Shape model fitting using Metropolis-Hastings

Marcel Lüthi,

University of Basel

Agenda

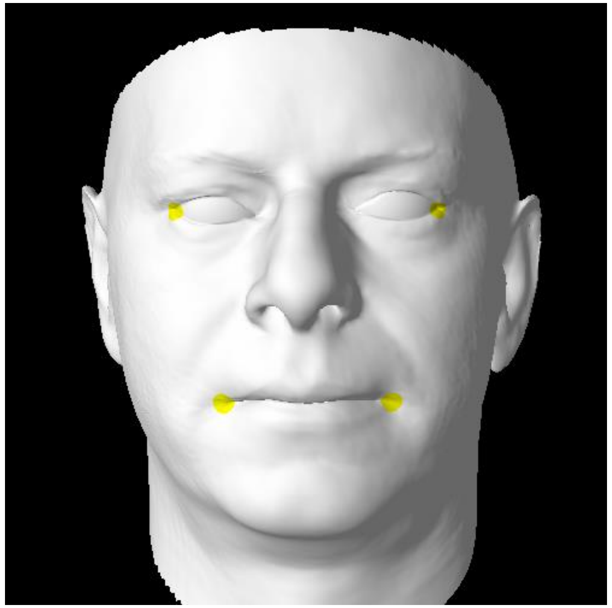
- Reminder – Metropolis-Hastings algorithm
- Case study: Landmark fitting with Metropolis-Hastings
  - General setting
  - Modelling
  - Sampling
- Other likelihood functions
  - Non-correspondence points
  - Active shape models
- Misc. Topics
  - Debugging Metropolis-Hastings sampler
  - Sequential Bayesian updating

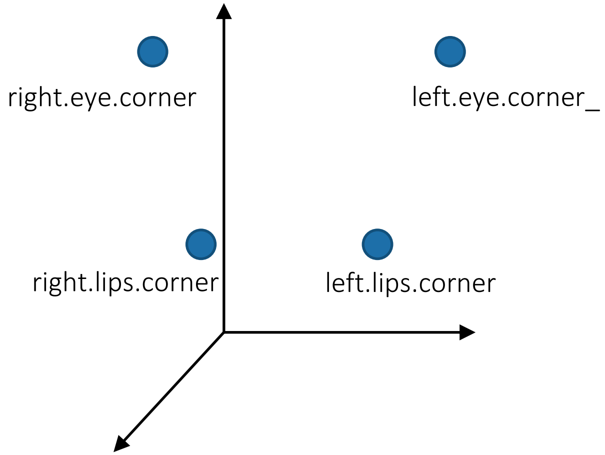
Case study: Landmark fitting with Metropolis-Hastings

Problem setting

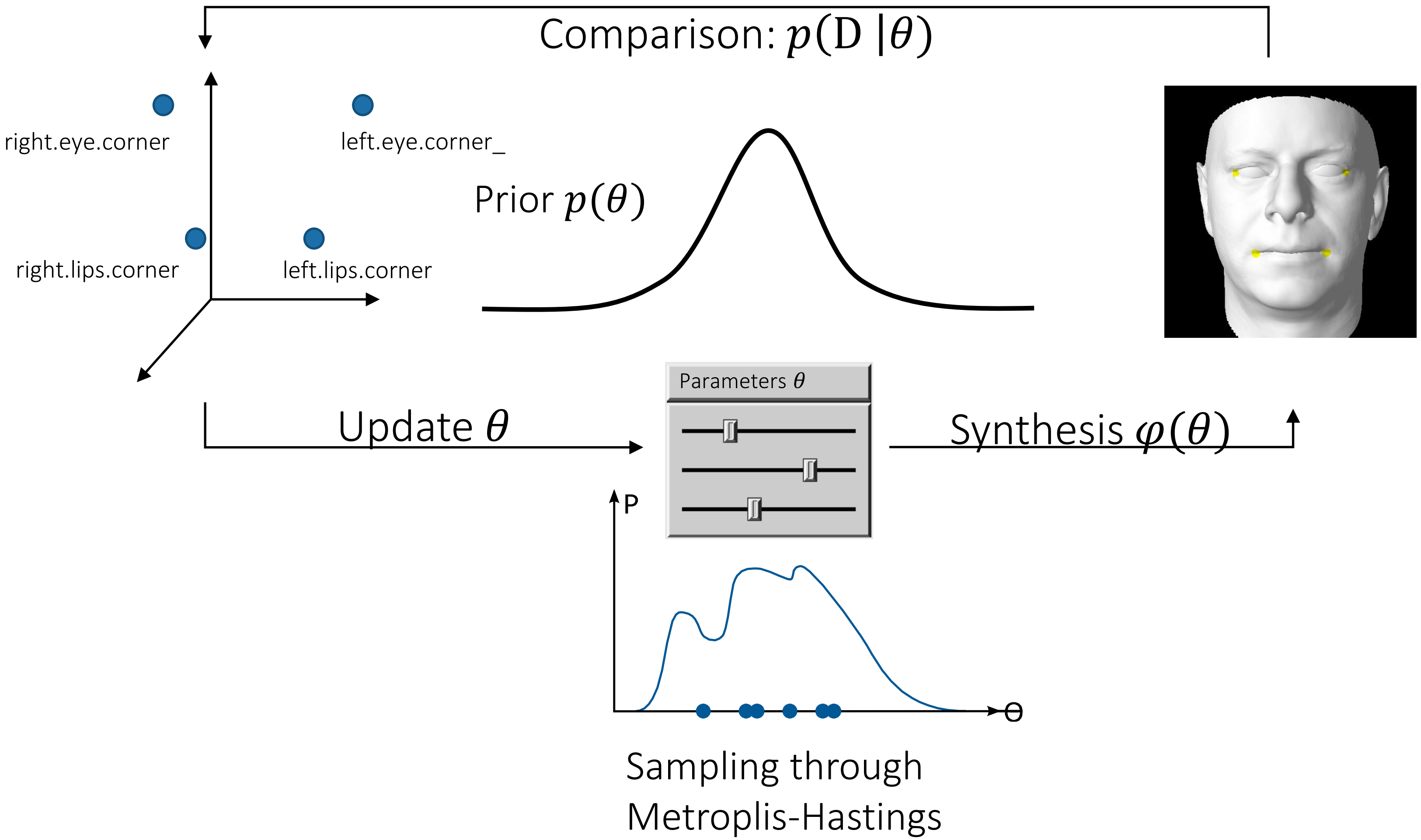
Given:
- Face model with m basis functions
- Landmarks points on model reference:
$$\mathrm{l}_1^{\mathrm{R}}, \dots, l_n^R$$
- Observed (corresponding) 3D-positions of landmark
$$l_1^T, \dots, l_n^T$$

Goal:
- Find faces matching the landmark points





Approach: Analysis by synthesis



Step 1: Defining the model parameters (our world)

- Shape-model-parameters:

$$\alpha_1, \dots, \alpha_m$$

- Pose-parameters
  - Translation:

$$t = (t_x, t_y, t_z)$$

  - Rotation: Euler angles (pitch, yaw, roll)

$$\varphi, \psi, \vartheta$$

Full parameter vector

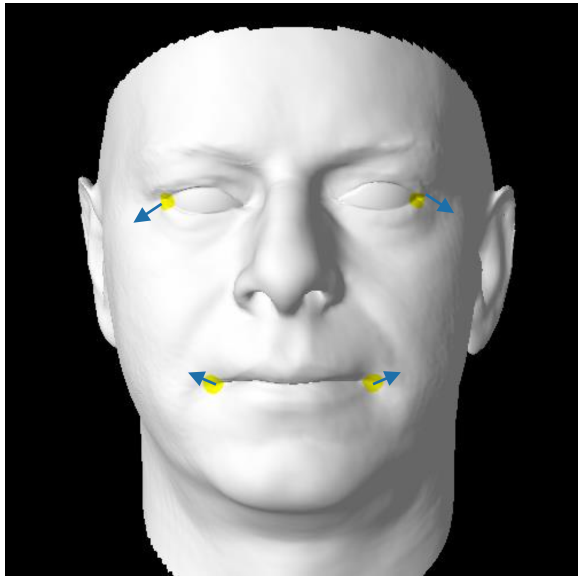
$$\theta = (\alpha, \varphi, \psi, \vartheta, t)$$

Step 2: Synthesis function

Shape-model transformation for landmark point l_i^R:

$$\varphi_S[\alpha](l_i^R) = l_i^R + \mu(l_i^R) + \sum_{i=n}^{m} \alpha_i \sqrt{\lambda_i} \phi_i(l_i^R)$$

KL-Expansion of GP Model $u \sim GP(\mu, k)$



Pose transformation:

$$\varphi_P[\varphi, \psi, \vartheta, t](l_i^R) = R_{\vartheta,\psi,\varphi}(l_i^R) + t$$

Full transformation:

$$\varphi[\theta](l_i^R) = (\varphi_p[\varphi, \psi, \vartheta, t] \circ \varphi_S[\alpha])(l_i^R)$$

Step 3: Likelihood function

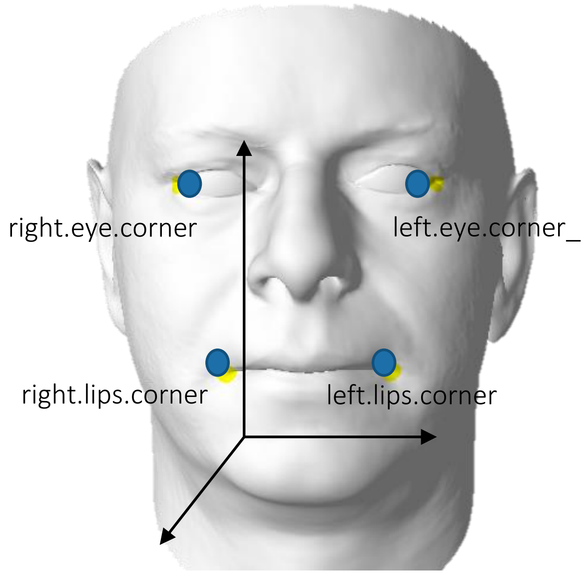
For one landmark pair (l_i^R, l_i^T):

$$p\left(l_i^T \middle| \theta, l_i^R\right) = N\left(\varphi[\theta]\left(l_i^R\right), I_{3x3}\sigma^2\right)$$

For all landmarks (assuming independence):

$$\left(l_1^T, \dots, l_n^T \middle| \theta, l_1^R, \dots, l_n^R\right) = \prod_i N\left(\varphi[\theta]\left(l_i^R\right), I_{2x2}\sigma^2\right)$$

Landmarks match target position up to zero-mean Gaussian noise.



Step 4: Prior distributions

From KL-Expansion of GP Model $u \sim GP(\mu, k)$

Shape - model priors:

$$\alpha_i \sim N(0,1)$$

Translation prior

- Assuming model is aligned to target:

$$t_x, t_y, t_z \sim N(0,10)$$

- Otherwise: $t_x, t_y, t_z \sim U(-1000, 1000)$

Rotation prior

- Assuming model is well aligned to target and roation center is center of mass of model:

$$\varphi, \psi, \vartheta \sim N(0, 0.1)$$

- Otherwise: $\varphi, \psi, \vartheta \sim U(-\pi, \pi)$

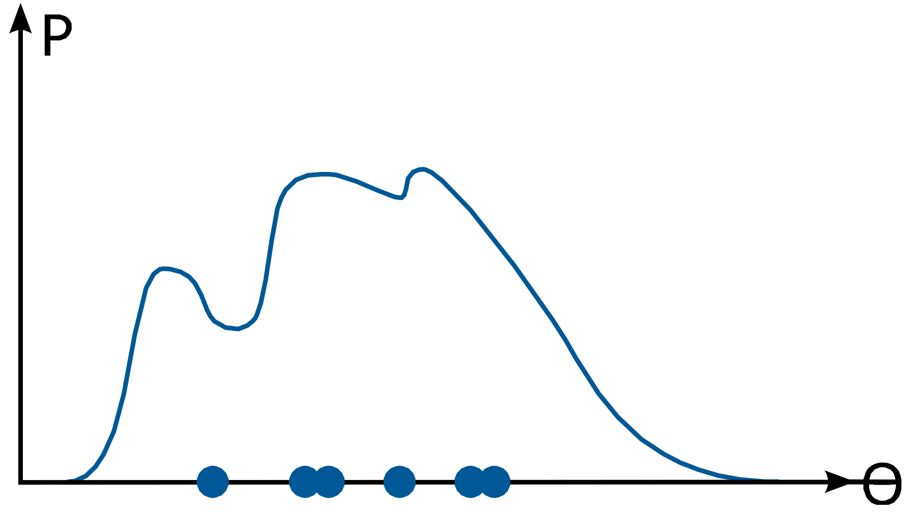
Step 5: Inference

Posterior distribution:

$$P\left(\theta \middle| l^T, l^R\right) = \frac{p\left(l^T|\theta, l^R\right)P(\theta)}{\int p(l^T|\theta, l^R)P(\theta)d\theta}$$

Intractable:

- Approximate using sampling



Step 5: Setup of Metropolis-Hastings algorithm

Proposals: Gaussian random walk proposals

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

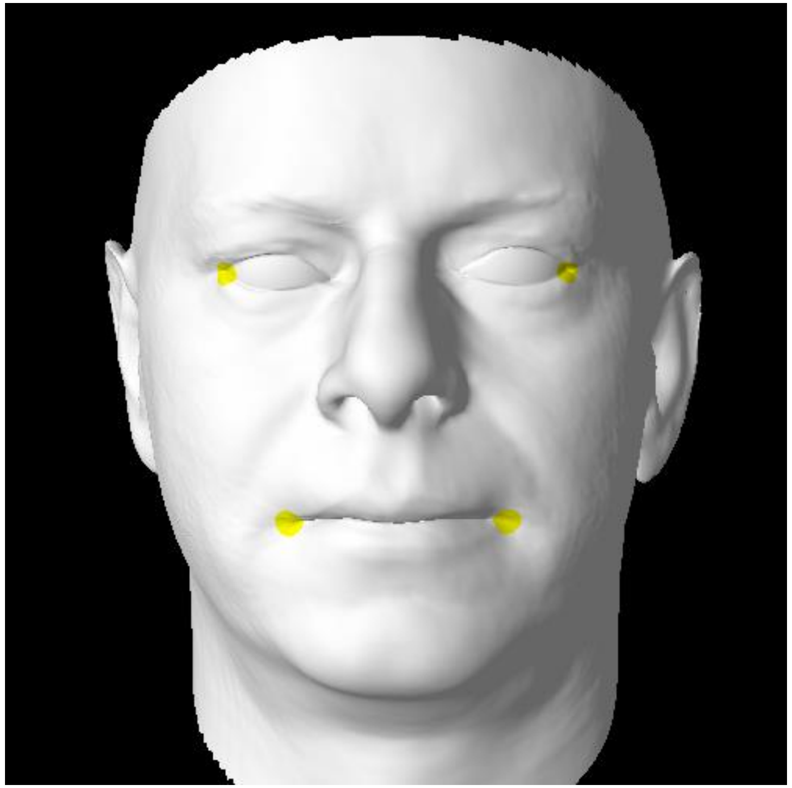
- Blockwise updates
  - Shape $N(\alpha, \sigma_S^2 I_{m\times m})$
  - Rotation $N(\varphi, \sigma_\varphi^2),\ N\left(\psi, \sigma_\psi^2\right), N(\vartheta, \sigma_\vartheta^2)$
  - Translation $N(t, \sigma_t^2 I_{3\times 3})$
- Large mixture distributions as proposals

  Choose proposal Q_i with probability c_i

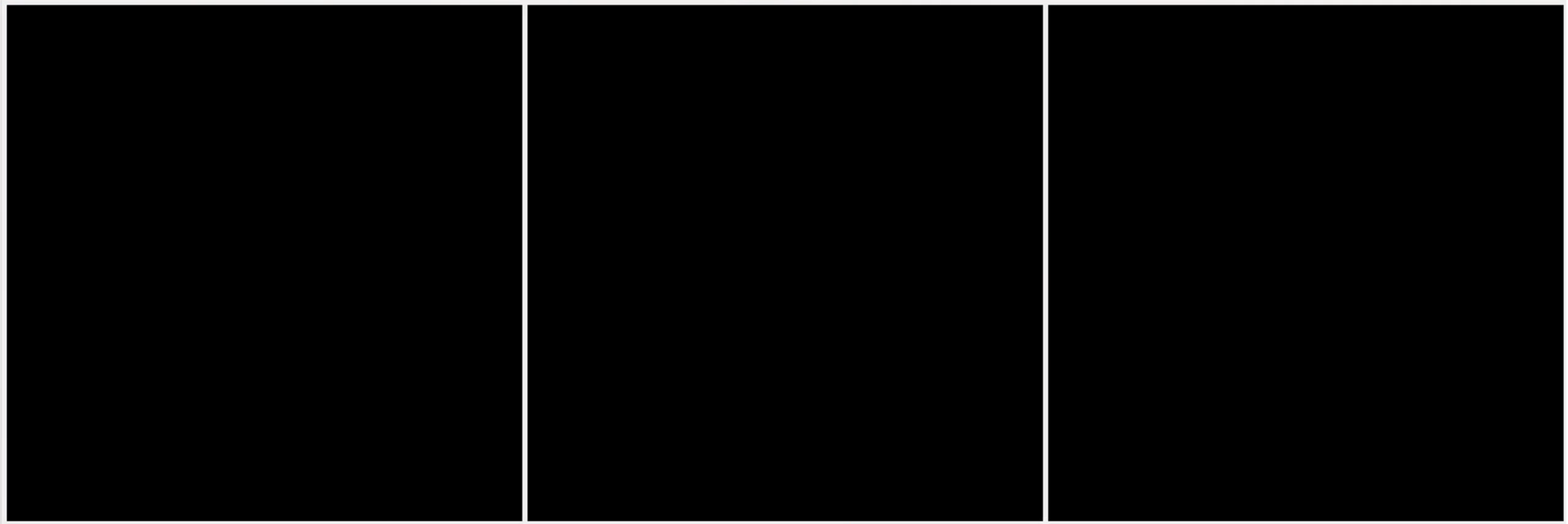
$$Q(\theta'|\theta) = \sum c_i Q_i(\theta'|\theta)$$

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



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

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

(Estimation from samples)

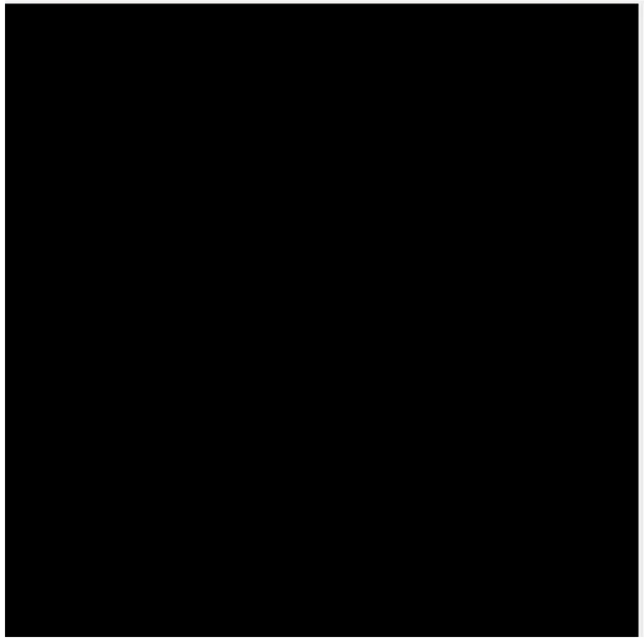
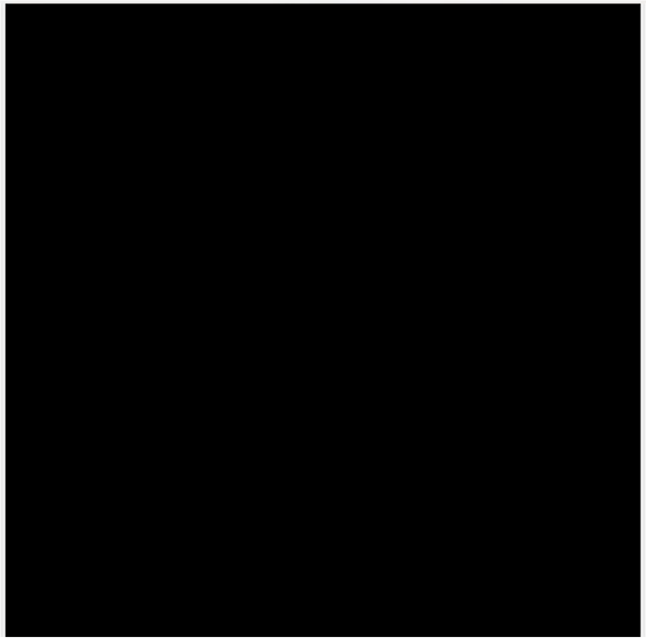
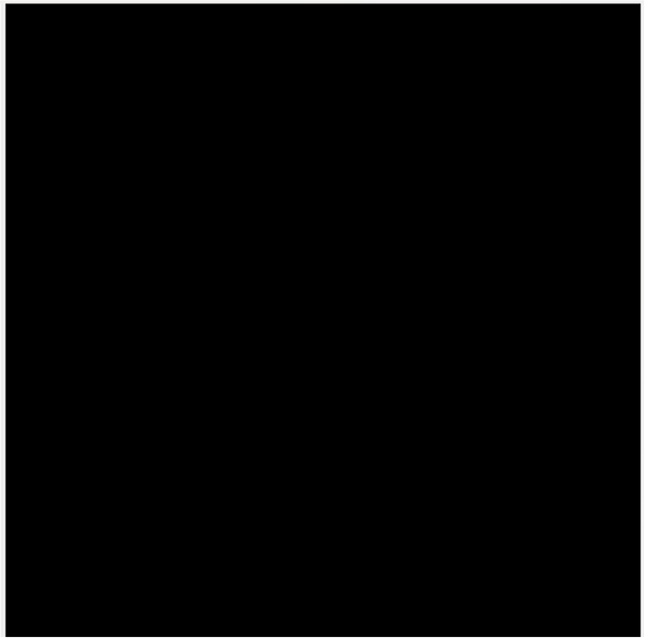
Posterior: Pose & Shape, 1mm

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

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

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

Posterior: Pose & Shape, 10mm

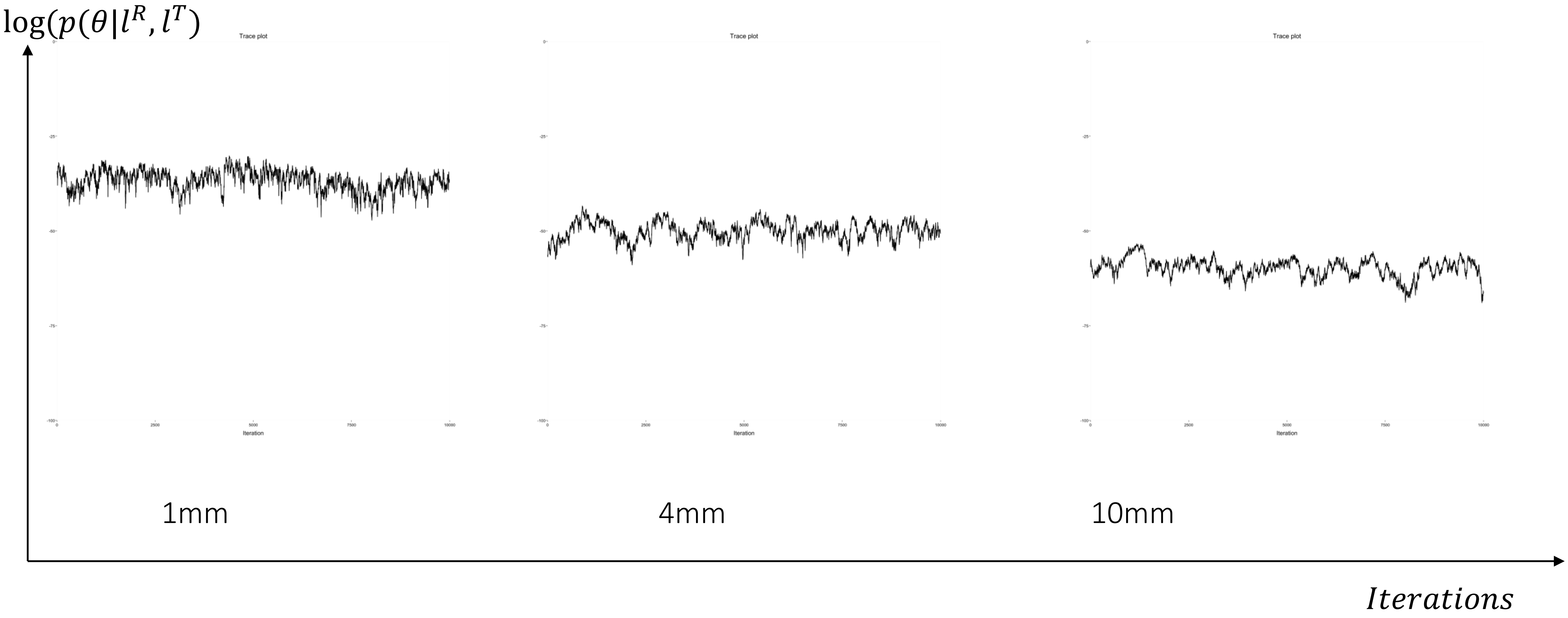




$\hat{\mu}_{\mathrm{yaw}} = 0.49$
$\hat{\sigma}_{\mathrm{yaw}} = 0.11\ (7°)$

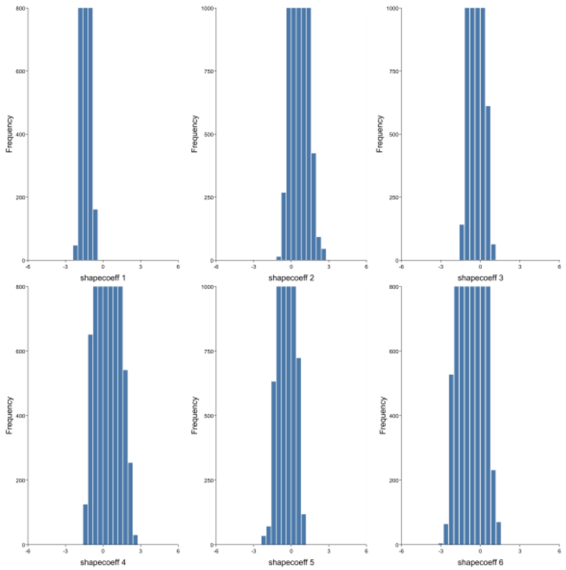
$\hat{\mu}_{\mathrm{t_x}} = -5\ \mathrm{mm}$
$\hat{\sigma}_{\mathrm{t_x}} = 10\ \mathrm{mm}$

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

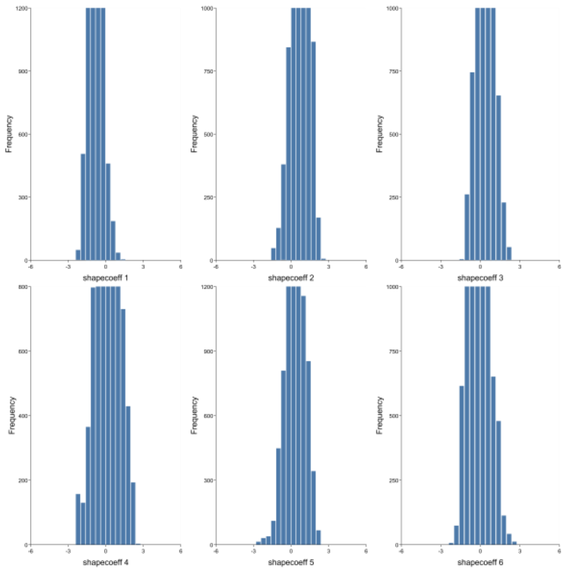
Traceplots


$\log(p(\theta|l^R, l^T)$
Trace plot
Trace plot
Trace plot
0
-25
-50
-75
-100
0
2500
5000
7500
10000
Iteration
1mm
4mm
10mm
$Iterations$

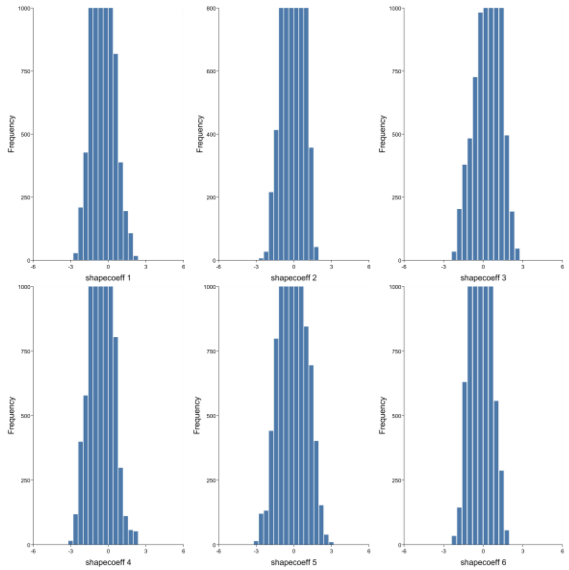
Marginal distributions (shape coefficients α)



1mm



4mm



10mm

Other likelihood functions

Reminder: Landmark likelihood

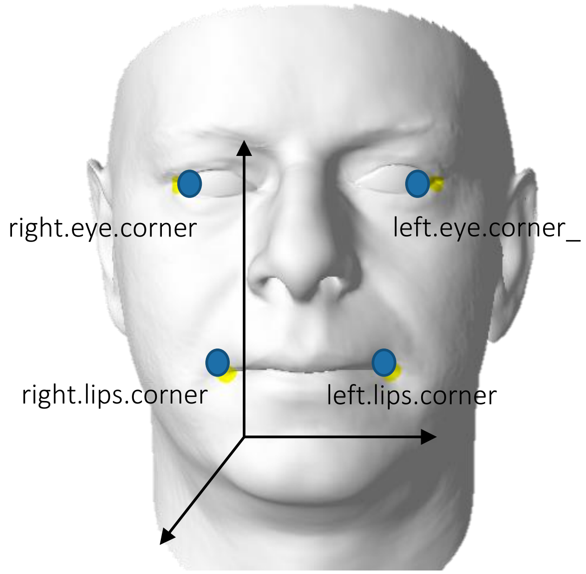
For one landmark pair (l_i^R, l_i^T):

$$p\left(l_i^T \middle| \theta, l_i^R\right) = N\left(\varphi[\theta]\left(l_i^R\right), I_{3x3}\sigma^2\right)$$

For all landmarks (assuming independence):

$$\left(l_1^T, \dots, l_n^T \middle| \theta, l_1^R, \dots, l_n^R\right) = \prod_i N\left(\varphi[\theta]\left(l_i^R\right), I_{2x2}\sigma^2\right)$$

Landmarks match target position up to zero-mean Gaussian noise.



Likelihood for points without correspondence

Match any point on target surface

$$\Gamma^T = \{p_1^T, \dots p_n^T\}$$

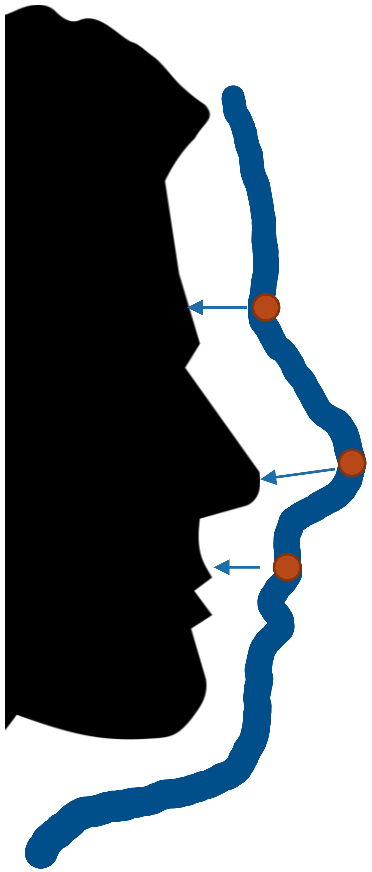
For point p_i^R on model

$$p\left(\Gamma^T \middle| \theta\right) = N\left(\text{closestPoint}\left(\Gamma^T, \varphi[\theta](p_i^R)\right), I_{3\times3}\sigma^2\right)$$

- *Corresponding points becomes closest point*

For set of points $p_1^R, \dots, p_n^R$

$$p\left(\Gamma^T \middle| \theta, p_1^R, \dots, p_n^R\right) = \prod_{i=1}^{n} N\left(\text{closestPoint}\left(\Gamma^T, \varphi[\theta](p_i^R)\right), I_{3\times3}\sigma^2\right)$$



Useful for registration/fitting of surface

Likelihood for points without correspondence

For landmark/point p_i^T on target without correspondence with model point

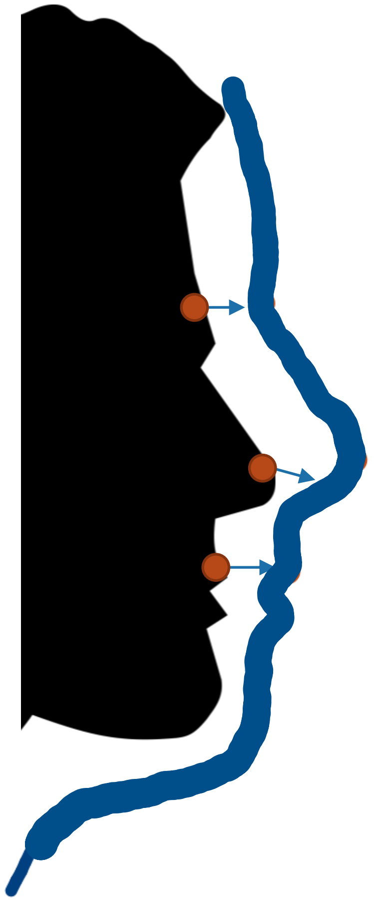
$$p(p_i|\theta) = N\big(\text{closestPoint}\big(\Gamma[\theta], p_i^T\big)\big), I_{3\times3}\sigma^2\big)$$

- $\Gamma[\theta]$ is model instance:

$$\Gamma[\theta] = \{\varphi[\theta](p_i^R) | p_i^R \in \Gamma^R\}$$

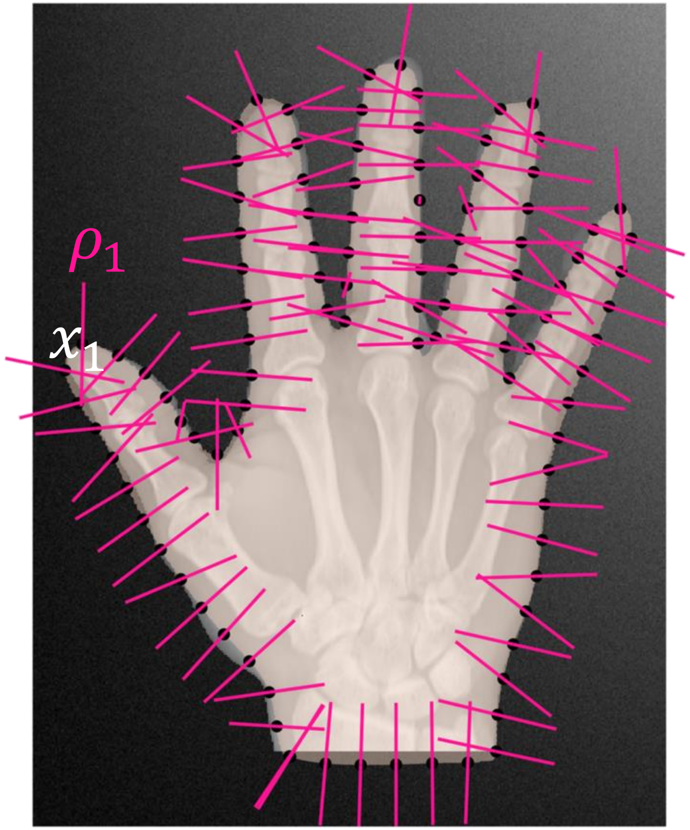
For set of points $p_1, \ldots, p_n$

$$p\big(p_1^T, \ldots, p_n^T \big| \theta\big) = \prod_{i=1}^{n} N\big(\text{closestPoint}\big(\Gamma[\theta], p_i^T\big)\big), I_{3\times3}\sigma^2\big)$$



Likelihood function: Active shape models

Shape is well matched if environment around profile points is likely under trained model.



- ASMs model each profile $\rho(x_i)$ as a normal distribution

$$p(\rho(x_i)) = N(\mu_i, \Sigma_i)$$

Extracts profile (feature) from image

- Single profile point x_i:

$$p(\rho(\varphi[\theta](x_i))|\theta, x_i) = N(\mu_i, \Sigma_i)$$

- Likelihood for all profile points:

$$p(\rho(\varphi[\theta](x))|\theta, \Gamma_R) = \prod_i N(\mu_i, \Sigma_i)$$

Misc. Topics

Sequential Bayesian updating

Update belief when data $M_1, \dots M_n$ becomes available

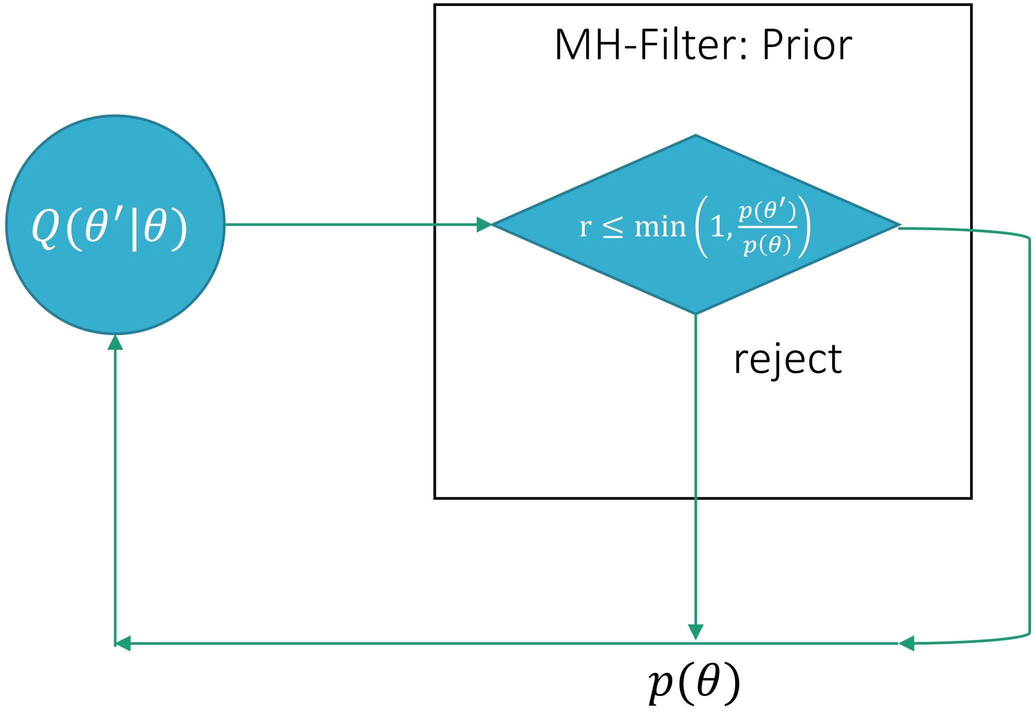
$$p(\theta) \rightarrow p(\theta|M_1) \rightarrow p(\theta|M_1, M_2) \rightarrow \cdots$$

Possible implementation in Metropolis-Hastings:

- Posterior of previous step becomes proposal distribution
$$Q(\theta'|\theta) = P(\theta')$$
- Known as Independent Metropolis-Hastings, as proposal does not depend on previous state

Metropolis-Hastings as filtering

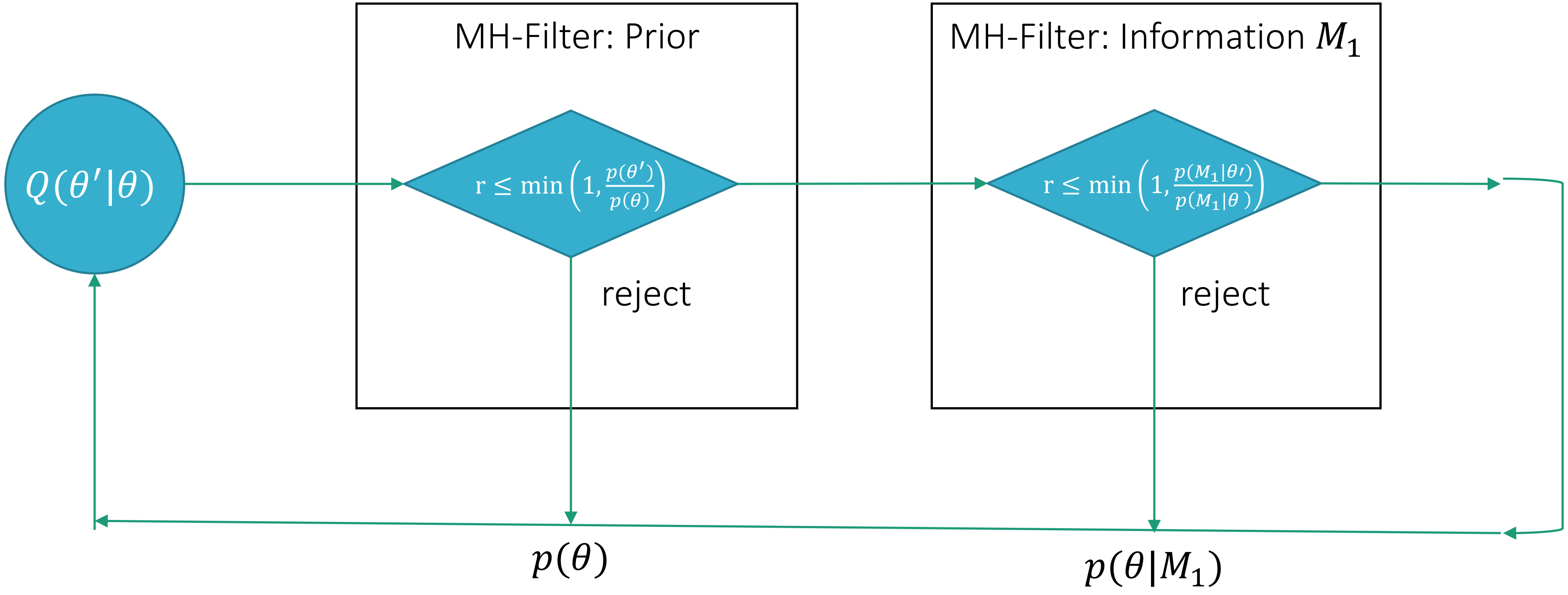
Sampling from $p(\theta)$ using Metropolis-Hastings can be seen as filtering



Sequential Bayesian updating using Metropolis-Hastings

Sequential belief update:

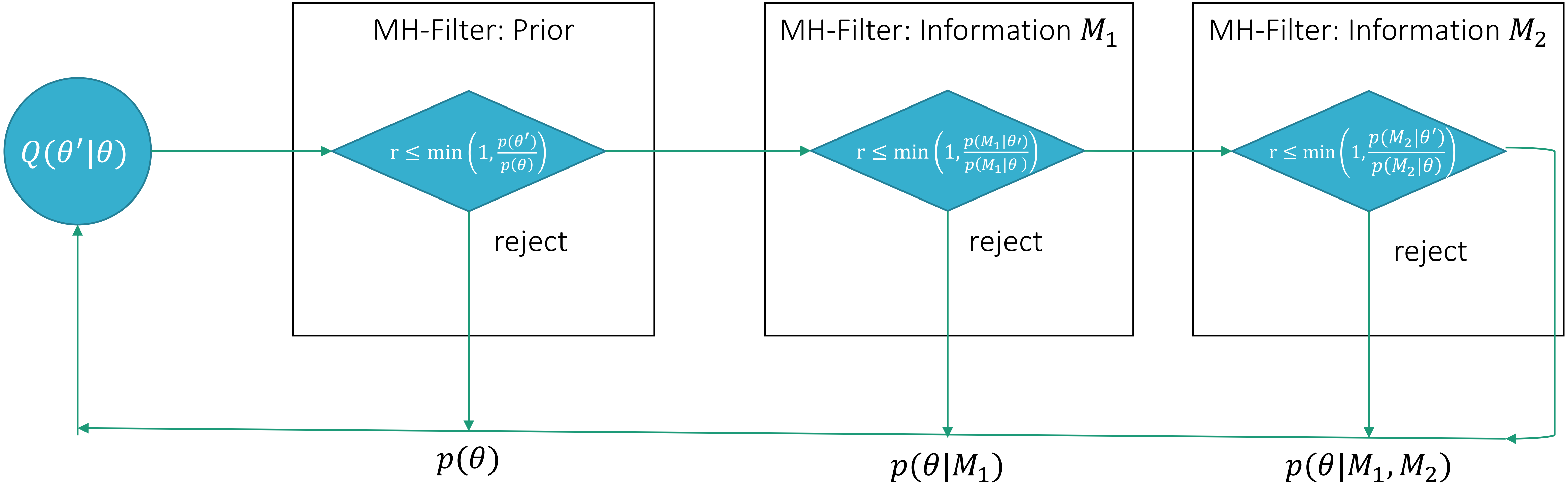
$$p(\theta) \rightarrow p(\theta|M_1)$$



Sequential Bayesian updating using Metropolis-Hastings

Sequential belief update:

$$p(\theta) \rightarrow p(\theta|M_1) \rightarrow p(\theta|M_1, M_2)$$



Implementation in Scalismo

Scalismo provides special *Filtering Proposal*

- Can be used like any other proposal

```
val priorEvaluator : DistributionEvaluator[Sample] = ???
val proposalGen : ProposalGenerator[Sample] = ???

val metropolisFilterProposalGen1 = MetropolisFilterProposal(proposalGen, priorEvaluator)

val likelihoodEvaluator1 : DistributionEvaluator[Sample] = ???
val metropolisFilterProposalGen2 = MetropolisFilterProposal(metropolisFilterProposalGen1, likelihoodEvaluator1)

val likelihoodEvaluator1 : DistributionEvaluator[Sample] = ???

val mh = MetropolisHastings(metropolisFilterProposalGen2 , likelihoodEvaluator2)
```

MH as propose-and-verify

- Metropolis algorithm formalizes *propose-and-verify idea*
    - *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.

MH as propose-and-verify

Advantage

- Can include proposals that fail sometimes
  - Example: Landmark detector with only 90% accuracy
  - Algorithm is robust to 10% failures of proposals

Disadvantage

- Can include proposals that fail always
  - Example: Buggy proposal
  - Algorithm is robust to failure. Buggy proposal is not detected

Solution: Logging

We need to log the acceptance rate of every proposal!

- Low acceptance rate indicates something is wrong -> Debug
- Optimal acceptance rate of random walk proposal:
  - Between 20 and 30 %
- More sophisticated (and more expensive) proposals should have higher acceptance rate

Summary: MCMC for 3D Fitting

- Modelling in the analysis-by-synthesis framework leads to intractable posterior computation
  - Need for using approximate inference

- Metropolis-Hastings algorithm provides powerful framework
  - Propose and verify
    - Propose update step
    - Verify and accept with probability
  - Can integrate uncertain information
  - Allows for sequential update of information

  - Samples converge to true distribution: More about this later!