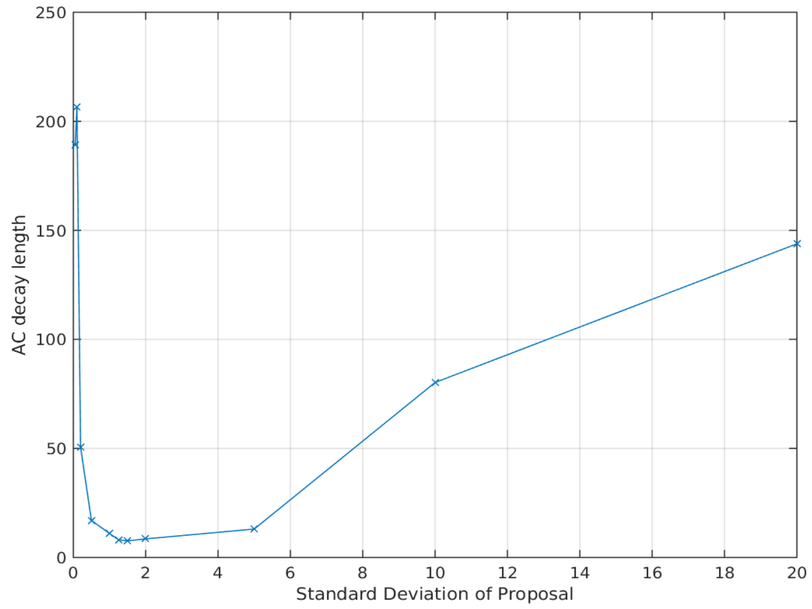
- Highly correlated targets

  Proposal should match target to avoid too many rejections

- Serial correlation
  - Results from rejection and too small stepping
  - Subsampling



Bishop. PRML, Springer, 2006



Propose-and-Verify Algorithm

- Metropolis algorithm formalizes: *propose-and-verify*
- *Steps are completely independent.*

Propose

Draw a sample x' from $Q(x'|x)$

Verify

With *probability* $\alpha = \min\left\{\frac{P(x')}{P(x)}\frac{Q(x|x')}{Q(x'|x)}, 1\right\}$ accept $\boldsymbol{x}'$ as new sample

MH as Propose and Verify

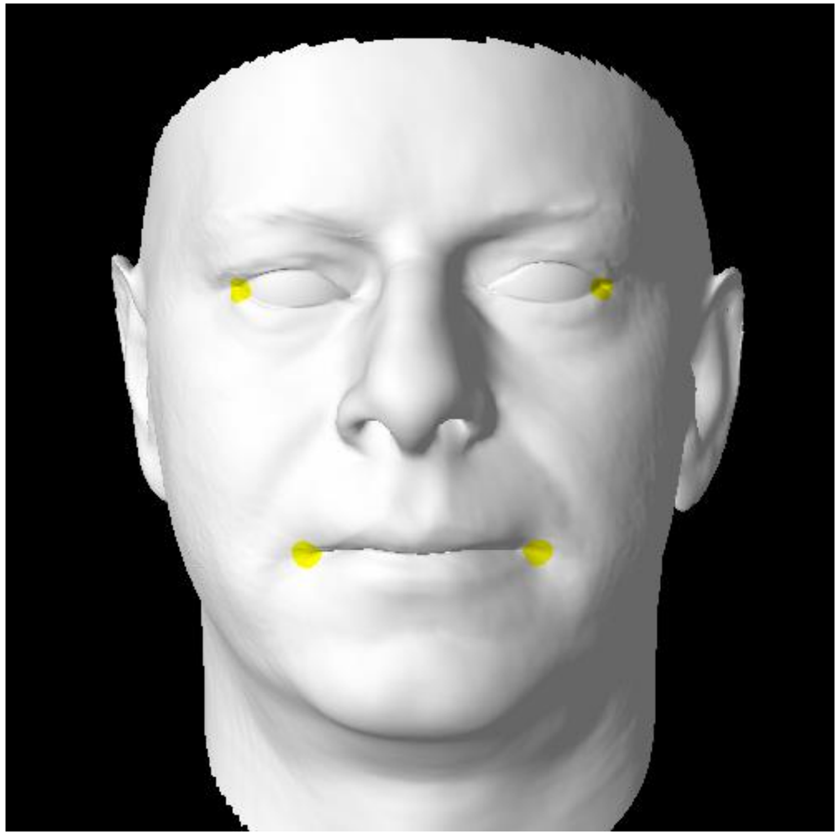
- Decouples the steps of finding the solution from validating a solution
- Natural to integrate uncertain proposals Q
  (e.g. automatically detected landmarks, ...)
- Possibility to include "local optimization" (e.g. a ICP or ASM updates, gradient step, …) as proposal

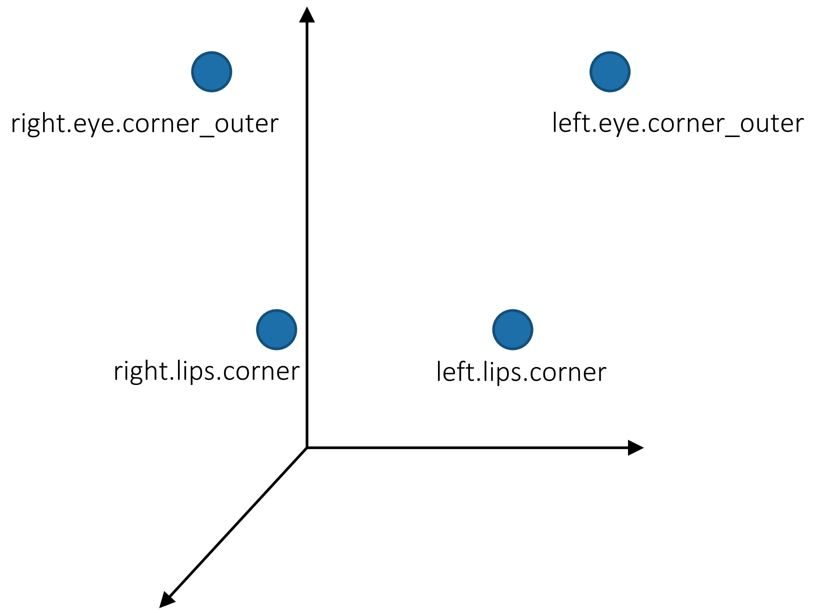
Anything more "informed" than random walk should improve convergence.

Fitting 3D Landmarks

3D Alignment with Shape and Pose

3D Fitting Example





3D Fitting Setup

***Goal**: Find posterior distribution for arbitrary pose and shape*

Shape transformation

$$\varphi_s[\alpha] = \mu(x) + \sum_{i=1}^{r} \alpha_i \sqrt{\lambda_i} \Phi_i(x)$$

Rigid transformation

- 3 angles (pitch, yaw, roll) φ, ψ, ϑ
- Translation $t = (t_x, t_y, t_z)$

$$\varphi_R[\varphi, \psi, \vartheta, t] = R_\vartheta R_\psi R_\varphi(\boldsymbol{x}) + t$$

Full transformation

$$\varphi[\theta](x) = (\varphi_R \circ \varphi_S)[\theta](x)$$

Observations

- Observed positions $l_T^1, \dots, l_T^n$
- Correspondence: $l_R^1, \dots, l_R^n$

Parameters

$$\theta = (\alpha, \varphi, \psi, \vartheta, t)$$

Posterior distribution:

$$P\left(\theta \middle| l_T^1, \dots, l_T^n\right) \propto p\left(l_T^1, \dots, l_T^R | \theta\right) P(\theta)$$

Proposals

- Gaussian random walk proposals

$$"Q(\theta'|\theta) = N(\theta'|\theta, \Sigma_\theta)"$$

- Update different parameter types block-wise
  - Shape $N(\boldsymbol{\alpha}'|\boldsymbol{\alpha}, \sigma_S^2 I_{m\times m})$
  - Rotation $N(\varphi'|\varphi, \sigma_\varphi^2),\ N\left(\psi'\middle|\psi, \sigma_\psi^2\right), N(\vartheta'|\vartheta, \sigma_\vartheta^2)$
  - Translation $N\left(\boldsymbol{t}'\middle|\boldsymbol{t}, \sigma_t^2 I_{3\times 3}\right)$

- Large mixture distributions as proposals

  - Choose proposal Q_i with probability c_i

$$Q(\theta'|\theta) = \sum c_i Q_i(\theta'|\theta)$$

3DMM Landmarks Likelihood

Simple models: **Independent Gaussians**

Observation of L landmark locations l_T^i in image

- Single *landmark position* model:

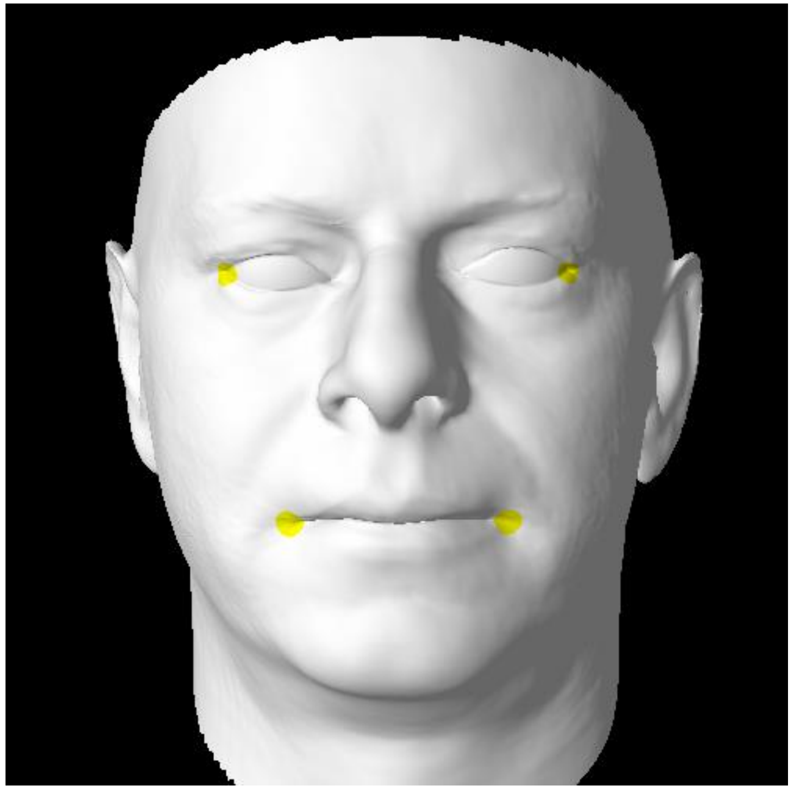
$$p(l_T|\theta, l_R) = N\left(\varphi[\theta](l_R), I_{3\times3}\sigma^2\right)$$

- *Independent* model (conditional independence):

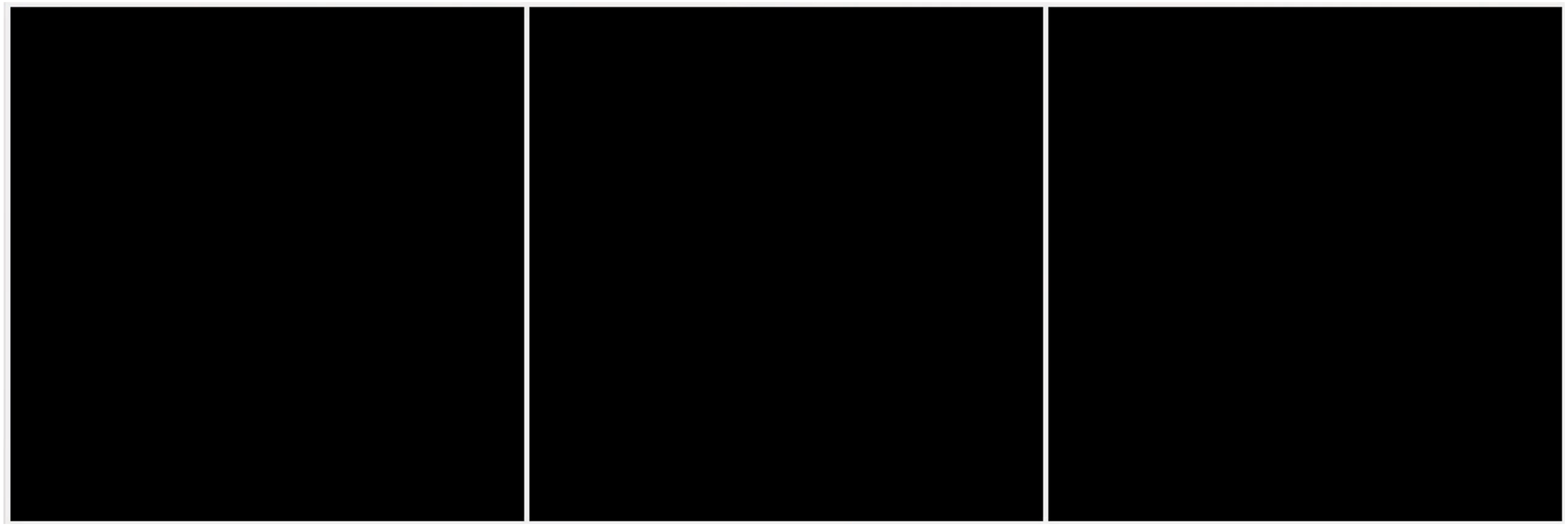
$$p\left(l_T^1, \dots, l_T^n|\theta\right) = \prod_{i=1}^{L} p_i\left(l_T^i|\theta\right)$$

3D Fit to landmarks

- Influence of landmarks uncertainty on final posterior?
  - $\sigma_{\mathrm{LM}} = 1\mathrm{mm}$
  - $\sigma_{\mathrm{LM}} = 4\mathrm{mm}$
  - $\sigma_{\mathrm{LM}} = 10\mathrm{mm}$
- Only 4 landmark observations:
  - Expect only weak shape impact
  - Should still constrain pose
- Uncertain landmarks should be looser



Posterior: Pose & Shape, 4mm



$\hat{\mu}_{\mathrm{yaw}} = 0.511$ $\hat{\mu}_{\mathrm{t_x}} = -1\ \mathrm{mm}$ $\hat{\mu}_{\alpha_1} = 0.4$

$\hat{\sigma}_{\mathrm{yaw}} = 0.073\ (4°)$ $\hat{\sigma}_{\mathrm{t_x}} = 4\ \mathrm{mm}$ $\hat{\sigma}_{\alpha_1} = 0.6$

(Estimation from samples)

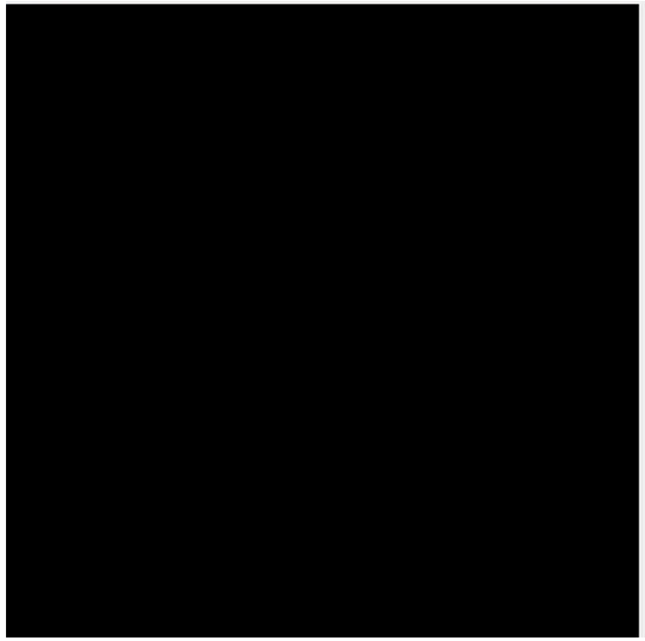
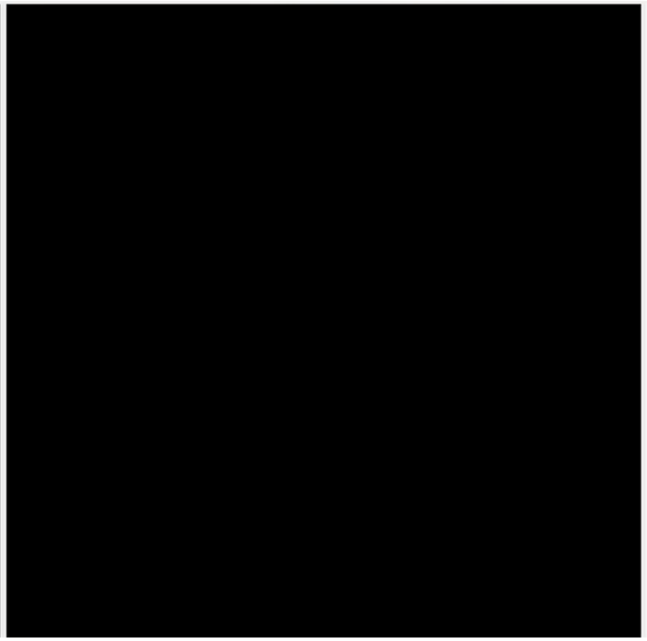
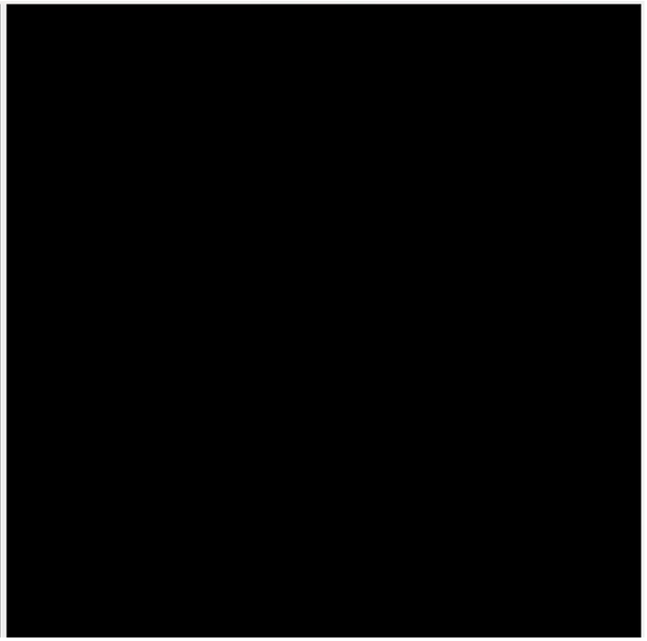
Posterior: Pose & Shape, 1mm

$\hat{\mu}_{\mathrm{yaw}} = 0.50$
$\hat{\sigma}_{\mathrm{yaw}} = 0.041\ (2.4°)$

$\hat{\mu}_{\mathrm{t_x}} = -2\ \mathrm{mm}$
$\hat{\sigma}_{\mathrm{t_x}} = 0.8\ \mathrm{mm}$

$\hat{\mu}_{\alpha_1} = 1.5$
$\hat{\sigma}_{\alpha_1} = 0.35$

Posterior: Pose & Shape, 10mm

$\hat{\mu}_{\mathrm{yaw}} = 0.49$
$\hat{\sigma}_{\mathrm{yaw}} = 0.11\ (7°)$

$\hat{\mu}_{\mathrm{t_x}} = -5\ \mathrm{mm}$
$\hat{\sigma}_{\mathrm{t_x}} = 10\ \mathrm{mm}$

$\hat{\mu}_{\alpha_1} = 0$
$\hat{\sigma}_{\alpha_1} = 0.6$

Summary: MCMC for 3D Fitting

- Probabilistic inference for fitting probabilistic models
  - Bayesian inference: posterior distribution
- Probabilistic inference is often intractable
  - Use *approximate* inference methods
- MCMC methods provide a powerful sampling framework
  - Metropolis-Hastings algorithm
    - Propose update step
    - Verify and accept with probability
- Samples converge to true distribution: More about this later!