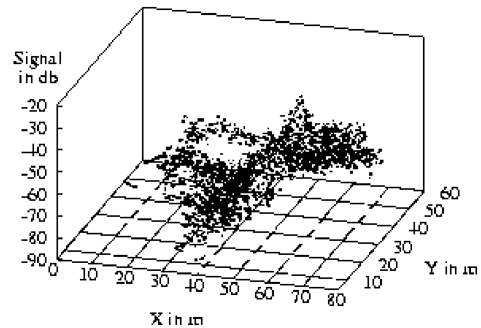
CSE 571
Probabilistic Robotics

Gaussian Processes

WiFi Sensor Model



High-level Idea

- Non-parametric regression model
- Distribution over functions
- Fully specified by training data and kernel function
- Output variables are jointly Gaussian
- Covariance given by distance of inputs in kernel space

GP Setting

- Outputs are noisy function of inputs:

$$y_i = f(\mathbf{x}_i) + \varepsilon$$

- Function values are jointly Gaussian:

$$\mathrm{cov}\left(f(\mathbf{x}_p), f(\mathbf{x}_q)\right) = k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}|\mathbf{x}_p - \mathbf{x}_q|^2\right)$$

- Considering noise:

$$\mathrm{cov}\left(y_p, y_q\right) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq}$$

$$p(\mathbf{Y} \mid \mathbf{X}) = \mathrm{N}(\mathbf{0}, K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})$$

GP Prediction

- Training data:

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} = (\mathbf{X}, \mathbf{y})$$

- Prediction given training samples:

$$p(y^* \mid \mathbf{x}^*, \mathbf{y}, \mathbf{X}) = N\left(\mu, \sigma^2\right)$$

$$\mu = K(\mathbf{X}^*, \mathbf{X}) \left(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}$$

$$\sigma^2 = K(\mathbf{x}^*, \mathbf{x}^*) - K(\mathbf{x}^*, \mathbf{X}) \left(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} K(\mathbf{X}, \mathbf{x}^*)$$

Hyperparameter Estimation

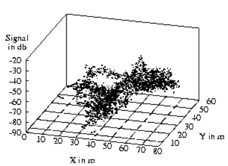
- Maximize data log likelihood:

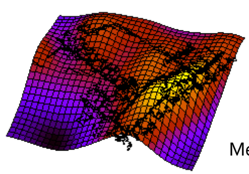
$$\log p(\mathbf{y} \mid \mathbf{X}) =$$

$$-\frac{1}{2}\mathbf{y}^T \left(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y} - \frac{1}{2}\log\left|K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}\right| - \frac{n}{2}\log 2\pi$$

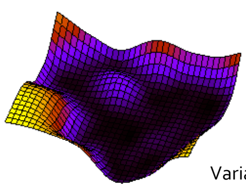
- Compute derivatives wrt. params $\theta = \langle \sigma_n^2, l, \sigma_f^2 \rangle$
- Optimize using conjugate gradient

Gaussian Process Sensor Model





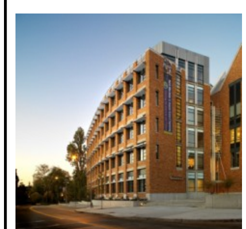
Mean

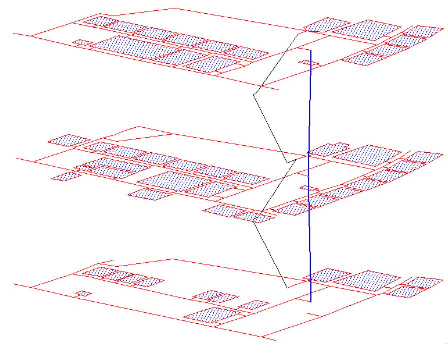


Variance

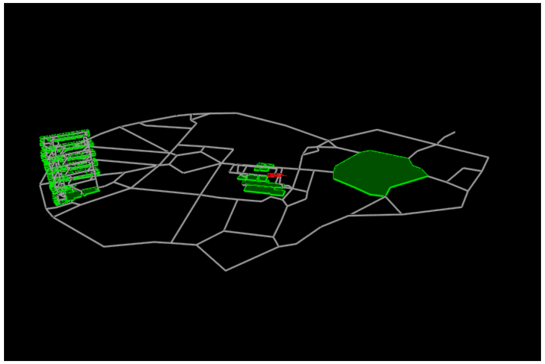
[Ferris-Haehnel-Fox: RSS-06]

Mixed Representation



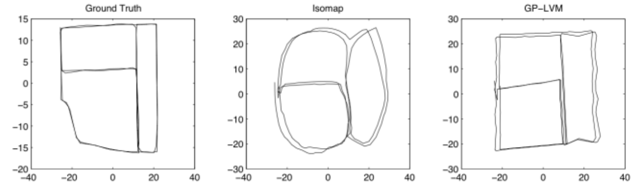


Example



WiFi-SLAM: Mapping without Ground Truth

[Ferris-Fox-Lawrence: IJCAI-07]



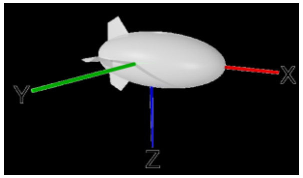
- GP-LVM: GP with latent / unobserved variables (locations)
- Can incorporate motion constraints

GP for Dynamic Systems

- Controller development benefits from accurate model
- Two approaches to system modeling
  - Parametric / physics-based models
  - Non-parametric / data-driven models
- Combining these two approaches yields superior model

Experimental Testbed

Non-linear Parametric Model



$$\dot{s} = \frac{d}{dt}\begin{bmatrix} p \\ \xi \\ v \\ \omega \end{bmatrix} = \begin{bmatrix} R_b^e v \\ H(\xi) \\ M^{-1}(\sum Forces - \omega * Mv) \\ J^{-1}(\sum Torques - \omega * J\omega) \end{bmatrix}$$

- 12-D state=[pos,rot,transvel,rotvel]
- Describes evolution of state as ODE
- Forces / torques considered: buoyancy, gravity, drag, thrust
- 16 parameters are learned by optimization on ground truth motion capture data

Non-linear Parametric Model

- ODE can be used to generate a step ahead prediction function f

$$s(k+1) = s(k) + f(s(k), u(k))$$

- Problems
  - Limited accuracy
  - Noise not explicit in model

GP System Model

- State transition learned directly

$$T(k) = s^{gt}(k+1) - s^{gt}(k)$$

- Training data for GP:

$$D_{GP} = \{[s^{gt}(k), u(k)], T(k)\}$$

- GP prediction

$$s(k+1) = s(k) + g([s(k), u(k)])$$

- Problems:
  - Generalizes poorly
  - Full coverage of state space difficult

Enhanced-GP model

- Target output takes parametric model f into account

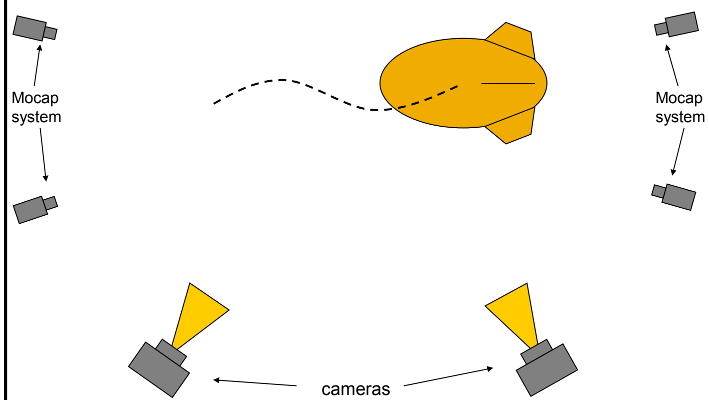
$$T(k) = s^{gt}(k+1) - s^{gt}(k) - f(s^{gt}(k), u(k))$$

- Enhanced-GP model equation

$$s(k+1) = s(k) + f(s(k), u(k)) + g_{EGP}([s(k), u(k)])$$

- Better accuracy
- Less training data necessary
- Noise incorporated into system

Experimental Setup



Observation model

- State: xyz pitch yaw
- Observation: xc yc width height theta
- Models: Parametric using computer graphics/vision, GP, EGP

Prediction error (process)

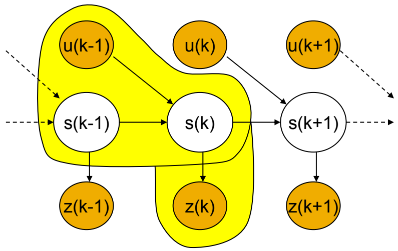
Propagation method	pos(mm)	rot(deg)	vel(mm/s)	rotvel(deg/s)
Param	3.3	0.5	14.6	1.5
GPonly	1.8	0.2	9.8	1.1
EGP	1.6	0.2	9.6	1.3

- Single step prediction error
- ¼ sec timesteps
- Avg over ~1000 test points

Prediction error (observation)

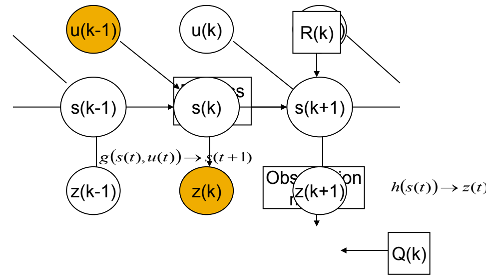
Modeling method	pos(pix)	Major axis(pix)	Minor axis(pix)	Theta(deg)
Param	7.1	2.9	5.7	9.2
Gponly	4.7	3.2	1.9	9.1
EGP	3.9	2.4	1.9	9.4

Incorporation into Filtering



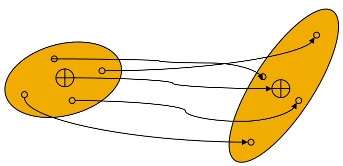
- Sequential state estimation
- Prediction step: $p(s_{t+1} \mid s_t, u_t)$
- Correction step: $p(z_t \mid s_t)$

Kalman filter



- Linear dynamical system
- Extended-KF / Unscented-KF: Locally linearized state propagation and observation models

Linearization via Unscented Transform



$$\chi = (\mu \quad \mu + \gamma\sqrt{\Sigma} \quad \mu - \gamma\sqrt{\Sigma})$$

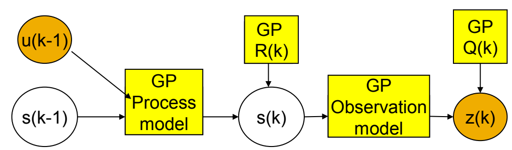
$$\text{for } i = 0 \ldots 2n: \chi'^{i} = g(u, \chi^{i})$$

$$\mu' = \sum_{i=0}^{2n} \omega_m^i \chi'^i$$

$$\Sigma' = \sum_{i=0}^{2n} \omega_c^i (\chi'^i - \mu')(\chi'^i - \mu')^T$$

- Determine sigma points based on covariance
- Propagate using nonlinear function *g*
- Use propagated sigma points to recreate mean and covariance

GP-UKF



- Use GP process and observation models
- Replace static noise parameters with uncertainty from GP

GP-UKF algorithm

- GP-UKF($\mu, \Sigma, \mathbf{u}, \mathbf{z}$):

-Determine sigma points
- $X_k^i = GP_\mu^g(X_{k-1}^i, \mathbf{u}_{k-1})$

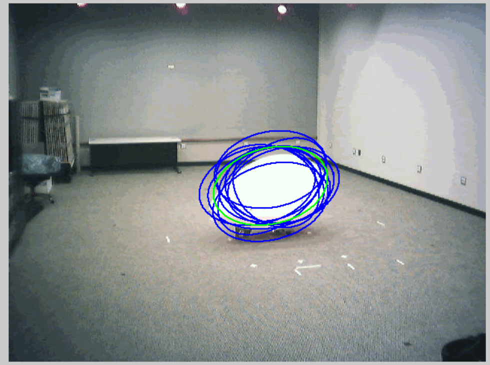
-Recover new mean and sigma
- $R_k = GP_\Sigma^g(\mu_{k-1}, \mathbf{u}_{k-1})$

-Determine sigma points
- $\hat{Z}_k^i = GP_\mu^h(\hat{X}_k^i)$

-Recover new mean and sigma
- $Q_k = GP_\Sigma^h(\hat{\mu}_k)$

-Perform correction

GP-UKF Tracking Example



Modeling accuracy

Prediction error

Propagation method	pos(mm)	rot(deg)	vel(mm/s)	rotvel(deg/s)
Param	3.3	0.5	14.6	1.5
GPonly	1.8	0.2	9.8	1.1
EGP	1.6	0.2	9.6	1.3

Observation error

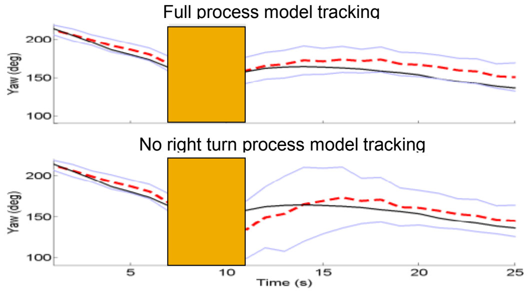
Modeling method	pos(pix)	Major axis(pix)	Minor axis(pix)	Theta(deg)
Param	7.1	2.9	5.7	9.2
GPonly	4.7	3.2	1.9	9.1
EGP	3.9	2.4	1.9	9.4

GP-UKF tracking accuracy

Tracking algorithm	pos(mm)	rot(deg)	vel(mm/s)	rotvel(deg/s)	MLL
UKF	141	9.6	141.5	8.1	2.1
GP-UKF (GPonly)	107.9	10.2	71.7	5.9	5.1
GP-UKF (EGP)	86	6.1	57.1	5.7	12.9

- Average tracking error
- Trajectory ~12 min long
- 0.5 sec timesteps

Dealing with training data sparsity



- Training data for right turns removed
- Uncertainty increases appropriately

Summary

- GPs provide flexible modeling framework
- Take data noise and uncertainty due to data sparsity into account
- Combination with parametric models increases accuracy and reduces need for training data
- Seamless integration into Bayes filters
- Compexity is problem:
  - Training: $O(n^3)$ Prediction $O(n^2)$

Further Issues

- Complexity can be reduced by removing / ignoring data points (sparse GP)
- Input dependent signal noise (heteroscedastic GP)
- Input dependent kernel parameters
- Can be used for dimensionality reduction (e.g. GP-LVM)
- Uncertainty provides means for active exploration and optimal sensor placement