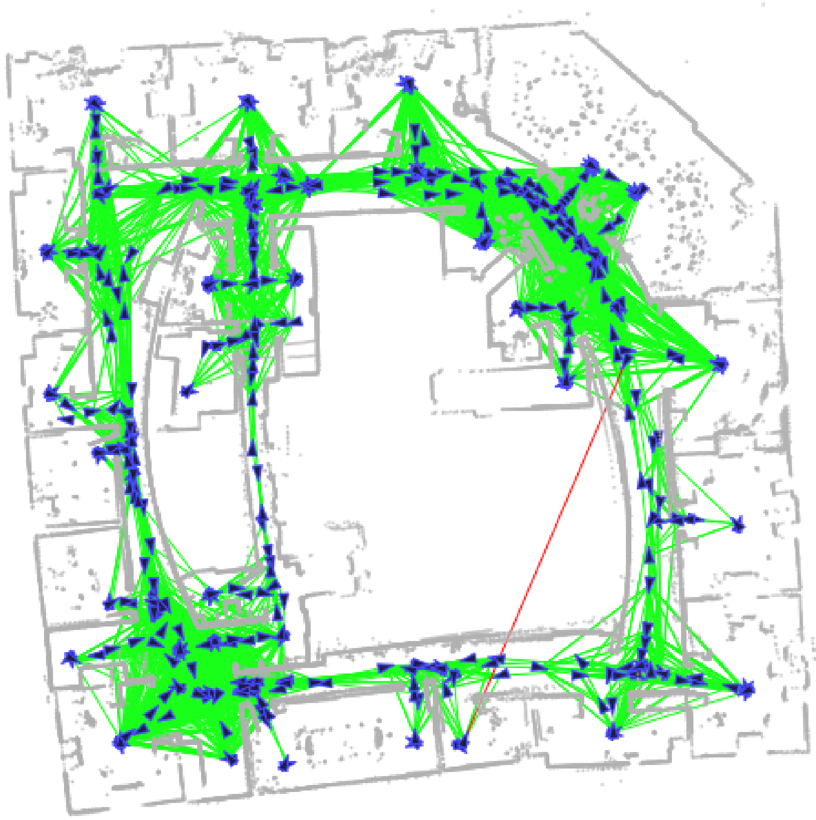
Learning Max Mixtures

Edwin Olson
ebolson@umich.edu
Computer Science and Engineering
University of Michigan

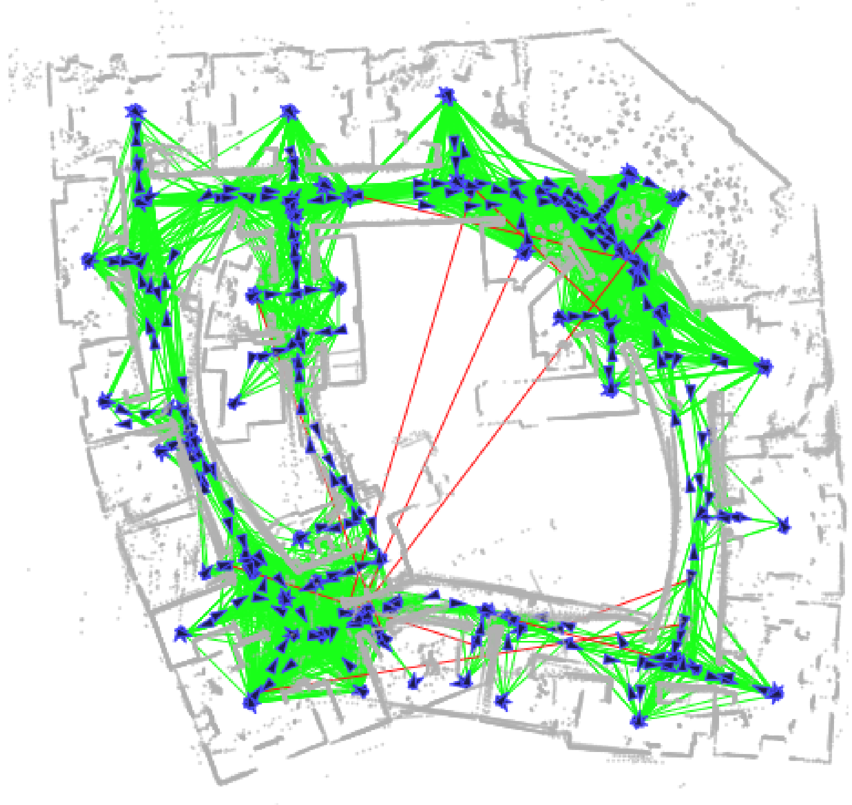
The ugly

Errors

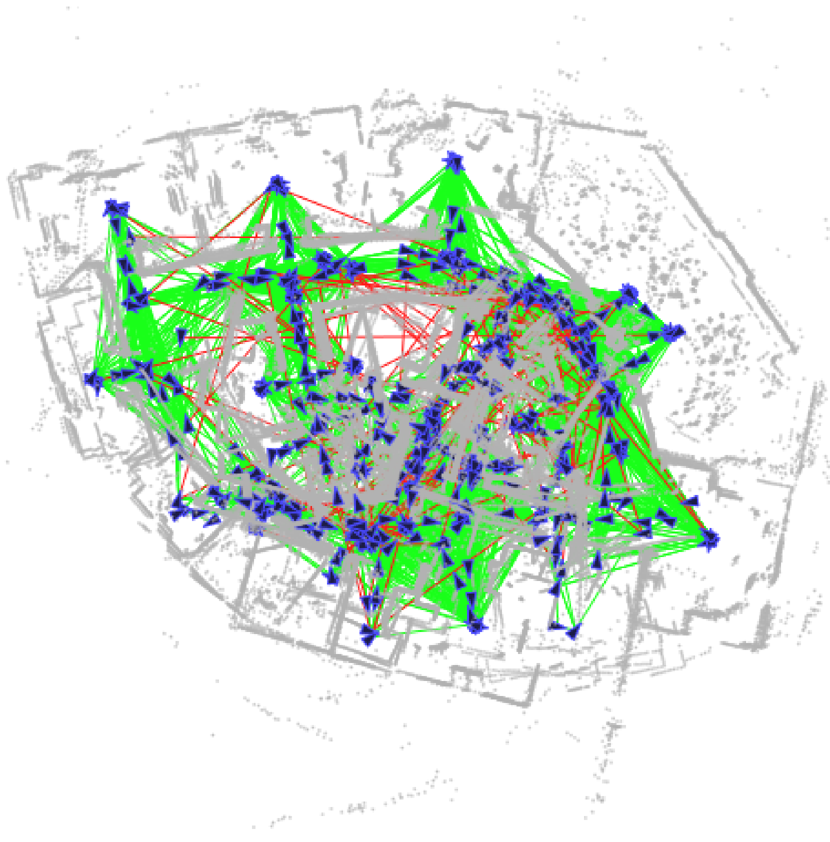
- Classical SLAM systems are very sensitive to errors
  - Natural question: How do we reduce the error rate?



1 error



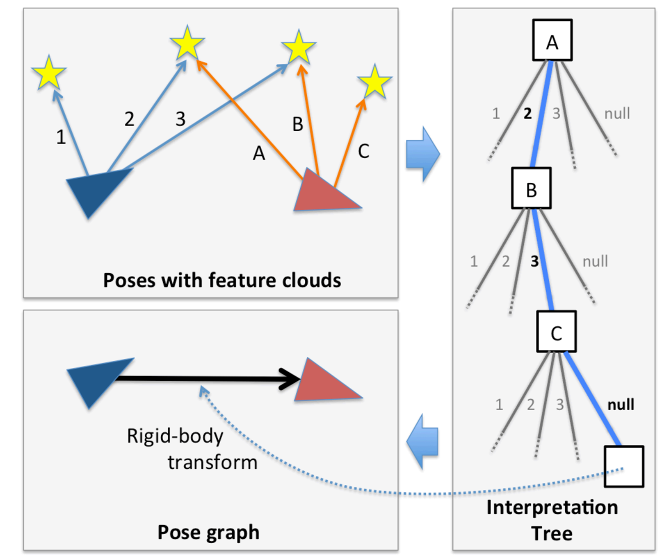
10 errors

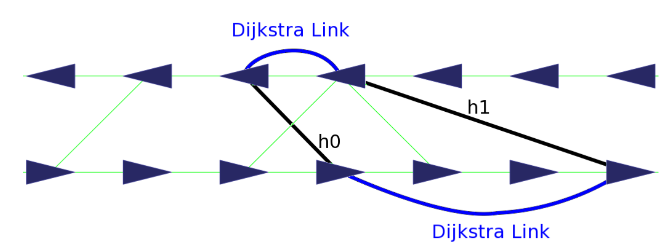


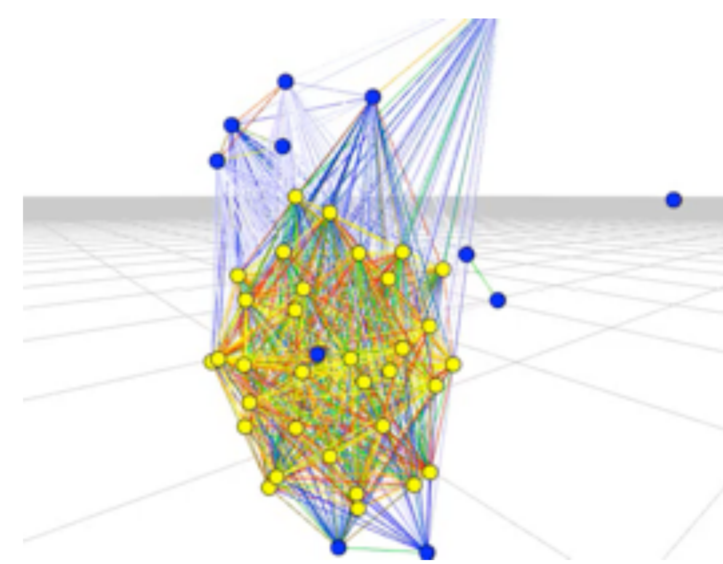
100 errors

Reducing Error Rates

- Neira: JCBB (2001)
- Bailey: CDDA (2002)
- Bosse: Loop Validation (2004)
- Us: SCGP (2008)
- Us: Correlative Scan Matching (2009)
- Us: IPJC (2012)







The problem

- Each of these methods pushes error rates closer to zero. Great!

- But mapping methods can diverge with even a single error. Not Great!

- *Outlier-rejection/loop-validation methods can postpone failure, but can't eliminate it!*

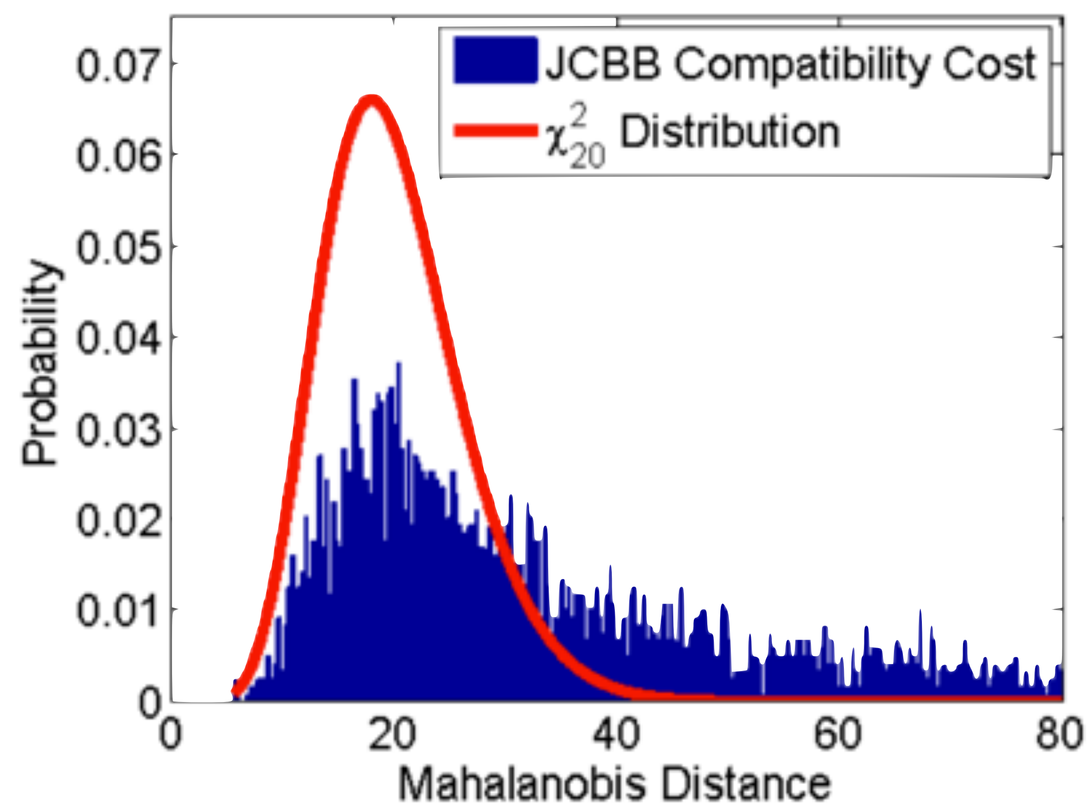
Let’s take a fresh perspective

Let’s take a fresh perspective

- What is an error, anyway?

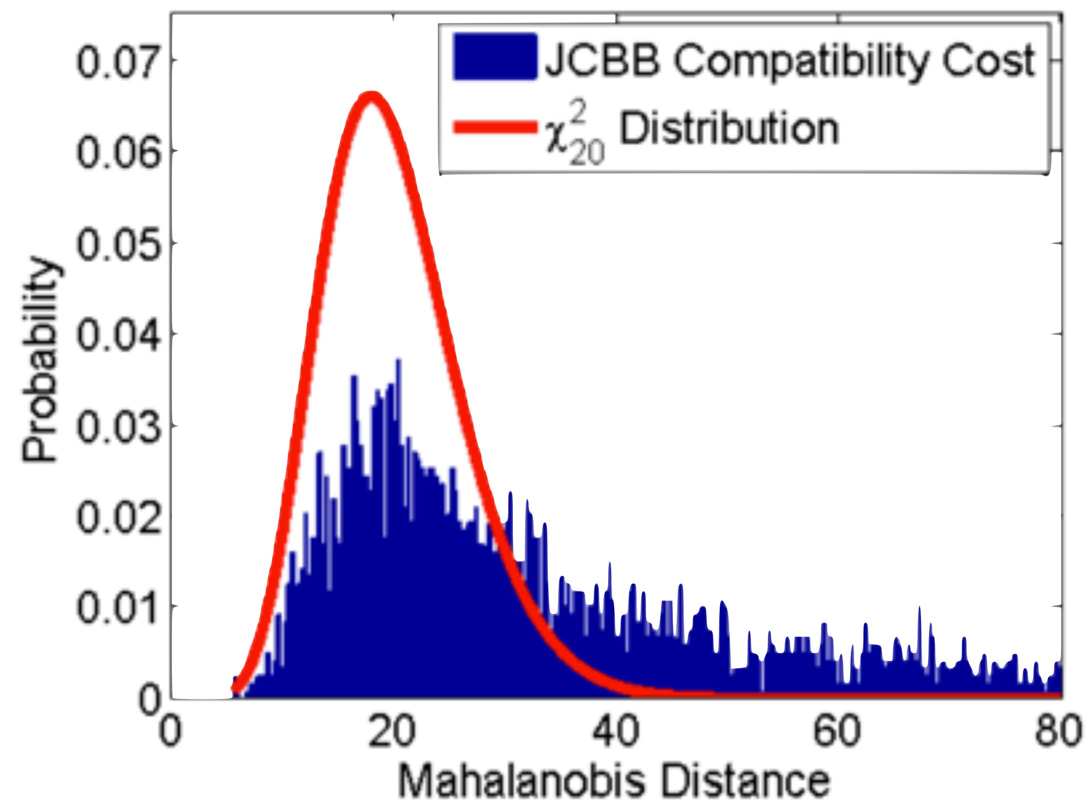
Let's take a fresh perspective

- What is an error, anyway?
    - It's an inconsistency between our probabilistic observation model and the empirical accuracy of our data association method.

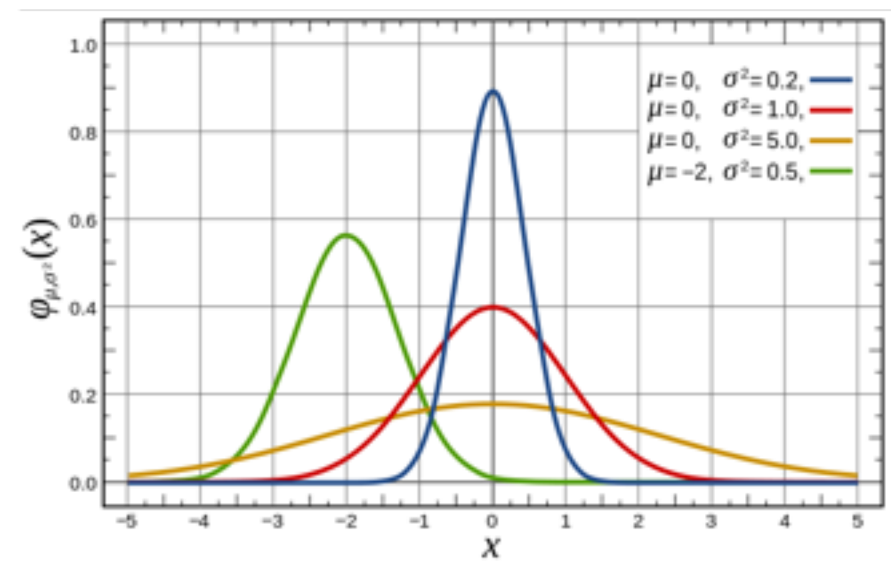


Let's take a fresh perspective

- What is an error, anyway?
  - It's an inconsistency between our probabilistic observation model and the empirical accuracy of our data association method.
  - Maybe the problem isn't outlier rejection, maybe the problem is that we're using the wrong probabilistic models!



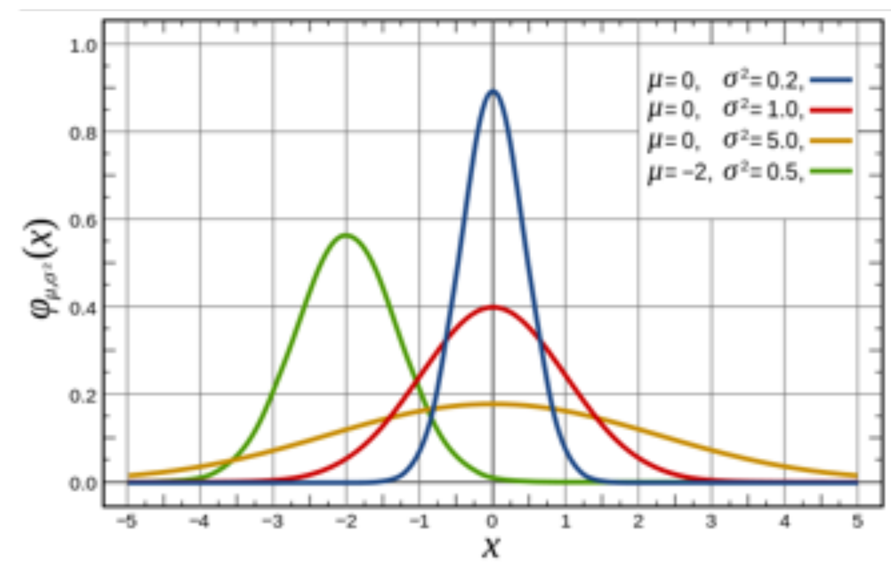
Gaussian error models



$$p(z_i|x) = N(\mu_i, \Lambda_i^{-1})$$

Gaussian error models

- Almost all robotics work uses Gaussian error models
  - Lead to very simple least-squares state estimation algorithms.
  - Believed to be sufficiently representative



$$p(z_i|x) = N(\mu_i, \Lambda_i^{-1})$$

Gaussian Errors = Easy inference

$$p(z_i|x) = N(\mu_i, \Lambda_i^{-1}) = \frac{1}{\gamma} e^{-\frac{1}{2}(x-\mu)^T \Lambda_i (x-\mu)}$$

$$x_{opt} = \operatorname{argmax}_x \prod p(z_i|x)$$

$$x_{opt} = \operatorname{argmax}_x \log\left(\prod_i p(z_i|x)\right)$$

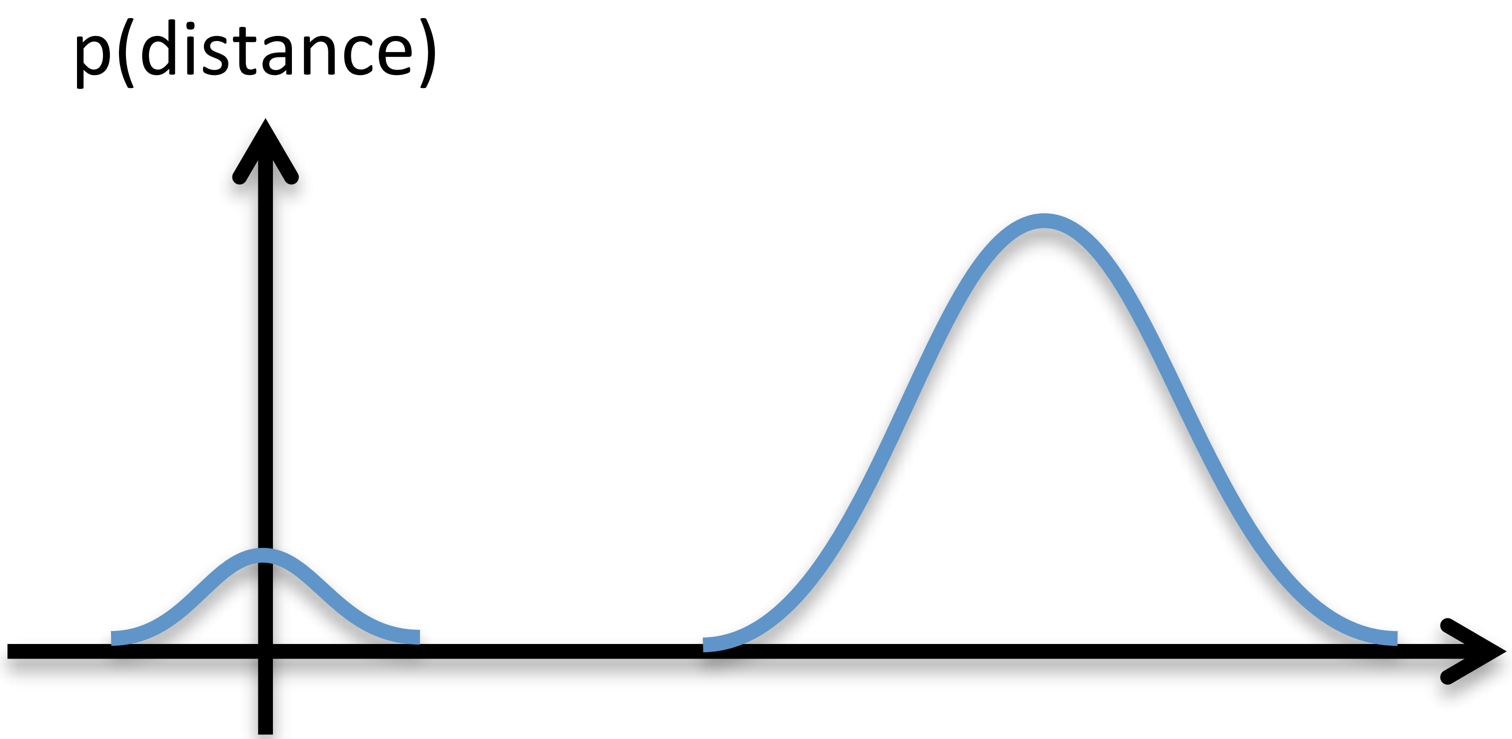
$$x_{opt} = \operatorname{argmax}_x \sum_i \log p(z_i|x)$$

$$x_{opt} = \operatorname{argmax}_x \sum_i (a_i x^2 + b_i x + c_i)$$

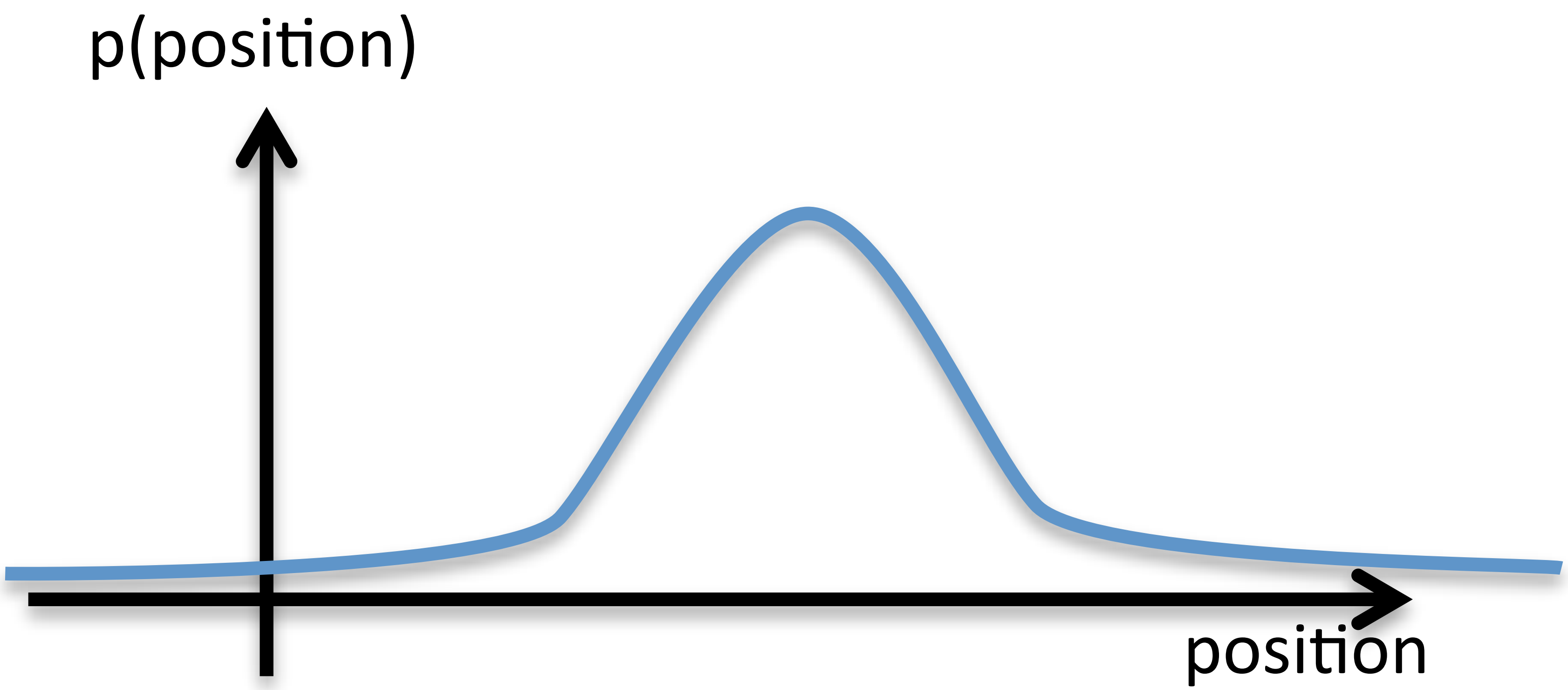
$$A x_{opt} = b \longleftarrow$$

unreasonably fast methods now available!

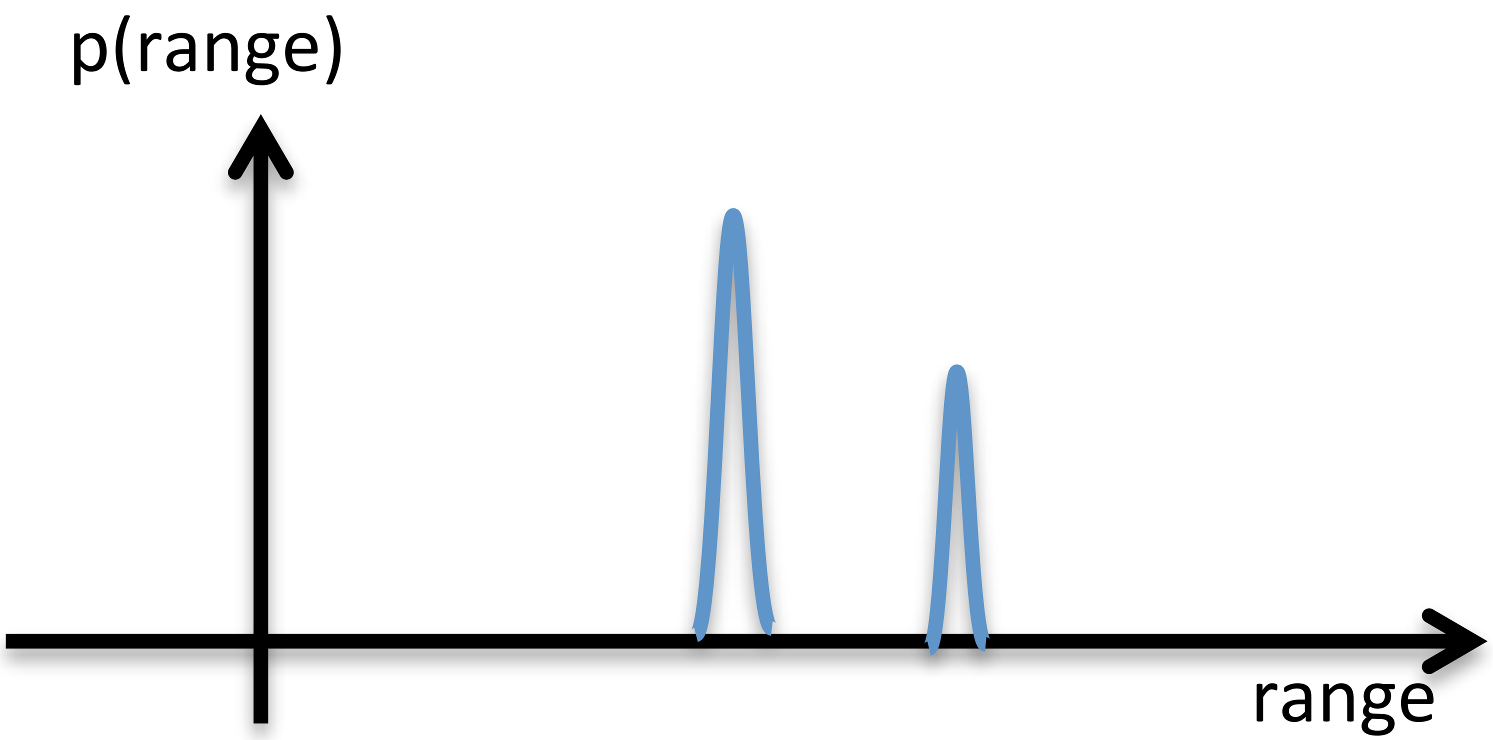
Real-world errors (maybe)



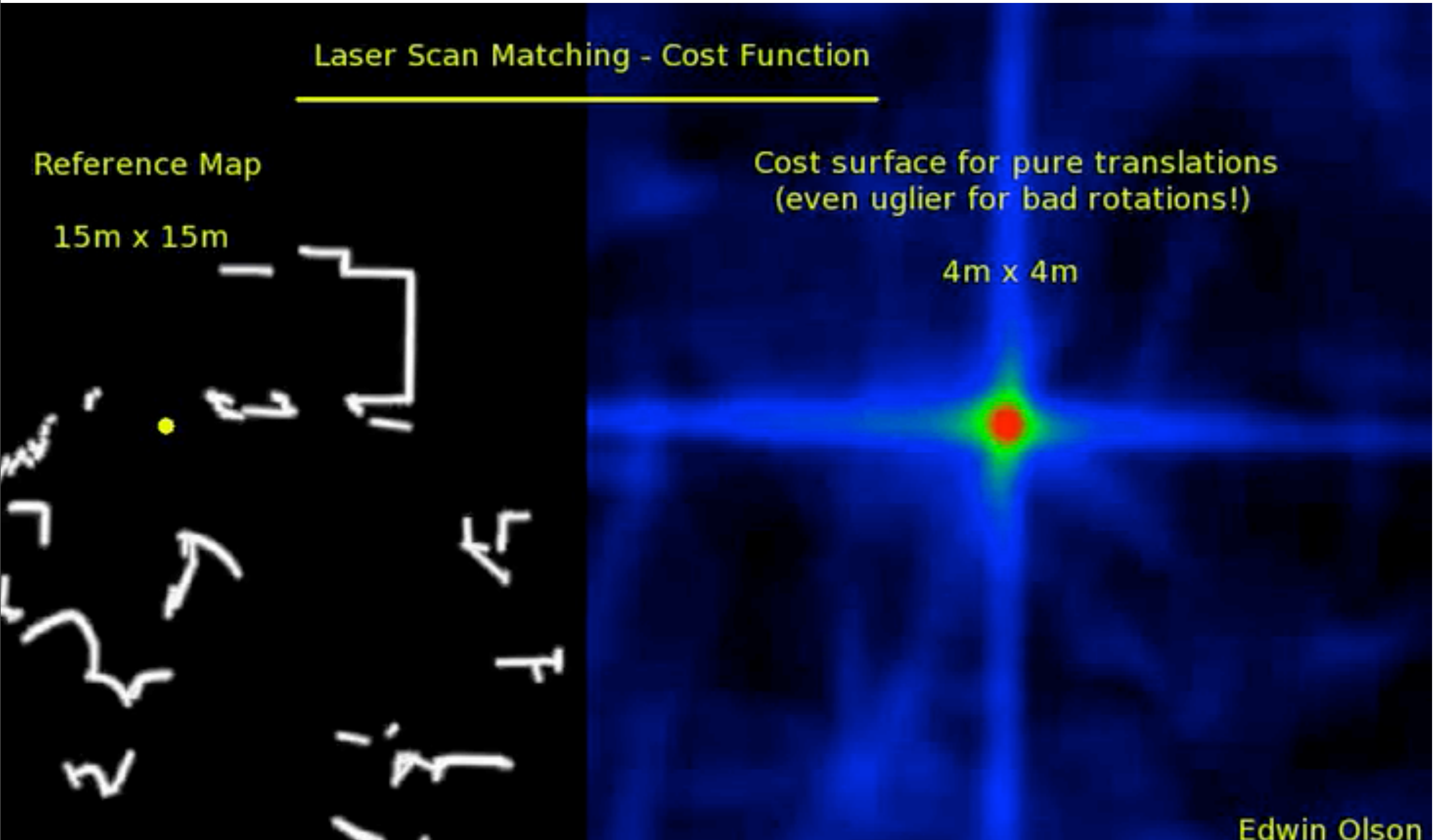
Odometry *Slip or Grip*



Loop closure: null hypothesis



Sonar multi-path / surface reflections / loop closing with aliasing


Laser Scan Matching - Cost Function
Reference Map
15m x 15m
Cost surface for pure translations
(even uglier for bad rotations!)
4m x 4m
Edwin Olson

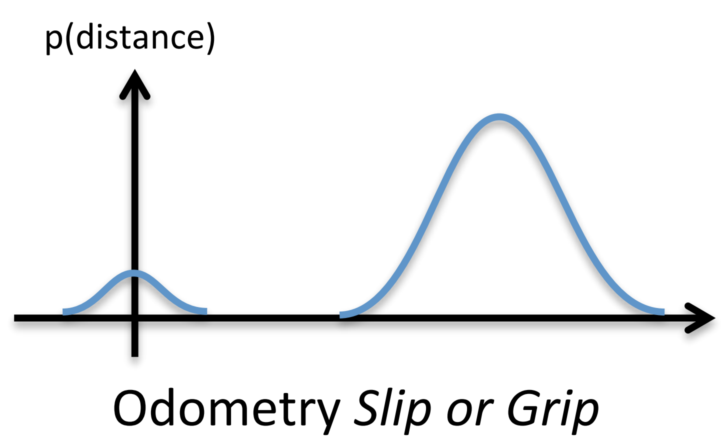
Two basic problems

- How can we represent these more complex error models?
- How do we solve the resulting inference problems?

Sums of Gaussians

- One "obvious" way to represent more types of error functions

$$p(z_i|x) = N(\mu_i, \Lambda_i^{-1}) \quad \Longrightarrow \quad p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1})$$



$$p(z_{31}|x) = 0.1N(0, 0.25) + 0.9N(1, 0.5)$$

Sums of Gaussians

$$p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1})$$

$$x_{opt} = \operatorname{argmax}_x \prod p(z_i|x)$$

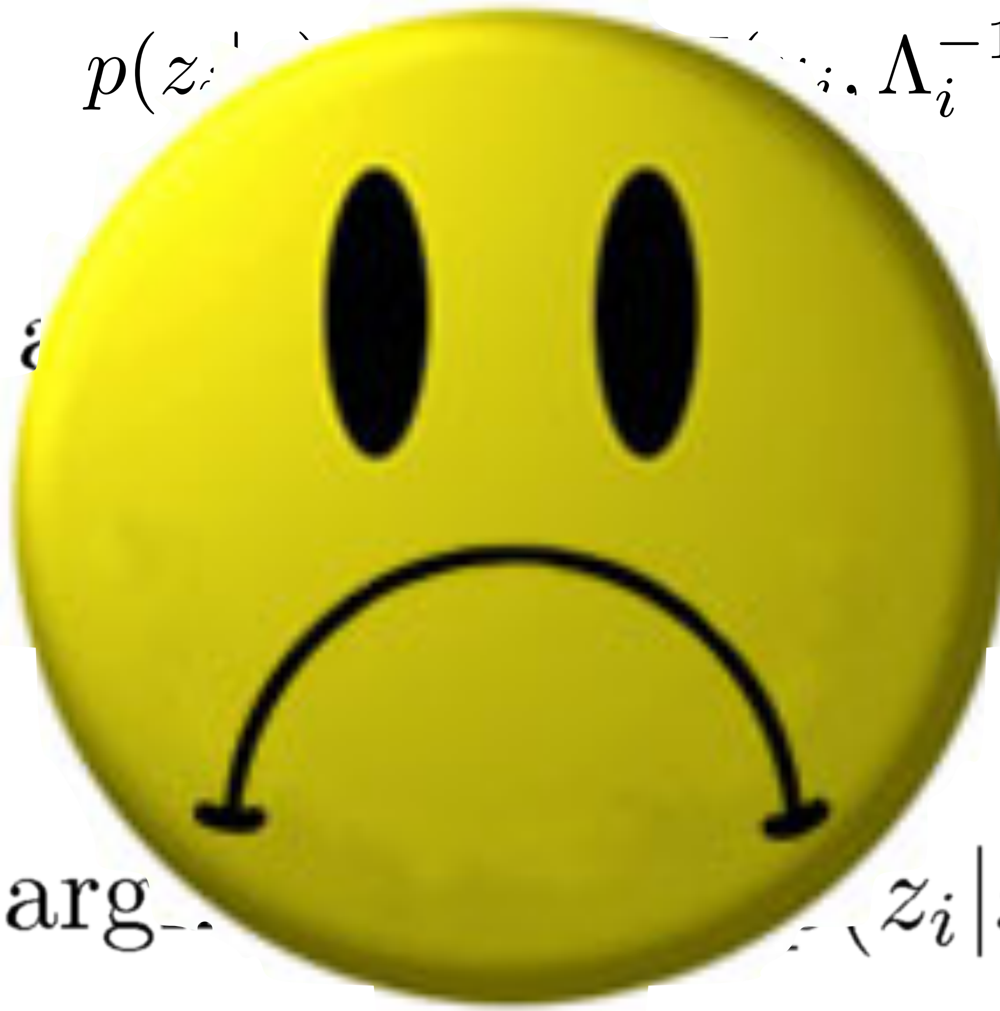
$$x_{opt} = \operatorname{argmax}_x \log \left(\prod_i p(z_i|x) \right)$$

$$x_{opt} = \operatorname{argmax}_x \sum_i \log p(z_i|x)$$

$$x_{opt} = \operatorname{argmax}_x \sum_i \log \left(\sum_j w_j N(\mu_j, \Lambda_j^{-1}) \right)$$

Sums of Gaussians

$p(z_i \ldots \Lambda_i^{-1})$



$x_{opt} = a\ldots$

$x_{opt} = \ldots |x) \Big)$

$x_{opt} = \arg\ldots (z_i|x)$

$$x_{opt} = \text{argmax}_x \sum_i \log \left(\sum_j w_j N(\mu_j, \Lambda_j^{-1}) \right)$$

Challenge

- Can we find a way of representing more complex error functions?

- *AND* make sure that we can actually solve the resulting problem?

Our approach

$$p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1})$$

$$p(x|z) \propto \prod_i p(z_i|x)$$

Our approach

- Use mixture models for more realistic probability distributions
  - Change SUM to MAX

$$p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1})$$

$$p(x|z) \propto \prod_i p(z_i|x)$$

Our approach

- Use mixture models for more realistic probability distributions
    - Change SUM to MAX

$$p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1}) \quad \Rightarrow \quad p(z_i|x) = \max_i w_i N(\mu_i, \Lambda_i^{-1})$$

$$p(x|z) \propto \prod_i p(z_i|x)$$

- *Can* "push" the log past the MAX...

Our approach

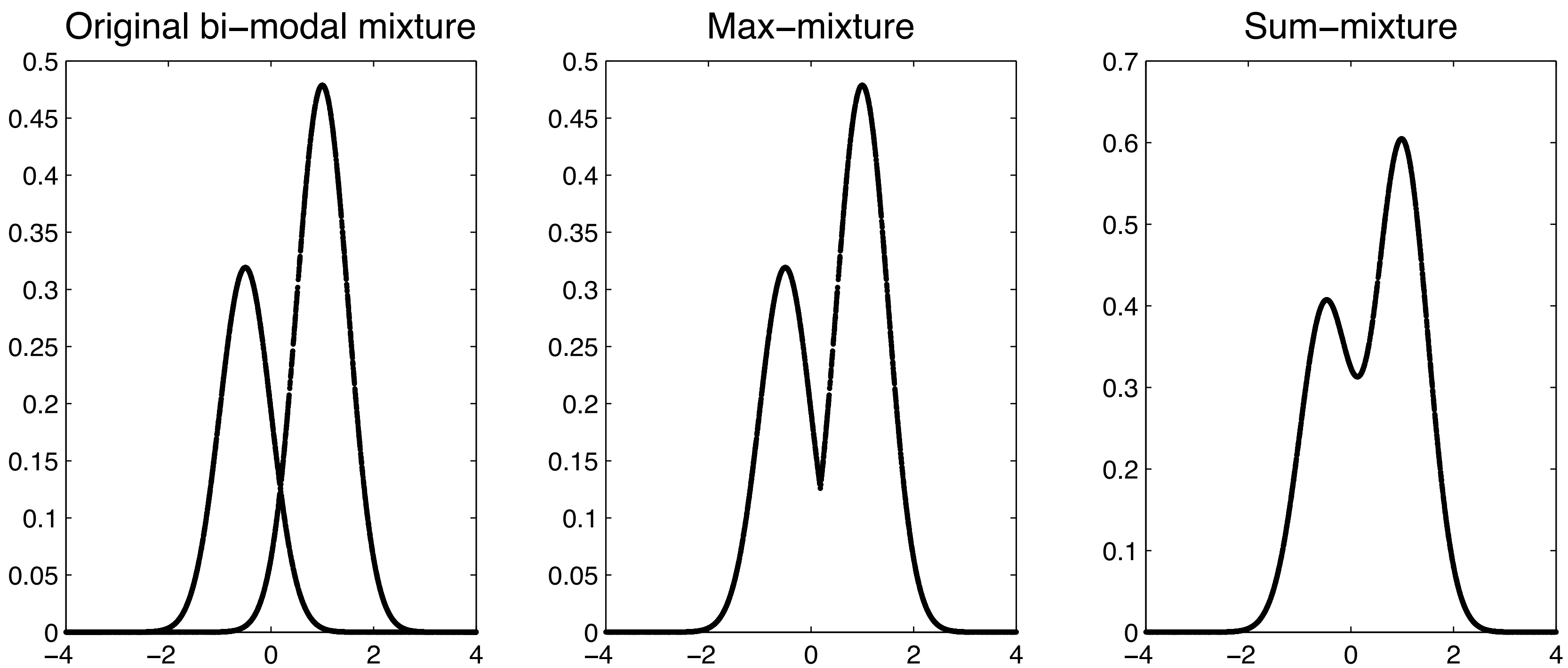
- Use mixture models for more realistic probability distributions
  - Change SUM to MAX

$$p(z_i|x) = \sum_i w_i N(\mu_i, \Lambda_i^{-1}) \Rightarrow p(z_i|x) = \max_i w_i N(\mu_i, \Lambda_i^{-1})$$

$$p(x|z) \propto \prod_i p(z_i|x)$$

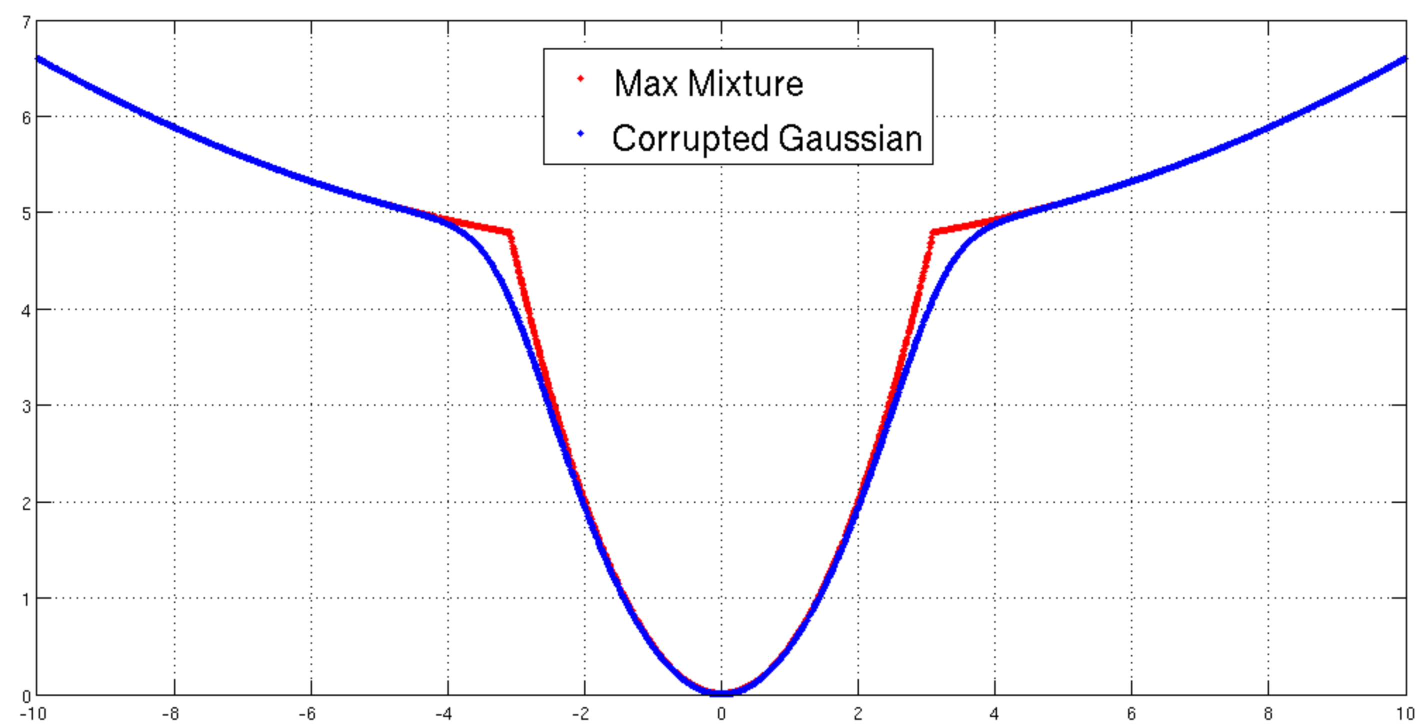
  - *Can* "push" the log past the MAX...
    - Results in Ax=b, just like with simple Gaussian error models

Max Mixtures: Examples



Comparison between sum and max mixtures

Cost function interpretation (Corrupted Gaussian)



Inference on Max Mixtures

- Max mixture formulation is highly suggestive of an optimization strategy:

- Inference on Gaussians:
  - For each observation $p(z_i|x)$
    - Compute contribution to A matrix and b vector from the Gaussian.
  - Solve $Ax = b$

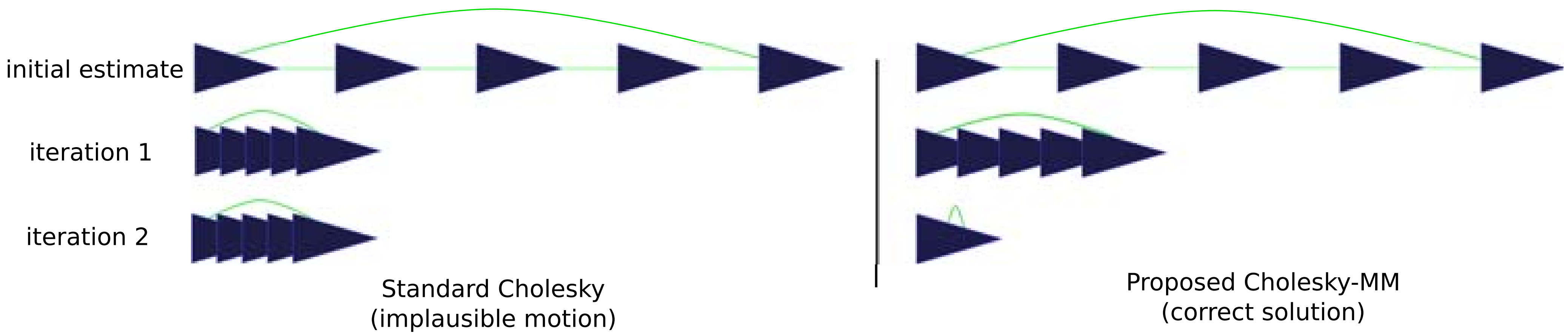
- Inference on Max Mixtures
  - For each observation $p(z_i|x)$
    - Find mixture component j that is most likely given x.
    - Compute contribution to A matrix and b vector from Gaussian mixture component j.
  - Solve $Ax = b$

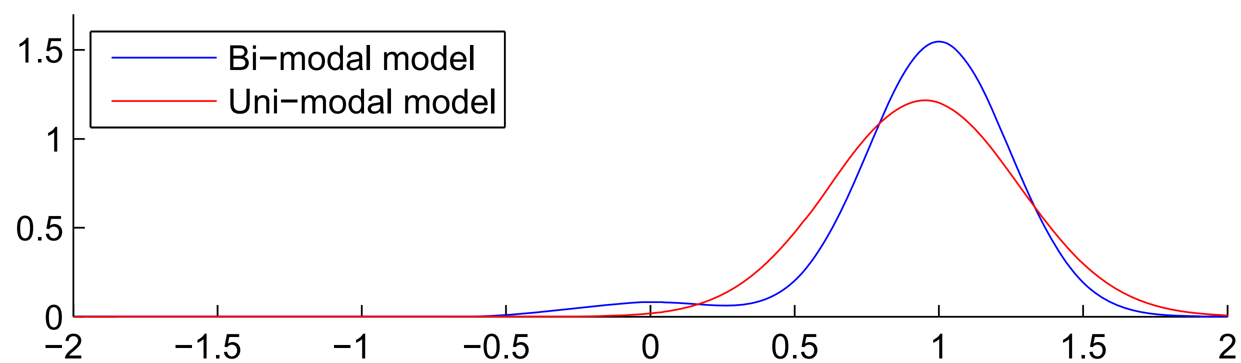
The $20 question

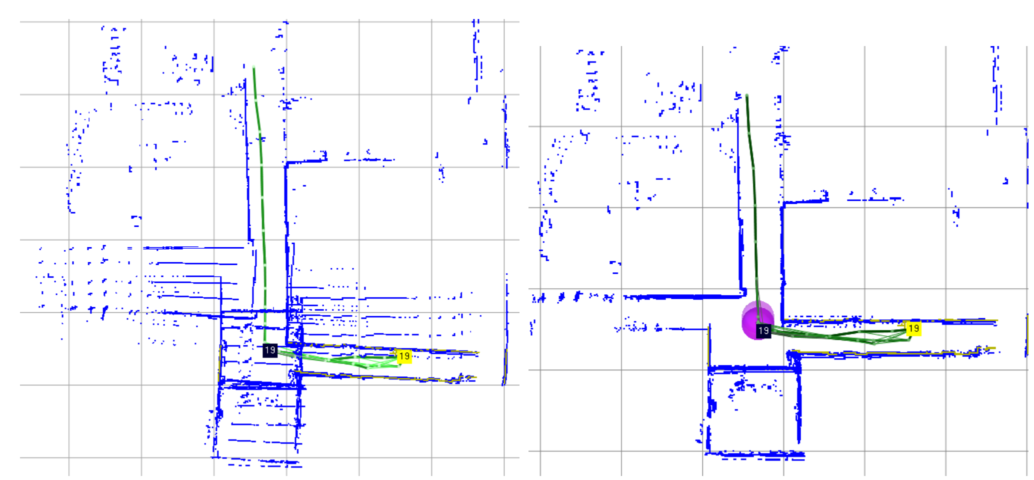
- Least-Squares regression (inference on Gaussians) is convex*:
  - A single minimum
  - All starting points lead to the minimum
  - Well behaved optimization problem!

- This is *not* true of max mixtures:
  - Exponentially many local minima!
  - Does this scheme robustly find the global minimum?

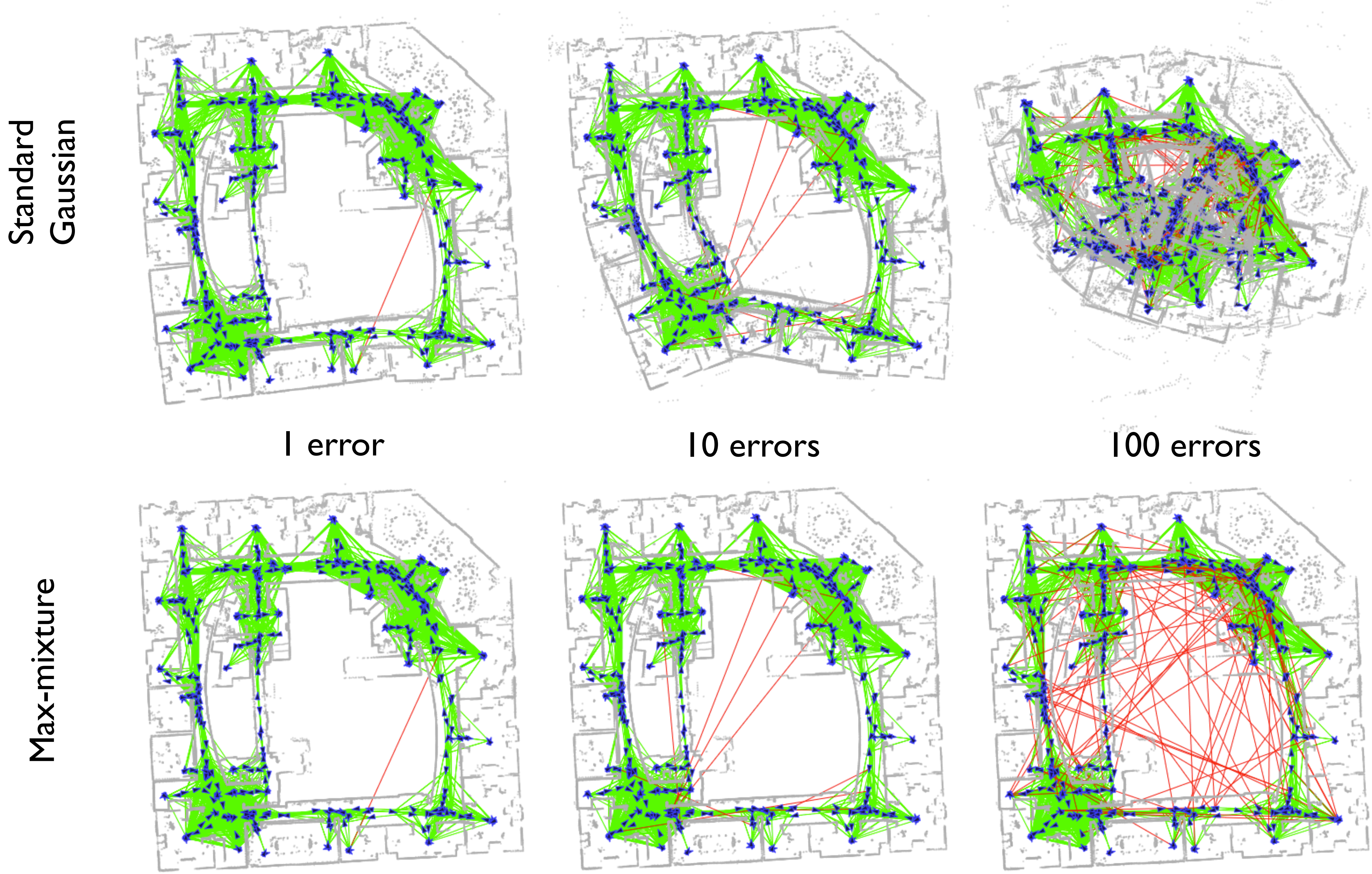
Slip or Grip (Toy Problem)



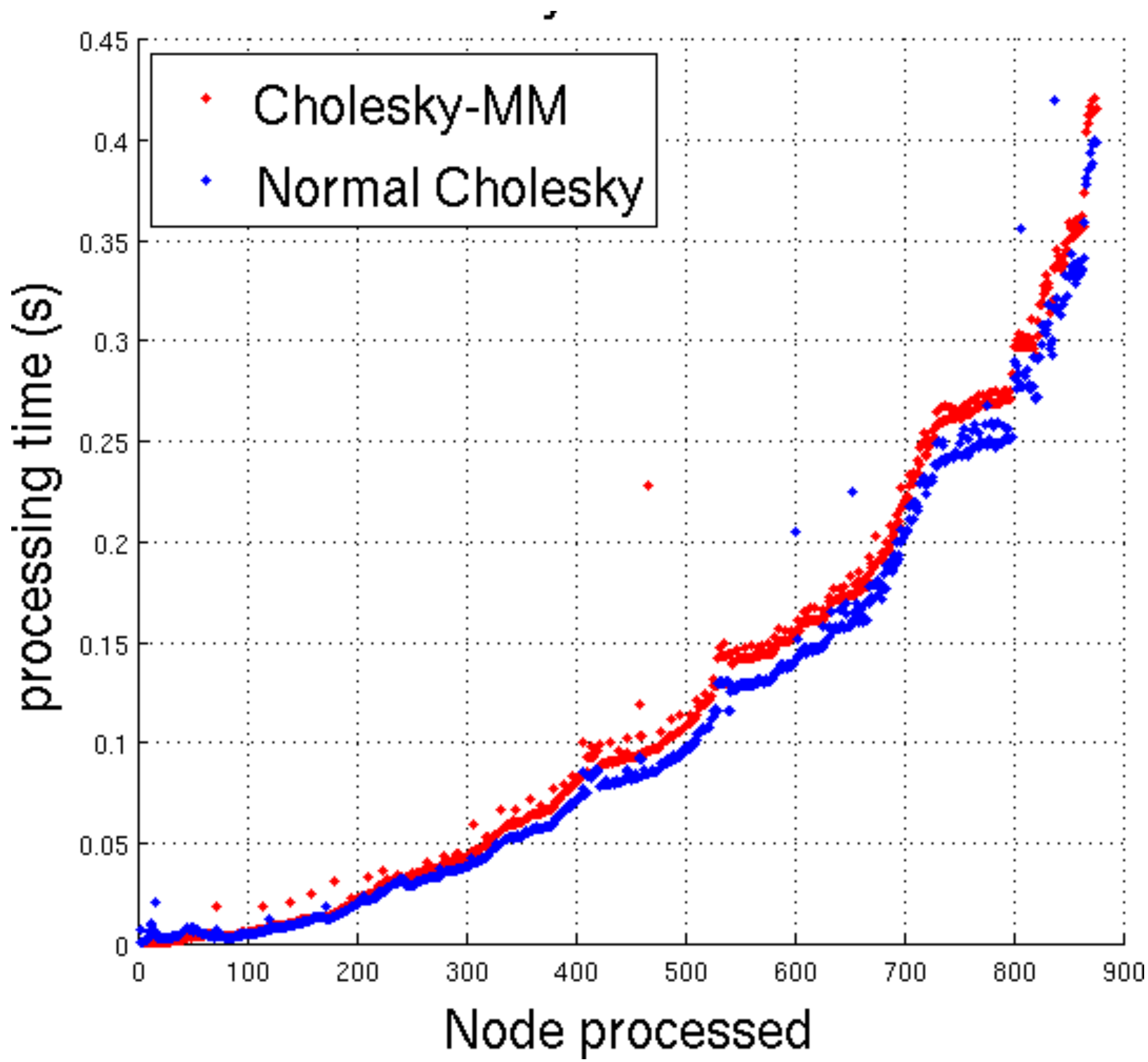




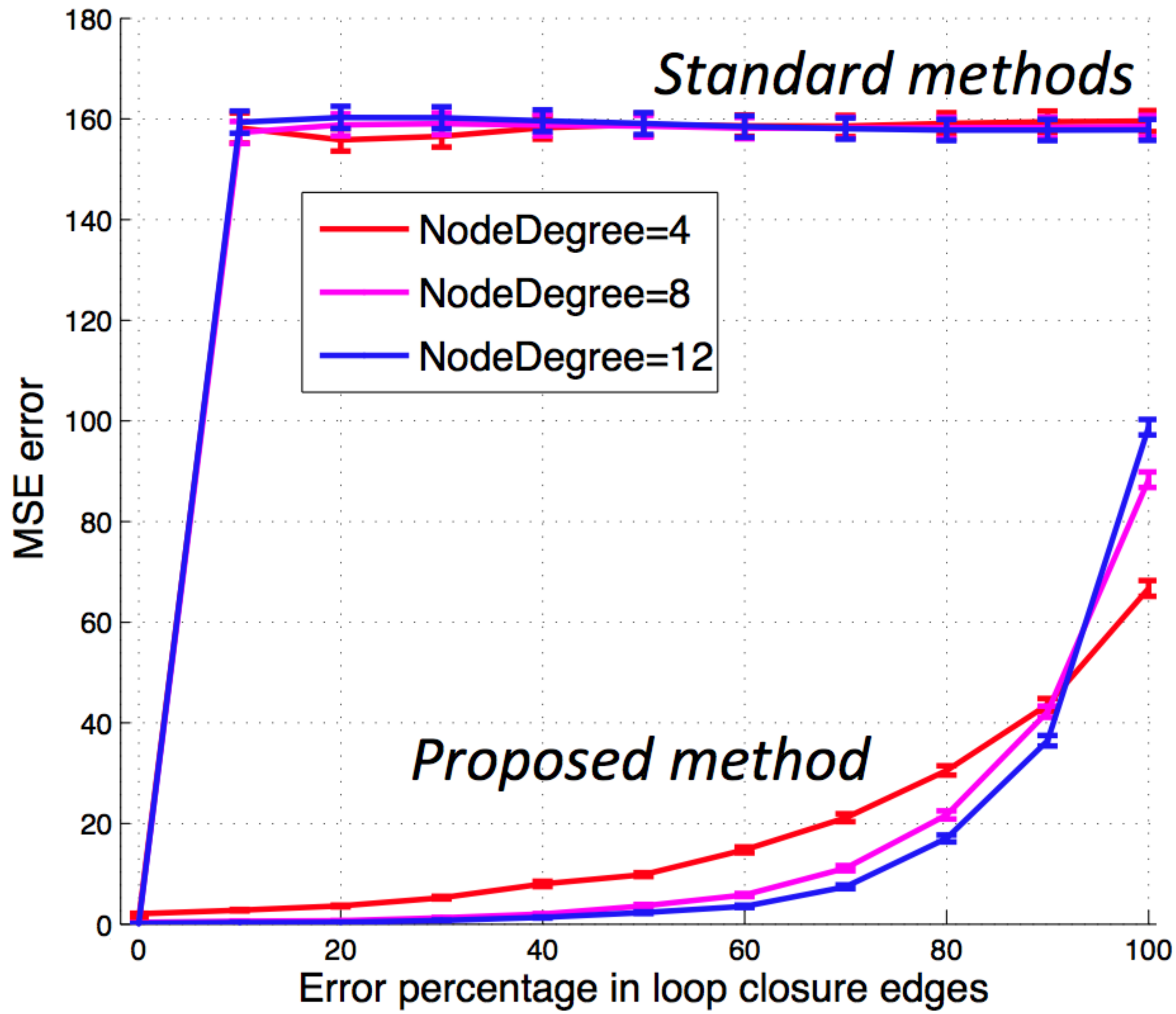
Results



Results: CPU Time

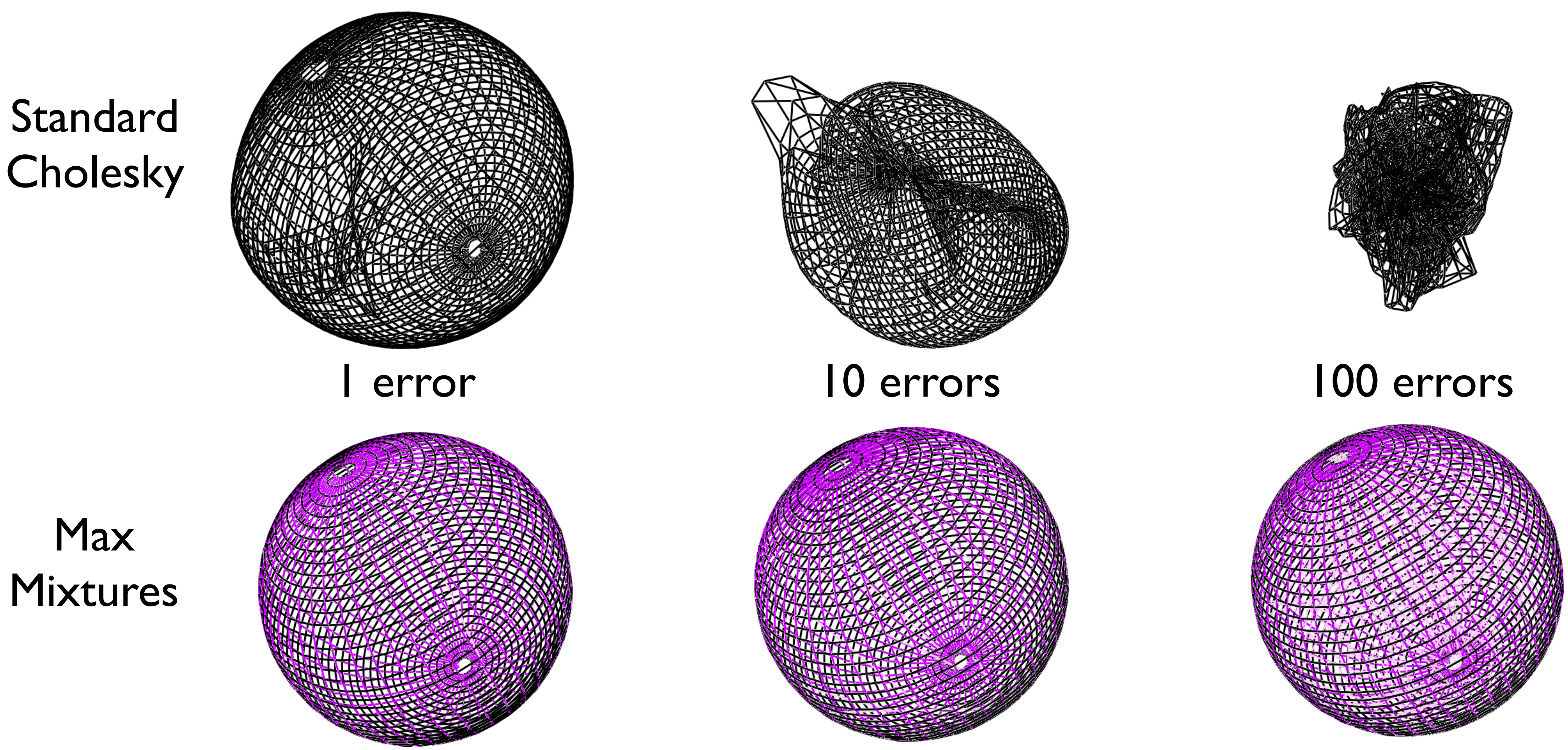


Results: Robustness

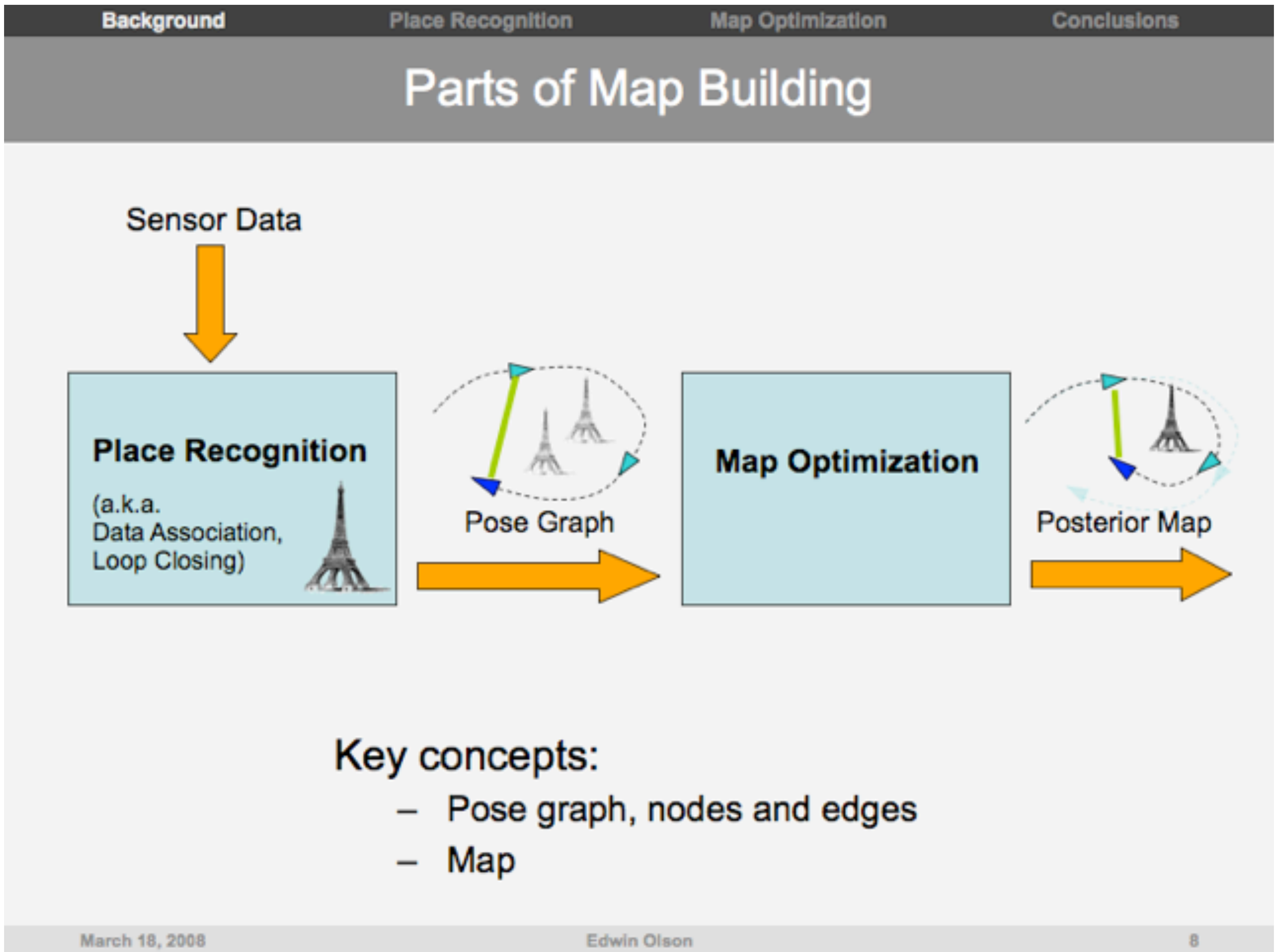


Does it work in 6DOF?

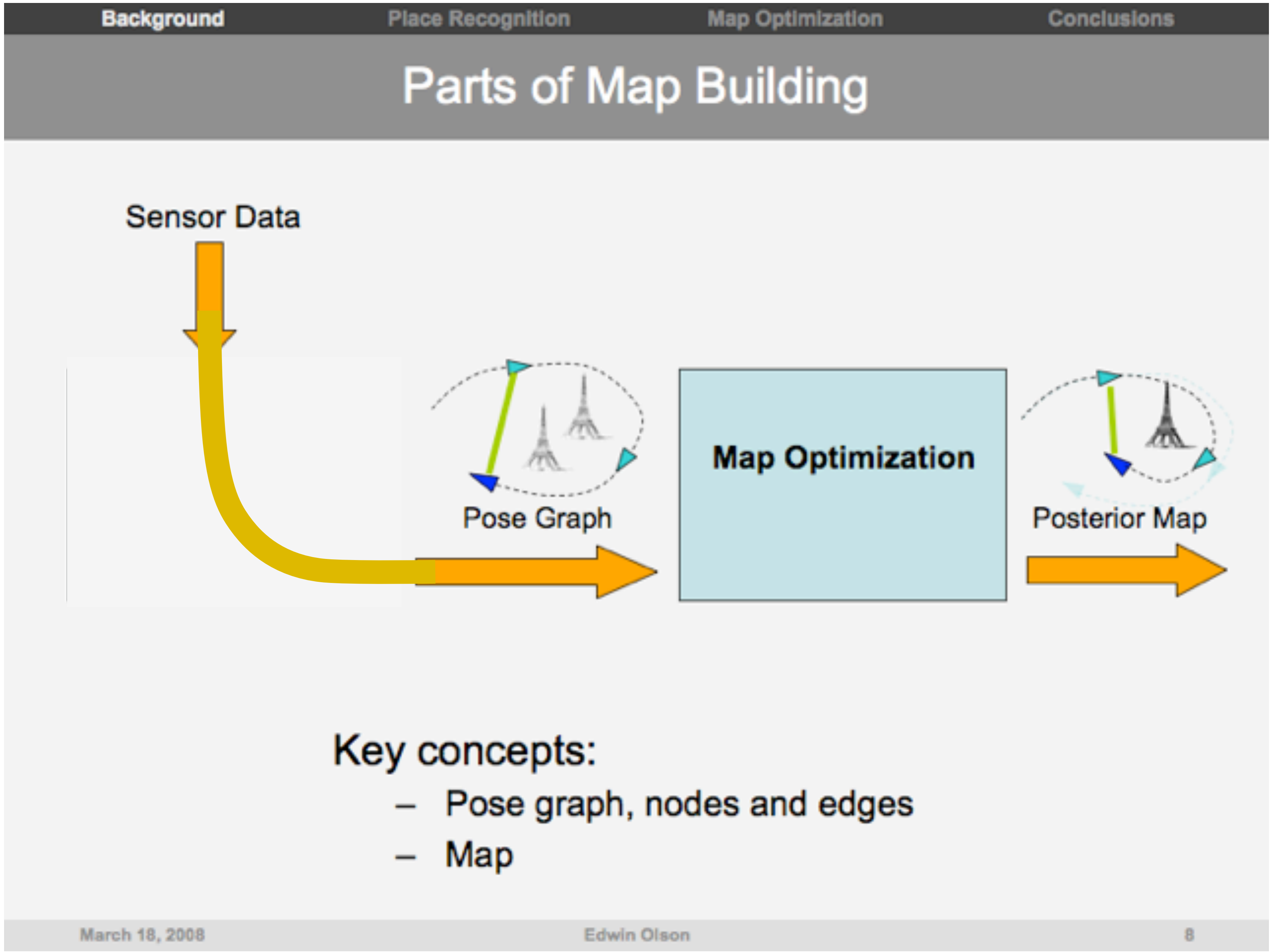
- 3D generally much harder than 2D:
  - Rotations exacerbate local minima problems due to non-linear effects.



“Extreme SLAM”

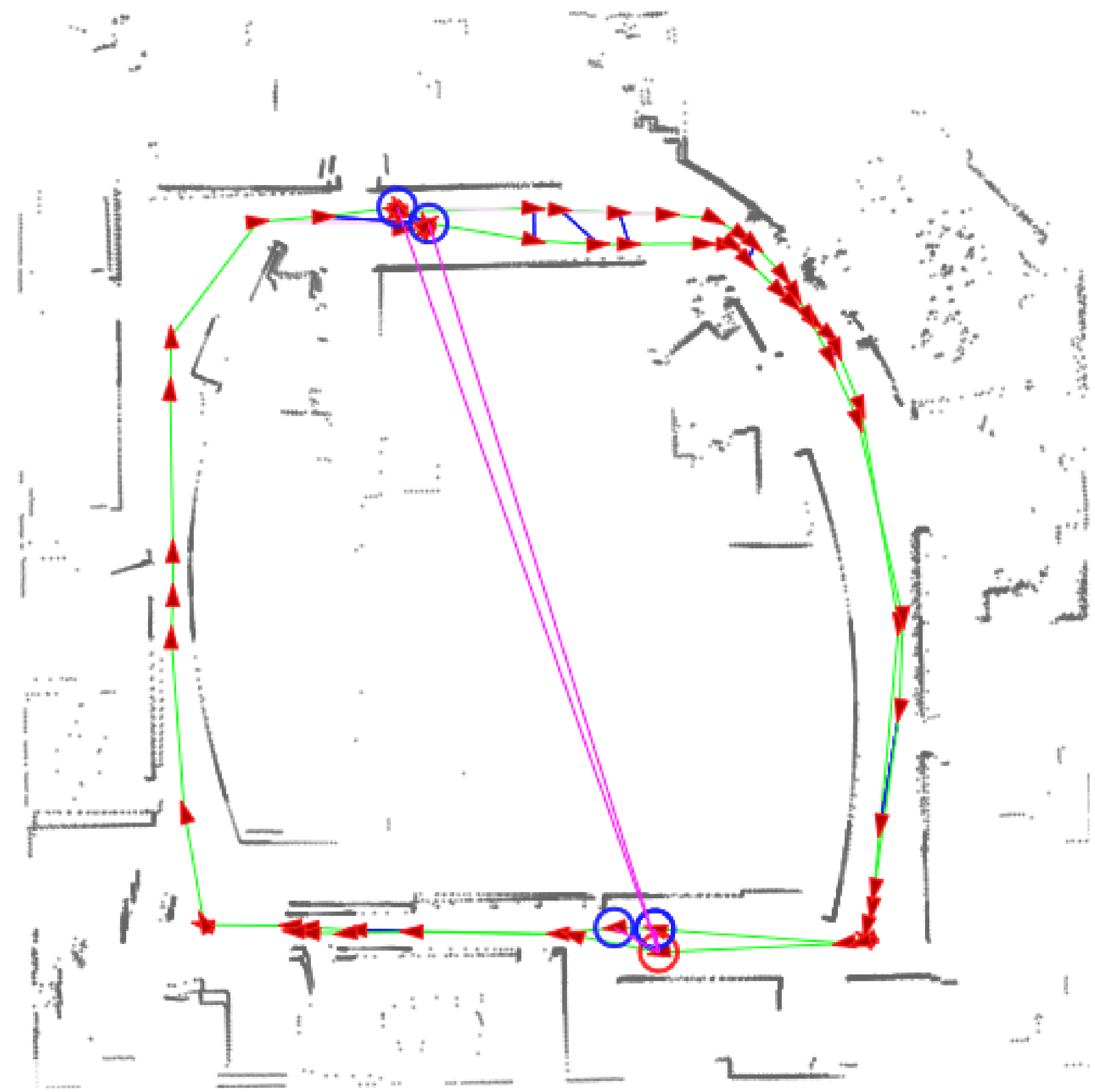


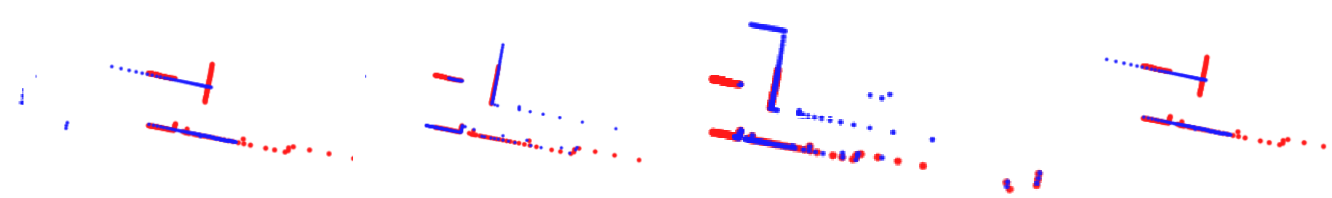
“Extreme SLAM”



Eliminating the front-end

- Create a max-mixture with N+1 components
  - Best N matches (based on scan matching)
  - One "null" component.
  - (Encodes an interesting mutual-exclusion property!)





- No loop validation, no geometric constraints.

Eliminating the front-end

Open-Loop

With Unvalidated Loop Closures

Performance Analysis

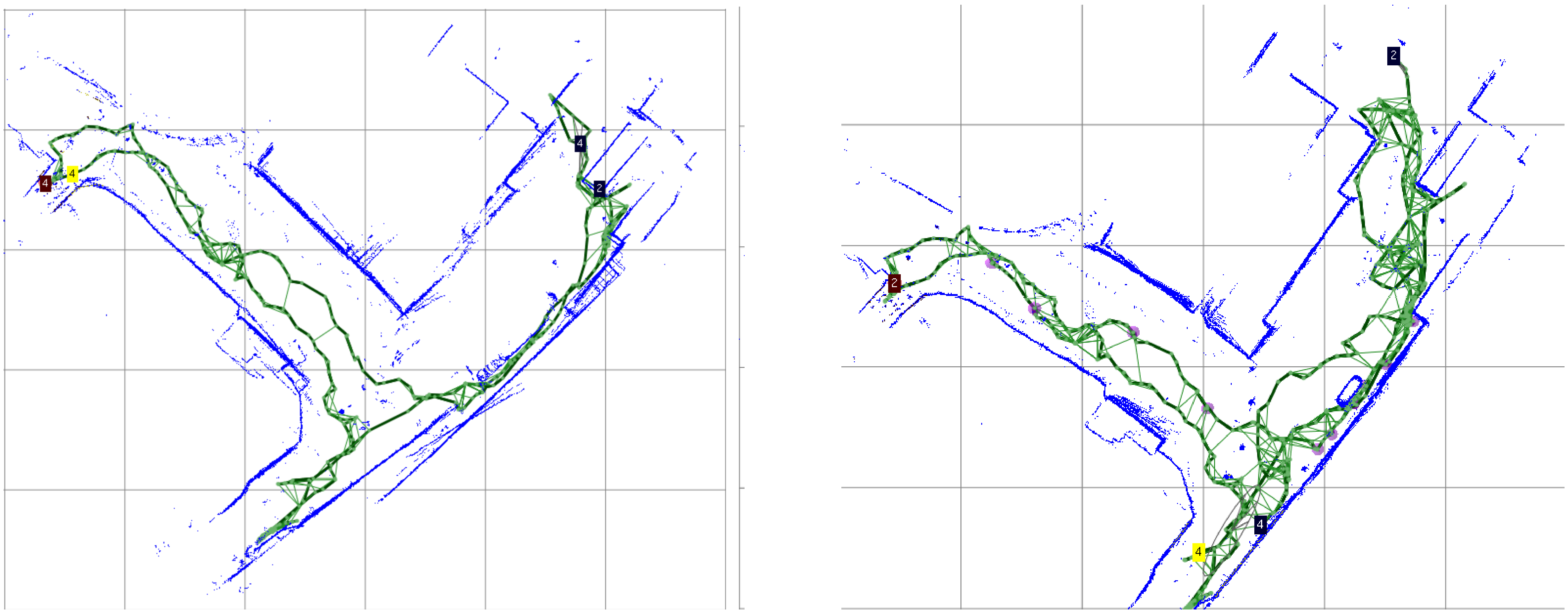
- Suppose we have N hypotheses relating a pose to earlier poses (like previous problem)
  - Could use ONE max mixture with N+1 Components
  - Could use N max mixtures with 2 Components
- Which one is better?

Effect of encoding on CPU time

Dataset		Switchable constraints	bi-modal MM	k-modal MM
manhattan	iter time (s)	0.90 s	0.74 s	0.13 s
with $k = 2$	fill-in (%)	1.50 %	2.89 %	0.17 %
outliers = 2099	#loop edges	4198	4198	2099
	#components	-	2	3
manhattan	iter time (s)	1.5 s	1.2 s	0.13 s
with $k = 3$	fill-in (%)	1.70 %	4.30 %	0.17 %
outliers= 4198	#loop edges	6277	6277	2099
	#components	-	2	4

- N+1 component mixtures exhibit much better scaling!
  - Max mixture selects *one* component to be dominant
  - Thus, has fewer "links" between poses...
  - Sparser A matrix ==> faster matrix factoring methods.

Good things lead to more good things



- A system that has accumulates errors has trouble closing new loops.
- Robustly handling errors not only avoids divergence on the loops you have, it increases the number of loops you'll find!

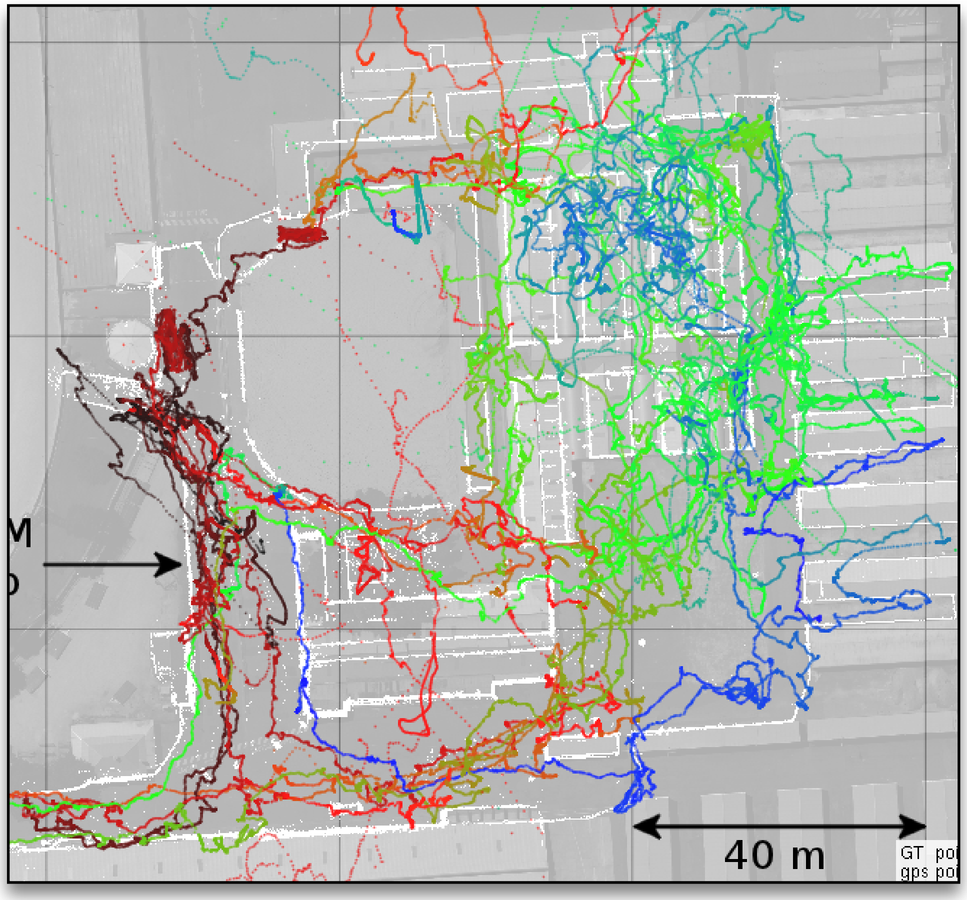
Some Perspective

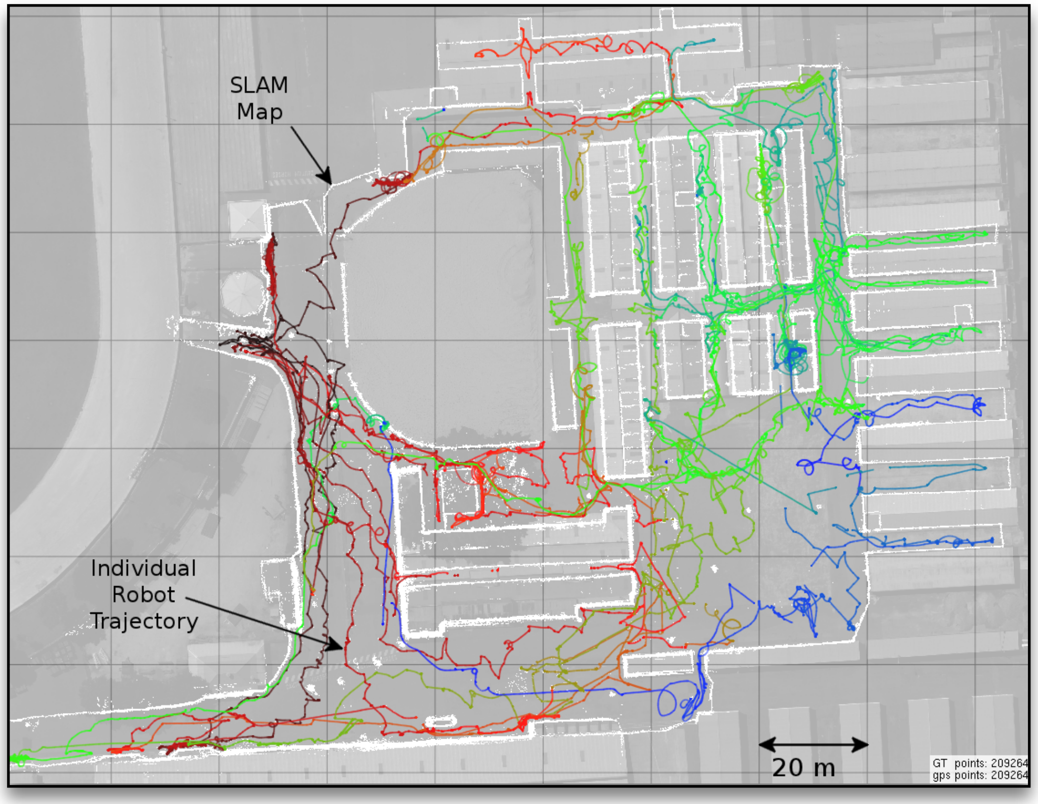
- Other approaches exist for multi-modal inference.
  - FastSLAM (Montemerlo)
  - Multiple Hypothesis Tracking
- Both have complexity that scales with the complexity of the *posterior*
  - FastSLAM: Particles
  - MHT: Hypotheses (each with an EKF)
- The complexity of the posterior grows exponentially, forcing these methods to prune. Can lead to failures (e.g. particle depletion).
- MaxMixtures is fundamentally different:
  - Memory complexity grows linearly with the size of the problem.
  - Never have to approximate the problem.
  - But, no guarantee that we find the maximum-likelihood solution!

Learning GPS Covariances

- Typically very hard to get good covariance estimates
  - Multi-path / Urban canyons
  - Indoor/Outdoor transitions

- Lots of interesting meta-data about sensor observations, e.g.:
  - # visible satellites
  - HDOP (based on geometry of satellites)
  - vendor’s covariance estimate





Can we learn covariances?

- Idea:
  - Construct a feature vector **f** from this meta-data
  - Learn weight vector **w** such that:

$$\sigma = w^T f$$

Feature Encoding

- Constant feature:

$$f^T = [1]$$

- Add HDOP:

$$f^T = [\ z_{hdop} \ \ 1]$$

- Add # satellites

$$f^T = [\ \underbrace{0 \quad \ldots \quad 0 \quad 1 \quad 0 \quad \ldots \quad 0}_{\text{One-Hot encoding}} \quad z_{hdop} \quad 1\ ]$$

Other feature types

- Idea: Generate features from other sensor modalities
  - Estimate “indoorness” from LIDAR data?

- (Not doing that here, but it’s something we’re looking at)

Extension to Max Mixture

- Learning weights **w** tells us covariance

$$\sigma = w^T f$$

- Extension to max mixture is easy:
  - Fit multiple sigmas. (Assume means are the same)

$$\sigma_i = w_i^T f$$

$$p(z|x) = \max_i \alpha_i N(u, \sigma_i)$$

  - And learn mixing weights (alphas) too.

Evaluating the learned weights

- Standard approach:
  - Pick w's that maximize the likelihood of the data

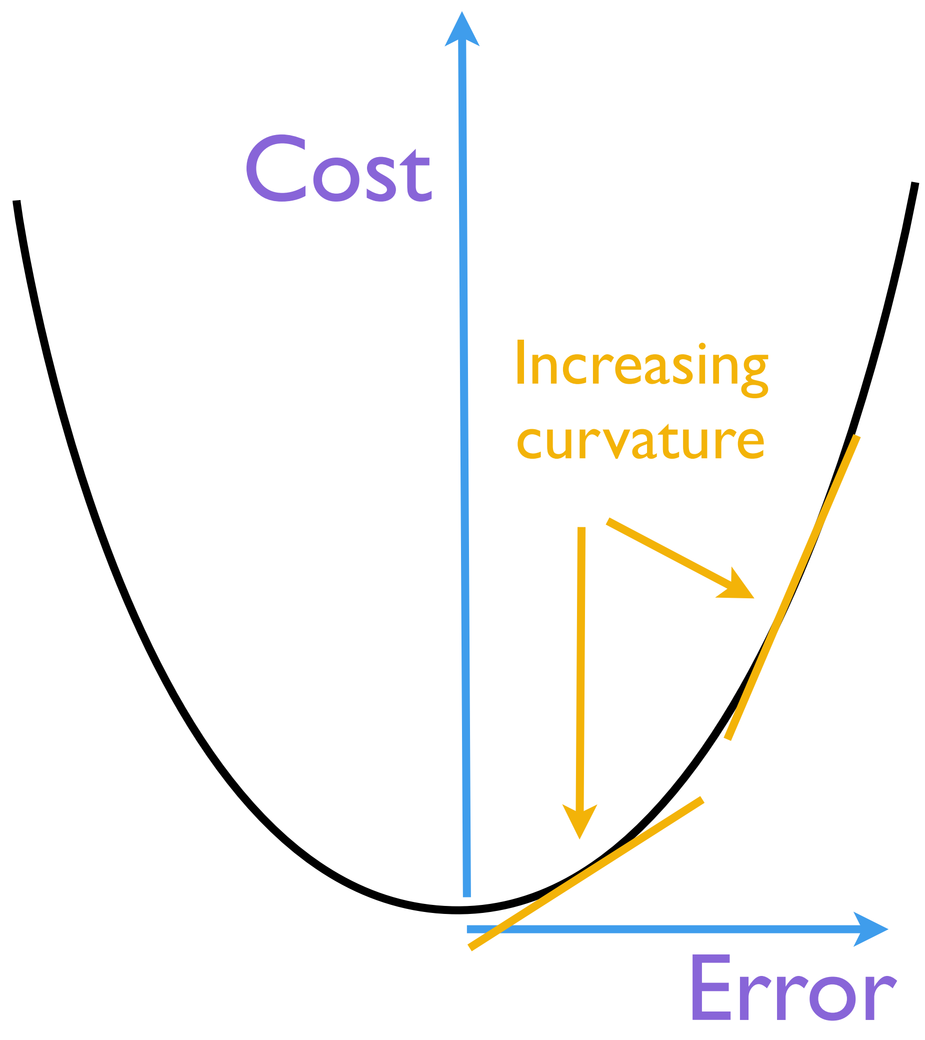
$$
\begin{aligned}
w^* &= \operatorname{maxarg}_w \prod_j p(z_j|x) \\
&\propto \operatorname{maxarg}_w \prod_j e^{-\frac{1}{2}(z_j-\mu)^T (w^T f)^{-2} (z_j-\mu)}
\end{aligned}
$$

(shown here for standard Gaussian approach)

- (i.e., what weights **w** maximize the likelihood of all the observations?)

Non-mixture analysis

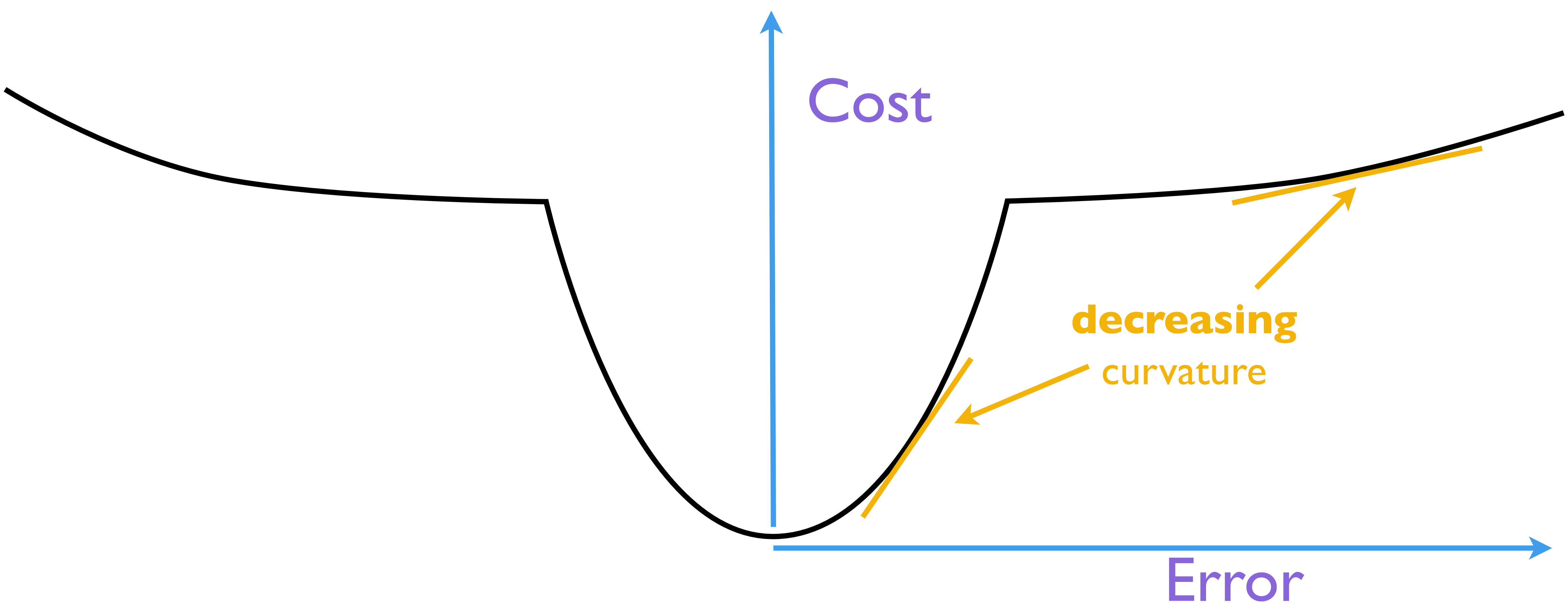
- The ML solution w* is good for two reasons:
  - It's the ML solution, and we're all good Bayesians, right?
  - It minimizes the occurrence of high X^2 observations
    - These have a increasing effect on the gradient in an optimization framework...
    - ... and are responsible for divergence of non-robust methods.



- I.e., In Gaussian case, low probability ==> high cost function curvature ==> divergence

Max Mixture Analysis

- It's *not* the case that low probability => high curvature => divergence.
- E.g., "null hypothesis" components: low weight (==> low probability) but high variance (==> low curvature)



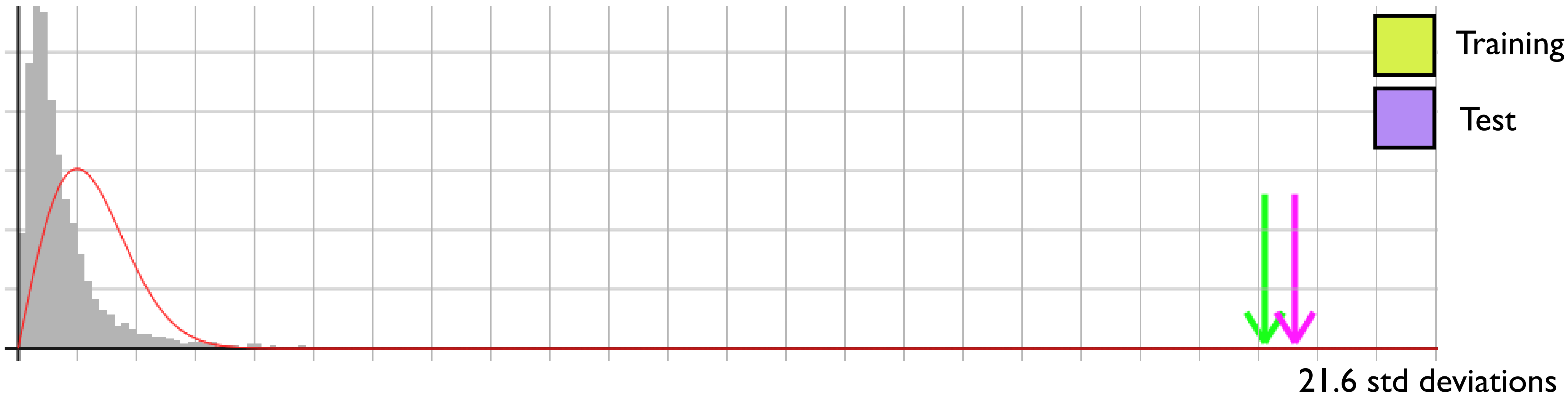
Evaluation

- So how do we evaluate a max mixture?
  - "Model Goodness": Maximum Likelihood
  - Convergence: minimize gradient for bad data

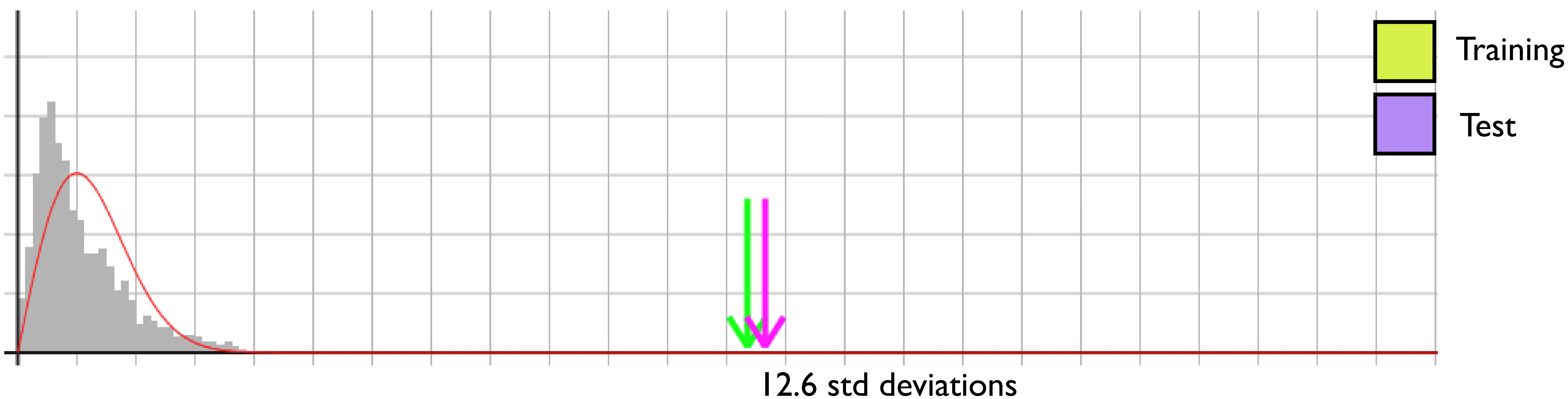
- Our current thinking:
  - Still maximize the likelihood of the data, but...
  - Keep an eye on the gradients as an interesting check...

Constant model, f = [1]

- Single Gaussian

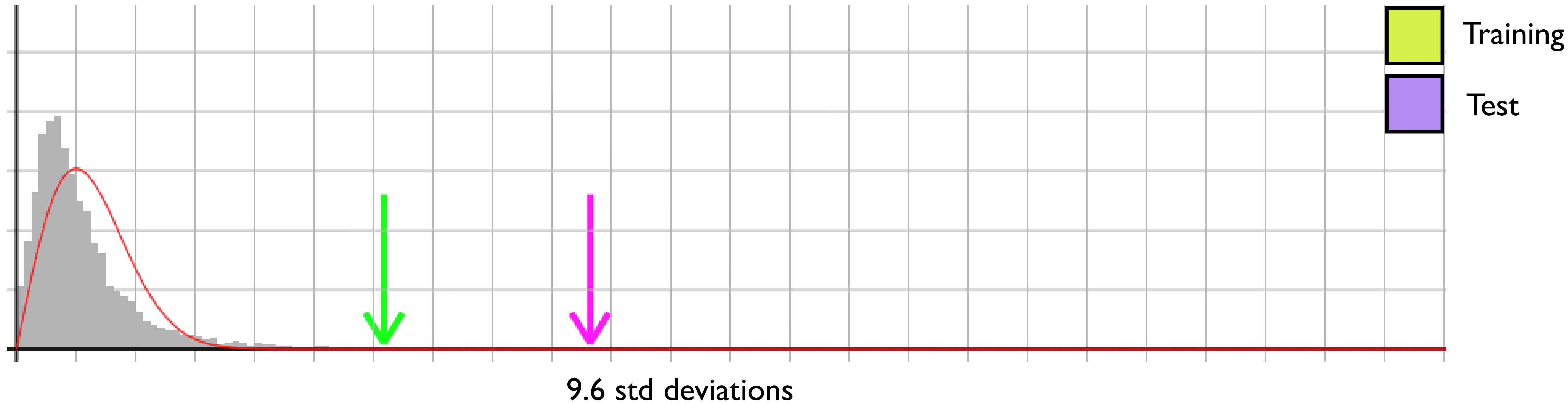


- Max Mixture of two Gaussians

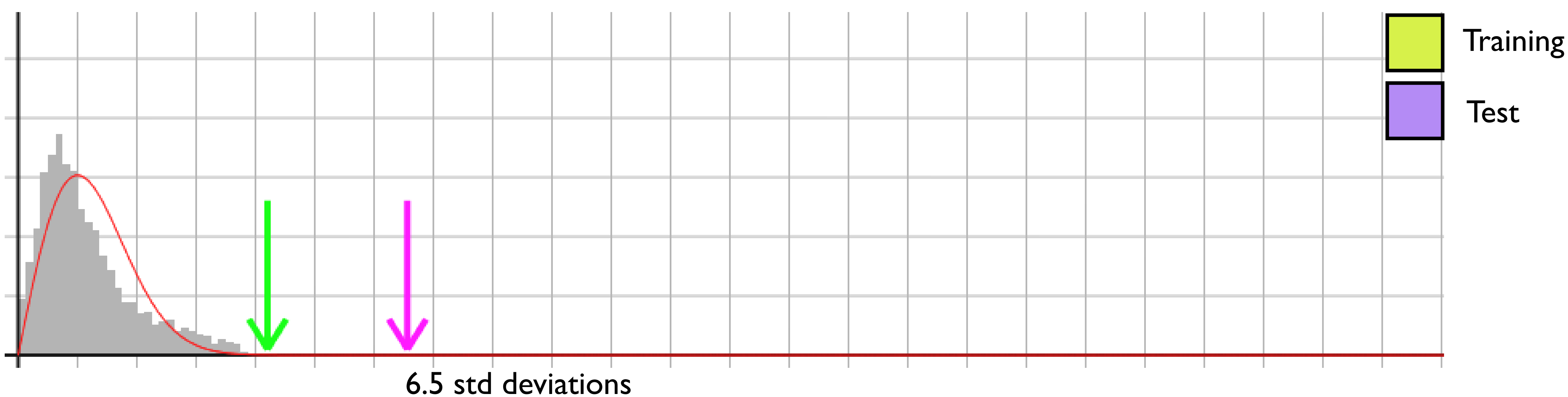


Vendor+ model, f=[v]

- Single Gaussian

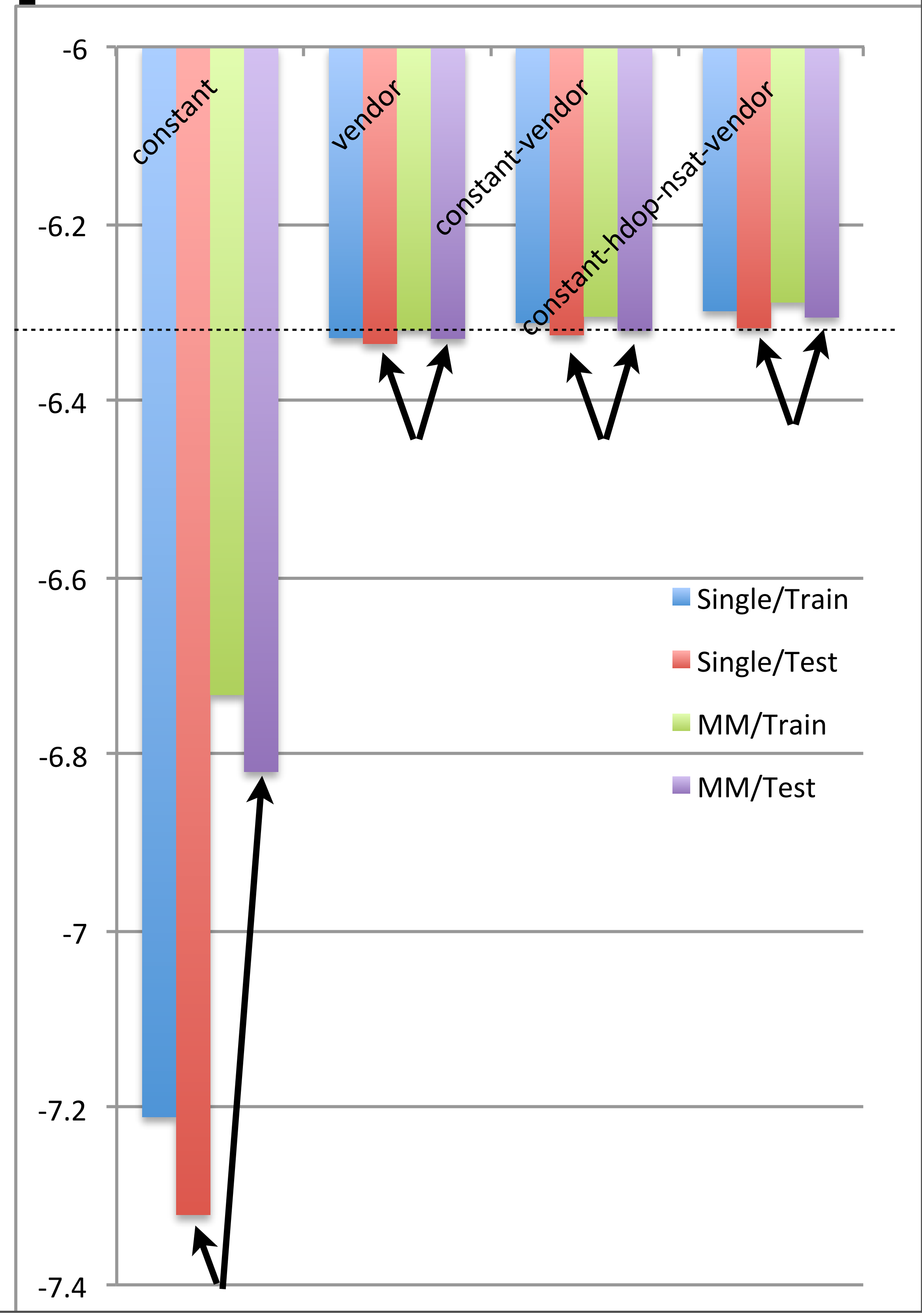


- Max Mixture of two Gaussians



Model Comparison

- Results:
    - More complex models ==> better predictions in an ML sense
    - Two component mixture model yields higher likelihood
- NB: Small numerical differences here are a big deal...
    - These are average likelihoods over thousands of observations.



Gradients

- Generally see lower worst-case gradient magnitudes

- Interestingly, it's not the same observations causing "problems" in both cases
  - The high gradient observations are those just barely clinging to the "inliner" component.

	Non-MM max gradient	MM max gradient
Constant	3.251	1.114
Vendor	2.331	2.024
Constant-Vendor	1.878	1.791
Constant-HDop-NSat-Vendor	2.127	1.836

Something Outrageous

- What do I care about?
  - Robustness (to outliers) (to initial estimates)
  - Where do error-models/hyper-parameters come from?

- What do I care less about?
  - Inference speed
  - Batch problems