CS491/691: Introduction to Aerial Robotics

Topic: Recapitulate 2 – Advanced Topics

Dr. Kostas Alexis (CSE)

Contents

- We will recapitulate selected topics in:
  - Kalman Filter
  - Extended Kalman Filter
  - Exploration Path Planning

CS491/691: Introduction to Aerial Robotics

Topic: State Estimation – Kalman Filter

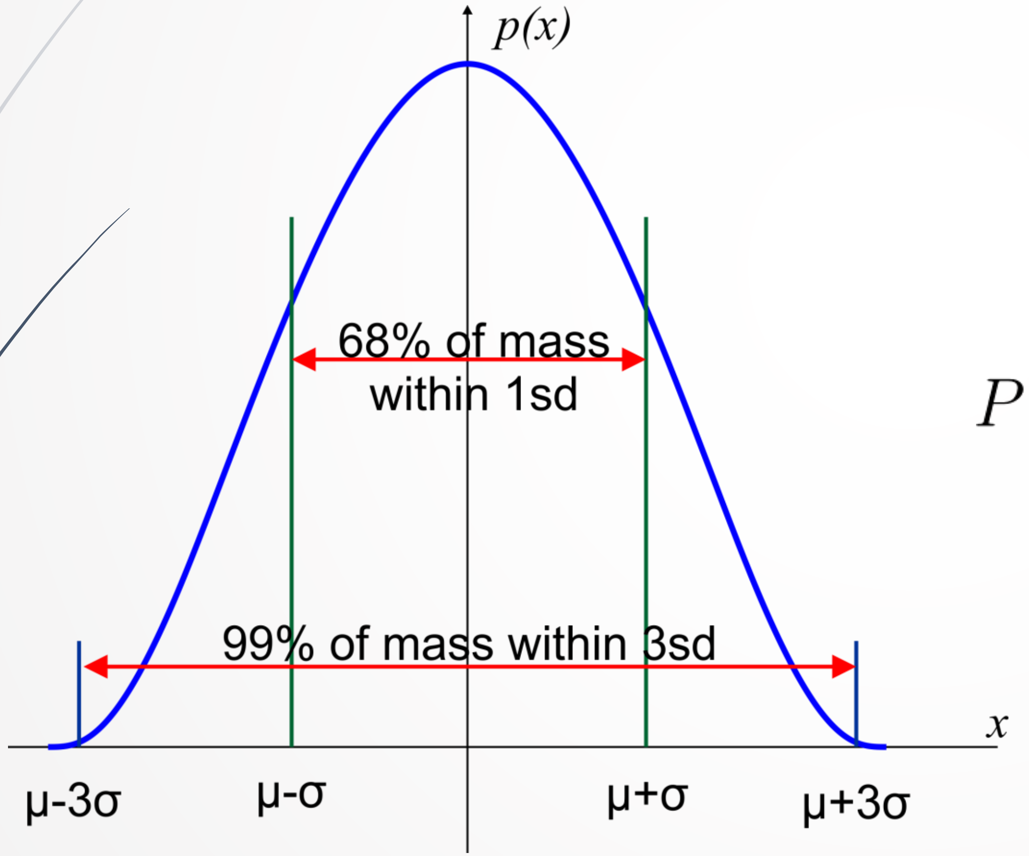
Dr. Kostas Alexis (CSE)

Kalman Filter

- Bayes filter is a useful tool for state estimation.
- Histogram filter with grid representation is not very efficient.
- How can we represent the state more efficiently?

Kalman Filter

- Univariate distribution



$$X \sim \mathcal{N}(\mu, \sigma^2)$$

mean

Variance (squared standard deviation)

$$P(X = x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2})$$

Kalman Filter

- Multivariate normal distribution: $\mathbf{X} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$
- Mean: $\mu \in \mathcal{R}^n$
- Covariance: $\mathbf{\Sigma} \in \mathbf{R}^{n \times m}$
- Probability density function:

$$p(\mathbf{X} = \mathbf{x}) = \mathcal{N}(\mathbf{x}; \mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu))$$

Properties of Normal Distributions

- Linear transformation – remains Gaussian

$$\mathbf{X} \sim \mathcal{N}(\mu, \mathbf{\Sigma}), \mathbf{Y} \sim \mathbf{A}\mathbf{X} + \mathbf{B}$$
$$\Rightarrow \mathbf{Y} \sim \mathcal{N}(\mathbf{A}\mu + \mathbf{B}, \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T)$$

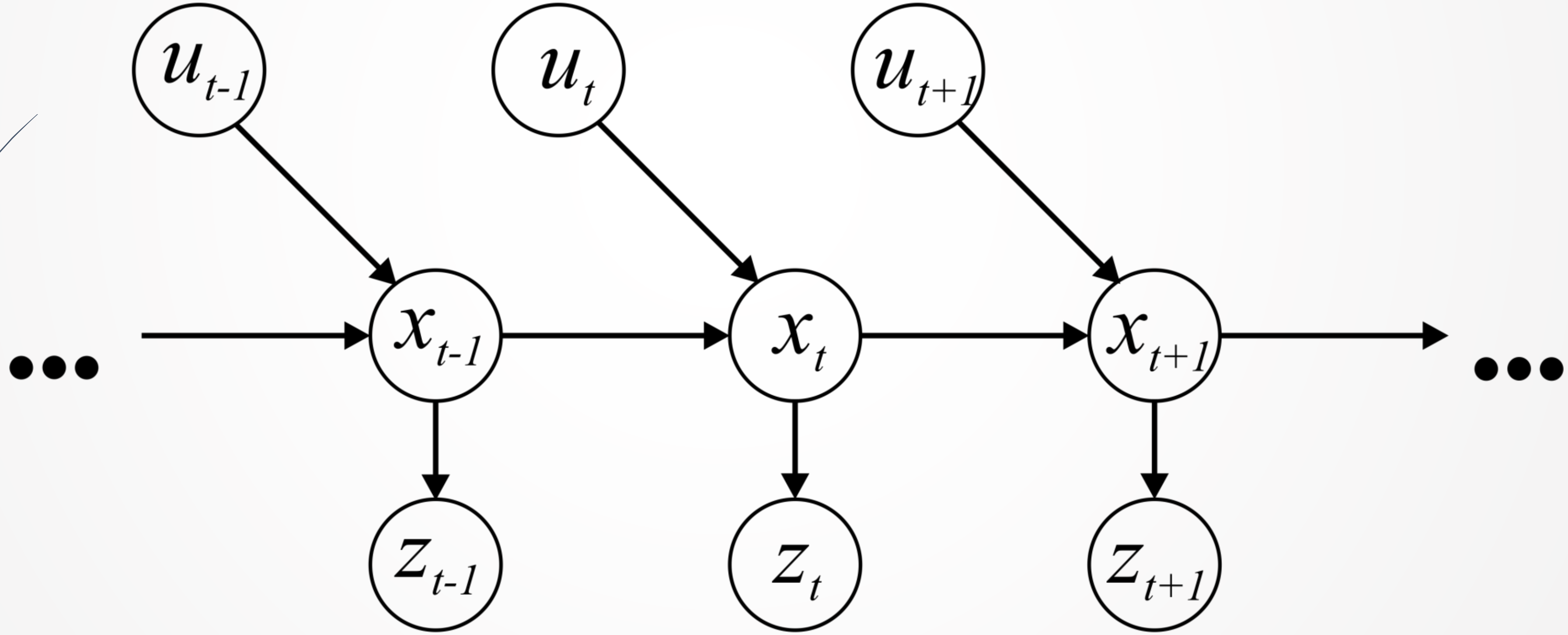
- Intersection of two Gaussians – remains Gaussian

$$\mathbf{X}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1), \mathbf{X}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \mathbf{\Sigma}_2)$$

$$\boxed{p(\mathbf{X}_1)p(\mathbf{X}_2) = \mathcal{N}\left(\frac{\mathbf{\Sigma}_2}{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}\boldsymbol{\mu}_1 + \frac{\mathbf{\Sigma}_1}{\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2}\boldsymbol{\mu}_2, \frac{1}{\mathbf{\Sigma}_1^{-1} + \mathbf{\Sigma}_2^{-1}}\right)}$$

Linear Process Model

- Consider a time-discrete stochastic process (Markov chain)



Linear Process Model

- Consider a time-discrete stochastic process
- Represent the estimated state (belief) with a Gaussian

$$\mathbf{x}_t \sim \mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$$

Linear Process Model

- Consider a time-discrete stochastic process
- Represent the estimated state (belief) with a Gaussian

$$\mathbf{x}_t \sim \mathcal{N}(\mu_t, \mathbf{\Sigma}_t)$$

- Assume that the system evolves linearly over time, then depends linearly on the controls, and has zero-mean, normally distributed process noise

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \epsilon_t$$

- With $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$

Linear Observations

- Further, assume we make observations that depend linearly on the state and that are perturbed zero-mean, normally distributed observation noise

$$\mathbf{z}_t = \mathbf{C}\mathbf{x}_t + \delta_t$$

- With $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$

Kalman Filter

- Estimates the state x_t of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \epsilon_t$$

- And (linear) measurements of the state

$$\mathbf{z}_t = \mathbf{C}\mathbf{x}_t + \delta_t$$

- With $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\delta_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$

Kalman Filter

- State $\mathbf{x} \in \mathbb{R}^n$
- Controls $\mathbf{u} \in \mathbb{R}^l$
- Observations $\mathbf{z} \in \mathbb{R}^k$
- Process equation $\mathbf{x}_t = \underbrace{\mathbf{A}}_{nxn}\mathbf{x}_{t-1} + \underbrace{\mathbf{B}}_{nxl}\mathbf{u}_t + \epsilon_t$
- Measurement equation $\mathbf{z}_t = \underbrace{\mathbf{C}}_{nxk}\mathbf{x}_t + \delta_t$

Kalman Filter

- Initial belief is Gaussian

$$Bel(x_0) = \mathcal{N}(\mathbf{x}_0; \mu_0, \mathbf{\Sigma}_0)$$

- Next state is also Gaussian (linear transformation)

$$\mathbf{x}_t \sim \mathcal{N}(\mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t, \mathbf{Q})$$

- Observations are also Gaussian

$$\mathbf{z}_t \sim \mathcal{N}(\mathbf{C}\mathbf{x}_t, \mathbf{R})$$

Recall: Bayes Filter Algorithm

- For each step, do:
  - Apply motion model

$$\overline{Bel}(\mathbf{x}_t) = \int p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) Bel(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}$$

  - Apply sensor model

$$Bel(\mathbf{x}_t) = \eta p(\mathbf{z}_t | \mathbf{x}_t) \overline{Bel}(\mathbf{x}_t)$$

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply motion model

$$\overline{Bel}(\mathbf{x}_t) = \int \underbrace{p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)}_{\mathcal{N}(\mathbf{x}_t;\mathbf{A}\mathbf{x}_{t-1}+\mathbf{B}\mathbf{u}_k t,\mathbf{Q})} \underbrace{Bel(\mathbf{x}_{t-1})}_{\mathcal{N}(\mathbf{x}_{t-1};\mu_{t-1},\mathbf{\Sigma}_{t-1})} d\mathbf{x}_{t-1}$$

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply motion model

$$\overline{Bel}(\mathbf{x}_t) = \int \underbrace{p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t)}_{\mathcal{N}(\mathbf{x}_t; \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_k t, \mathbf{Q})} \underbrace{Bel(\mathbf{x}_{t-1})}_{\mathcal{N}(\mathbf{x}_{t-1}; \mu_{t-1}, \mathbf{\Sigma}_{t-1})} d\mathbf{x}_{t-1}$$

$$= \mathcal{N}(\mathbf{x}_t; \mathbf{A}\mu_{t-1} + \mathbf{B}\mathbf{u}_t, \mathbf{A}\Sigma\mathbf{A}^T + \mathbf{Q})$$

$$= \mathcal{N}(\mathbf{x}_t; \bar{\mu}_t, \bar{\Sigma}_t)$$

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply sensor model

$$\overline{Bel}(\mathbf{x}_t) = \eta \; \underbrace{p(\mathbf{z}_t | \mathbf{x}_t)}_{\mathcal{N}(\mathbf{z}_t;\mathbf{C}\mathbf{x}_t,\mathbf{R})} \; \underbrace{\overline{Bel}(\mathbf{x}_t)}_{\mathcal{N}(\mathbf{x}_t;\bar{\mu}_t,\bar{\mathbf{\Sigma}}_t)}$$

$$= \mathcal{N}(\mathbf{x}_t; \bar{\mu}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\mu}), (\mathbf{I} - \mathbf{K}_t)\mathbf{C})\bar{\mathbf{\Sigma}})$$

$$= \mathcal{N}(x_t; \mu_t, \mathbf{\Sigma}_t)$$

- With $\mathbf{K}_t = \bar{\mathbf{\Sigma}}_t \mathbf{C}^T (\mathbf{C}\bar{\mathbf{\Sigma}}_t \mathbf{C}^T + \mathbf{R})^{-1}$ (**Kalman Gain**)

From Bayes Filter to Kalman Filter

Blends between our previous estimate $\bar{\mu}_t$ and the discrepancy between our sensor observations and our predictions.

The degree to which we believe in our sensor observations is the Kalman Gain. And this depends on a formula based on the errors of sensing etc. In fact it depends on the ratio between our uncertainty Σ and the uncertainty of our sensor observations R.

$$\underbrace{\boxed{\bar{\mu}_t}}_{\text{old mean}} + \underbrace{\boxed{\mathbf{K}_t}}_{\text{Kalman Gain}}(\mathbf{z}_t - \mathbf{C}\bar{\mu})$$

From Bayes Filter to Kalman Filter

- For each step, do:
  - Apply sensor model

$$\overline{Bel}(\mathbf{x}_t) = \eta \underbrace{p(\mathbf{z}_t|\mathbf{x}_t)}_{\mathcal{N}(\mathbf{z}_t;\mathbf{C}\mathbf{x}_t,\mathbf{R})} \underbrace{\overline{Bel}(\mathbf{x}_t)}_{\mathcal{N}(\mathbf{x}_t;\bar{\mu}_t,\bar{\mathbf{\Sigma}}_t)}$$

$$= \mathcal{N}(\mathbf{x}_t; \bar{\mu}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\mu}), (\mathbf{I} - \mathbf{K}_t)\mathbf{C})\bar{\mathbf{\Sigma}})$$

$$= \mathcal{N}(x_t; \mu_t, \mathbf{\Sigma}_t)$$

- With $\mathbf{K}_t = \bar{\mathbf{\Sigma}}_t \mathbf{C}^T (\mathbf{C}\bar{\mathbf{\Sigma}}_t\mathbf{C}^T + \mathbf{R})^{-1}$ (**Kalman Gain**)

Kalman Filter Algorithm

- For each step, do:
  - Apply motion model (prediction step)

$$\bar{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_t$$

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top + \mathbf{Q}$$

  - Apply sensor model (correction step)

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\boldsymbol{\mu}}_t)$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\bar{\boldsymbol{\Sigma}}_t$$

    - With $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t\mathbf{C}^\top(\mathbf{C}\bar{\boldsymbol{\Sigma}}_t\mathbf{C}^\top + \mathbf{R})^{-1}$

Kalman Filter Algorithm

Prediction & Correction steps can happen in any order.

- For each step, do:
  - Apply motion model (**prediction step**)

$$\bar{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_t$$

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top + \mathbf{Q}$$

  - Apply sensor model (**correction step**)

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\boldsymbol{\mu}}_t)$$

$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\bar{\boldsymbol{\Sigma}}_t$$

  - With $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t\mathbf{C}^\top(\mathbf{C}\bar{\boldsymbol{\Sigma}}_t\mathbf{C}^\top + \mathbf{R})^{-1}$

Kalman Filter Algorithm

Prediction & Correction steps can happen in any order.

Prediction

$$\bar{\boldsymbol{\mu}}_t = \mathbf{A}\boldsymbol{\mu}_{t-1} + \mathbf{B}\mathbf{u}_t$$

$$\bar{\Sigma}_t = \mathbf{A}\Sigma\mathbf{A}^\top + \mathbf{Q}$$

Correction

$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}\bar{\boldsymbol{\mu}}_t)$$

$$\Sigma_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C})\bar{\Sigma}_t$$

$$\mathbf{K}_t = \bar{\Sigma}_t\mathbf{C}^\top(\mathbf{C}\bar{\Sigma}_t\mathbf{C}^\top + \mathbf{R})^{-1}$$

Complexity

- Highly efficient: Polynomial in the measurement dimensionality k and state dimensionality n

$$O(k^{2.376} + n^2)$$

- Optimal for linear Gaussian systems
  - But most robots are nonlinear! This is why in practice we use Extended Kalman Filters and other approaches.

KF: Indicative Questions

- Describe the Kalman Filter for a linear process of the form $\dot{x} = Ax + Bu$
- Explain the statistical role of the Kalman Gain

CS491/691: Introduction to Aerial Robotics

Topic: Extended Kalman Filter

Dr. Kostas Alexis (CSE)

These slides relied on the lectures from C. Stachniss, J. Sturm and the book "Probabilistic Robotics" from Thurn et al.

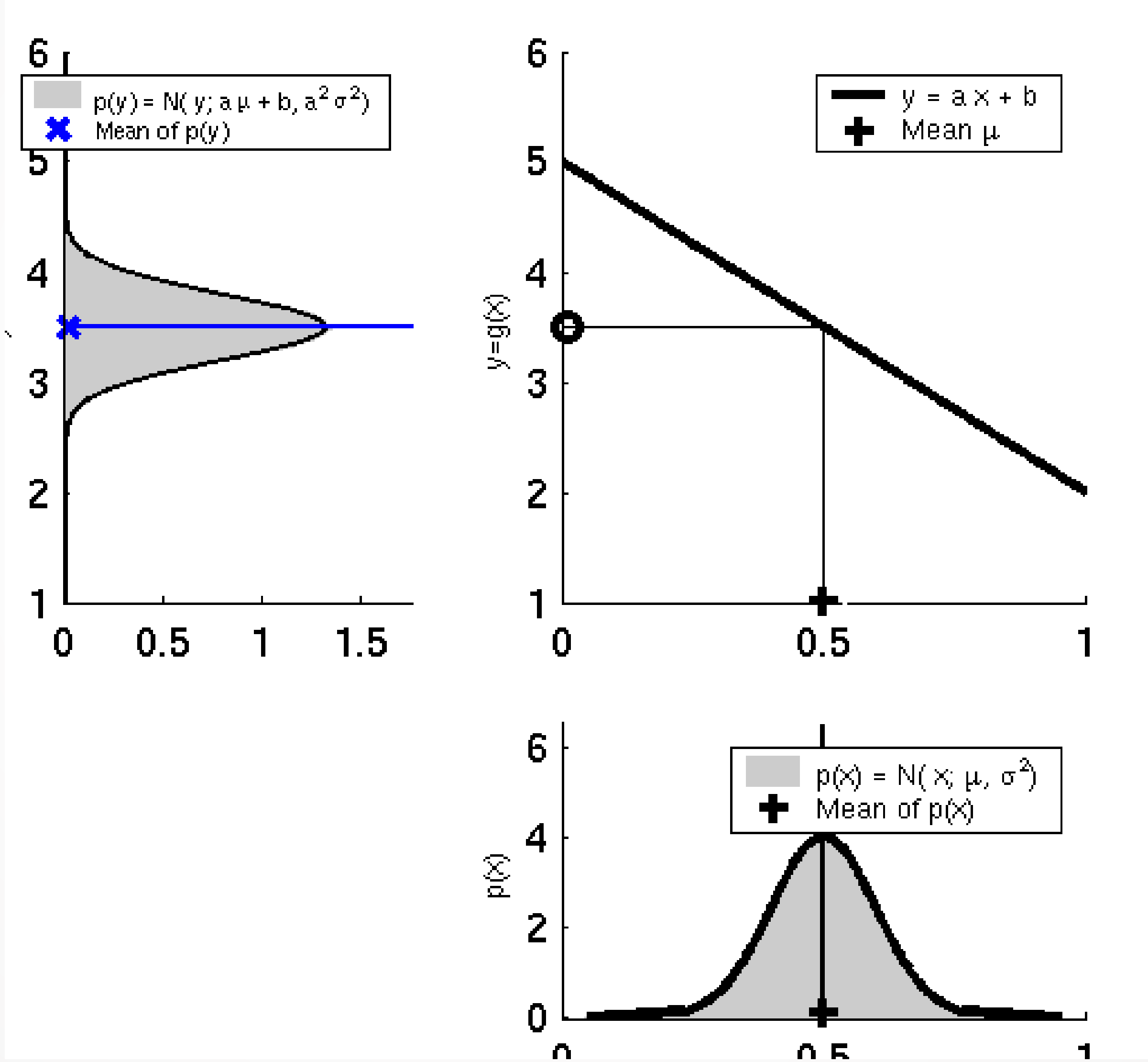
Kalman Filter Assumptions

- Gaussian distributions and noise
- Linear motion and observation model
  - **What if this is not the case?**

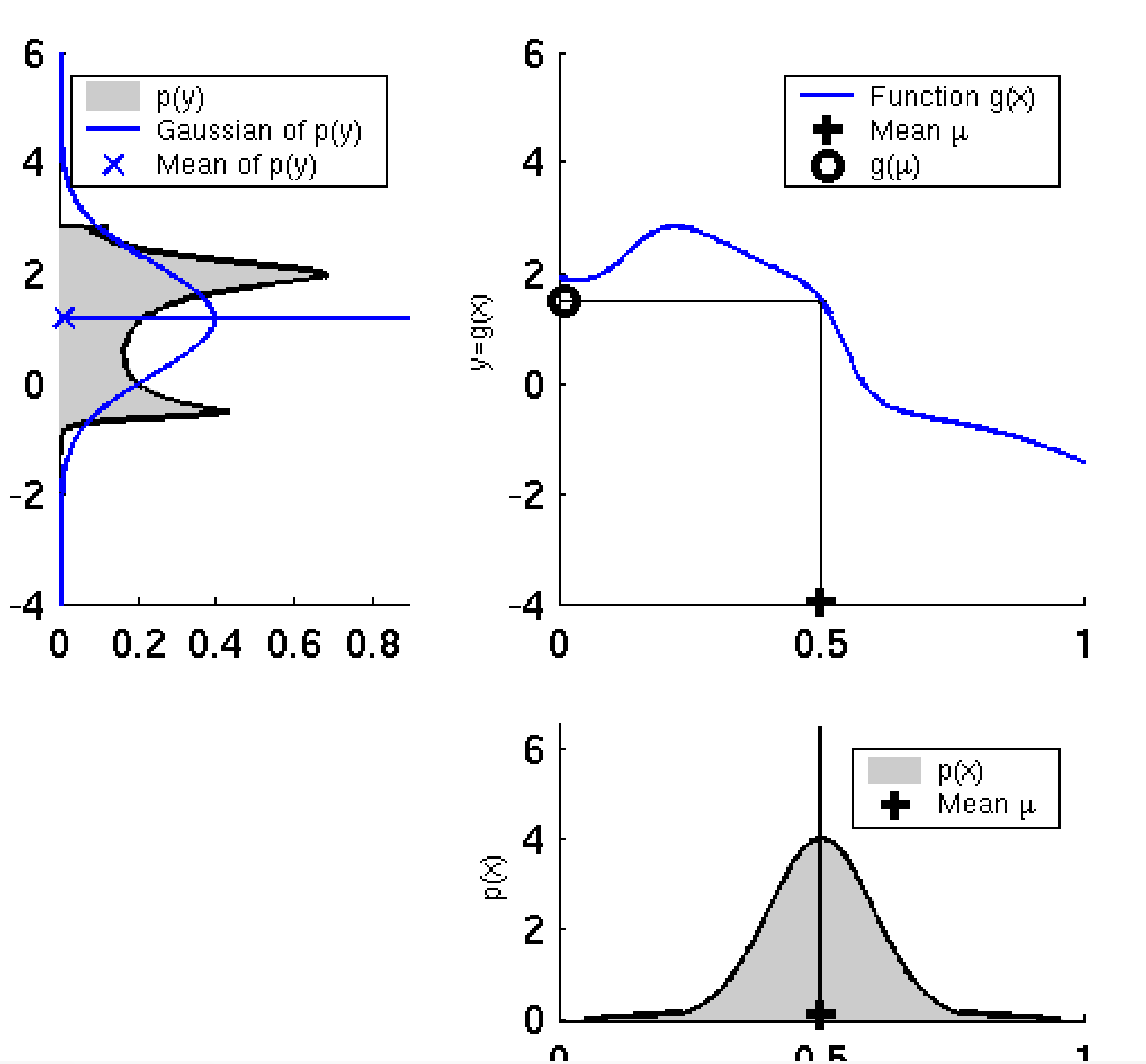
$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

$$z_t = C_t x_t + \delta_t$$

Linearity Assumption Revisited



Nonlinear Function



Nonlinear Dynamical Systems

- Real-life robots are mostly nonlinear systems.
- The **motion equations** are expressed as **nonlinear differential (or difference) equations**:

$$x_t = g(u_t, x_{t-1})$$

- Also leading to a **nonlinear observation function:**

$$z_t = h(x_t)$$

Taylor Expansion

- Solution: approximate via linearization of both functions
- **Motion Function:**

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$
$$= g(\mu_{t-1}, u_t) + G_t(x_{t-1} - \mu_{t-1})$$

- **Observation Function:**

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t}(x_t - \mu_t)$$
$$= h(\bar{\mu}_t) + H_t(x_t - \mu_t)$$

Reminder: Jacobian Matrix

- It is a non-square matrix *mxn* in general
- Given a vector-valued function:

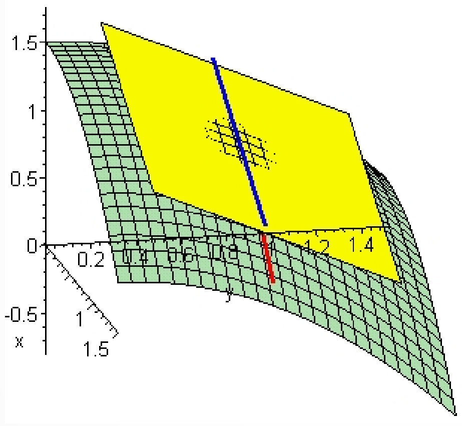
$$g(x) = \begin{pmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_m(x) \end{pmatrix}$$

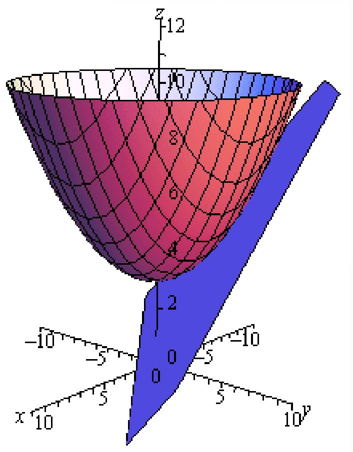
- The **Jacobian matrix** is defined as:

$$G_x = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{pmatrix}$$

Reminder: Jacobian Matrix

- It is the orientation of the tangent plane to the vector-valued function at a given point





Courtesy: K. Arras

- Generalizes the gradient of a scaled-valued function.

Extended Kalman Filter

- For each time step, do:
- **Apply Motion Model:**

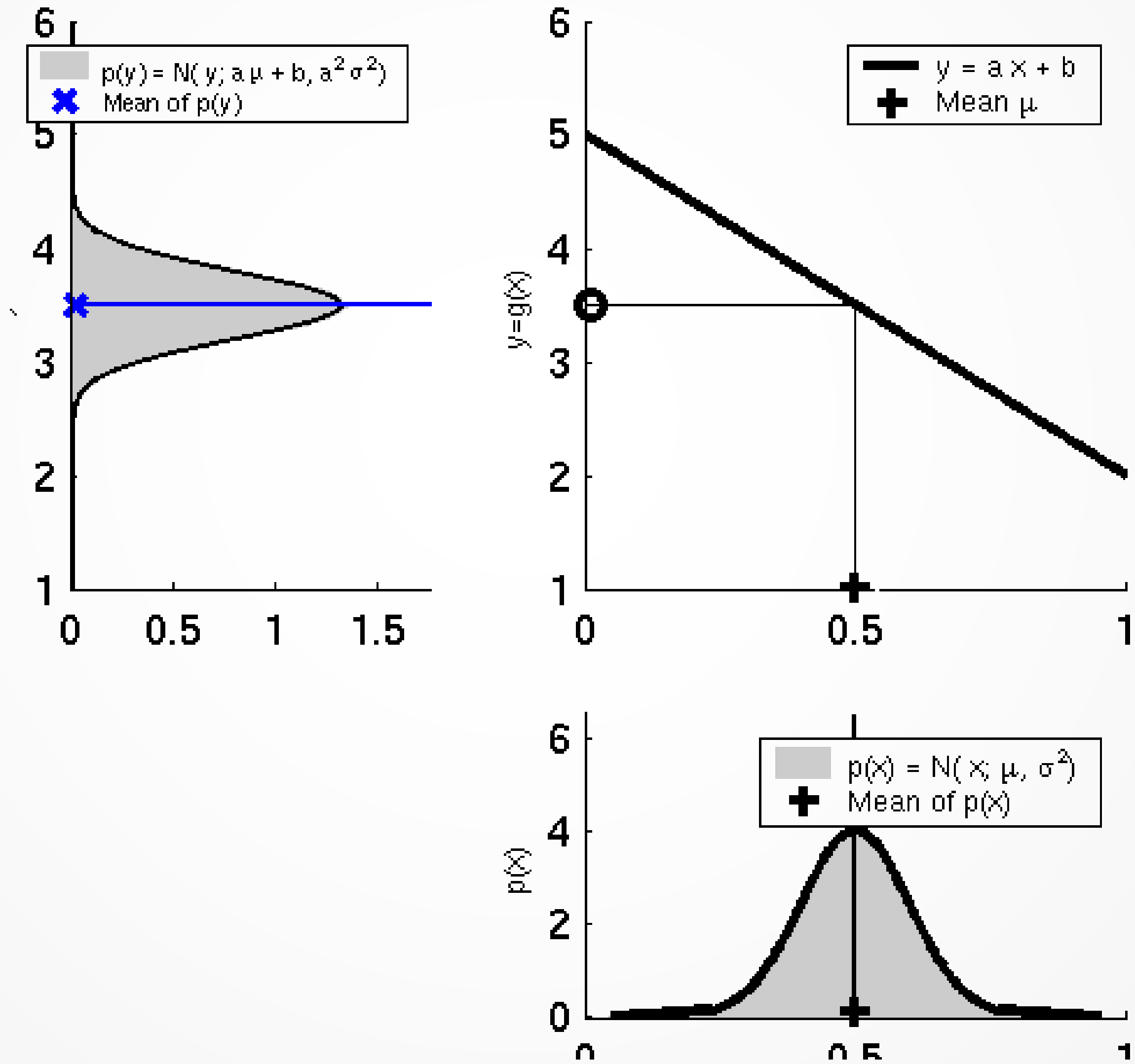
$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$
$$\bar{\Sigma}_t = G_t \Sigma G_t^\top + Q \quad \text{with} \quad G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}$$

- **Apply Sensor Model:**

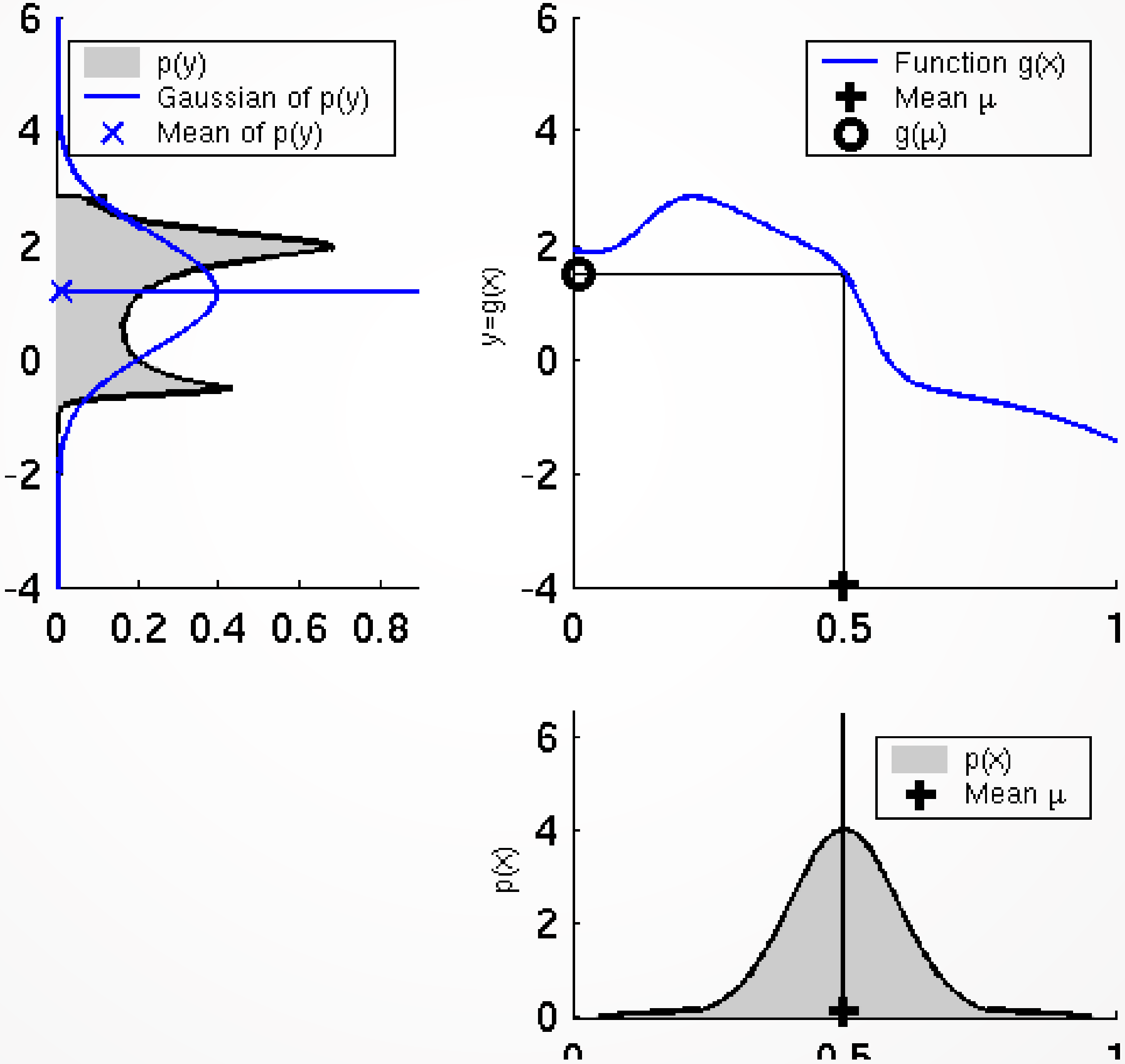
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

where $K_t = \bar{\Sigma}_t H_t^\top (H_t \bar{\Sigma}_t H_t^\top + R)^{-1}$ and $H_t = \dfrac{\partial h(\bar{\mu}_t)}{\partial x_t}$

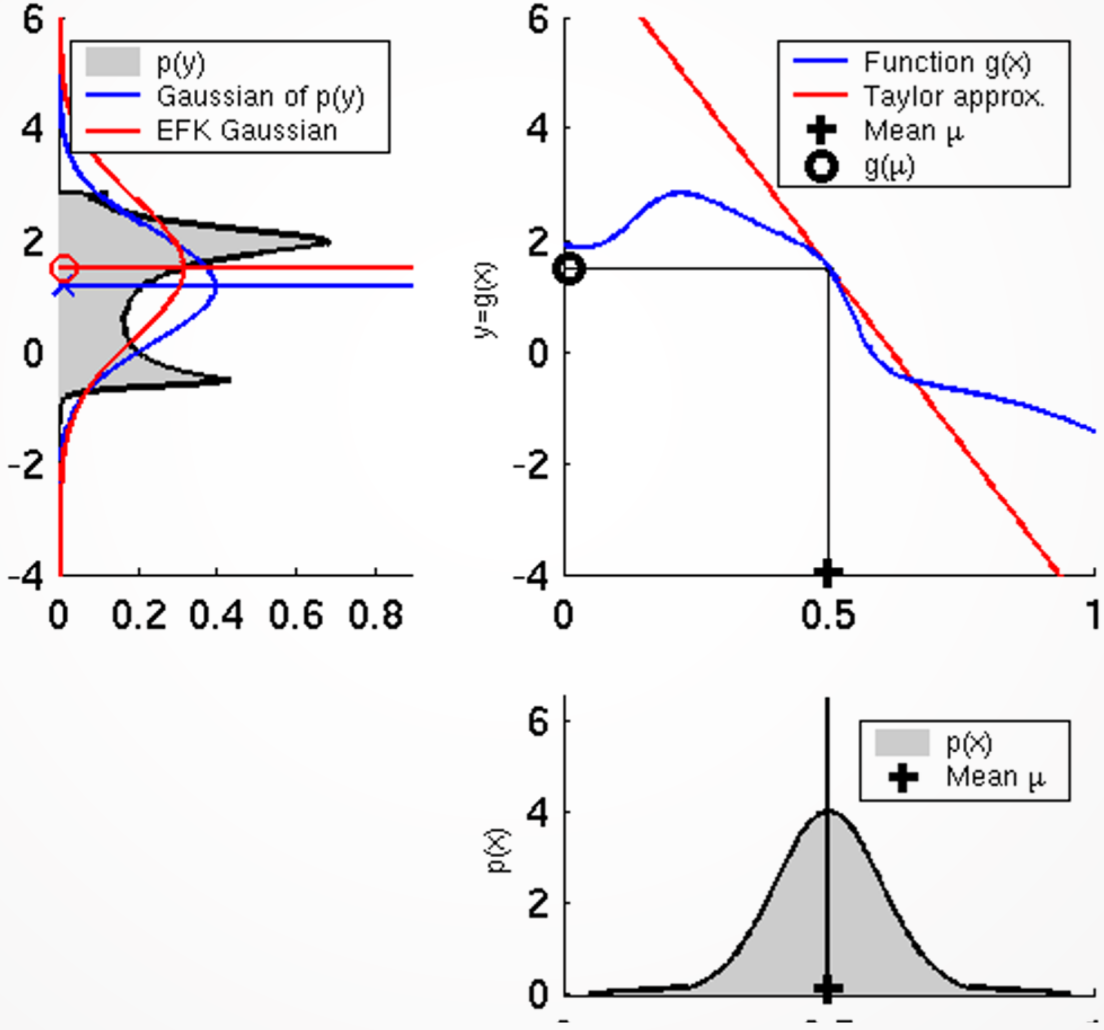
Linearity Assumption Revisited



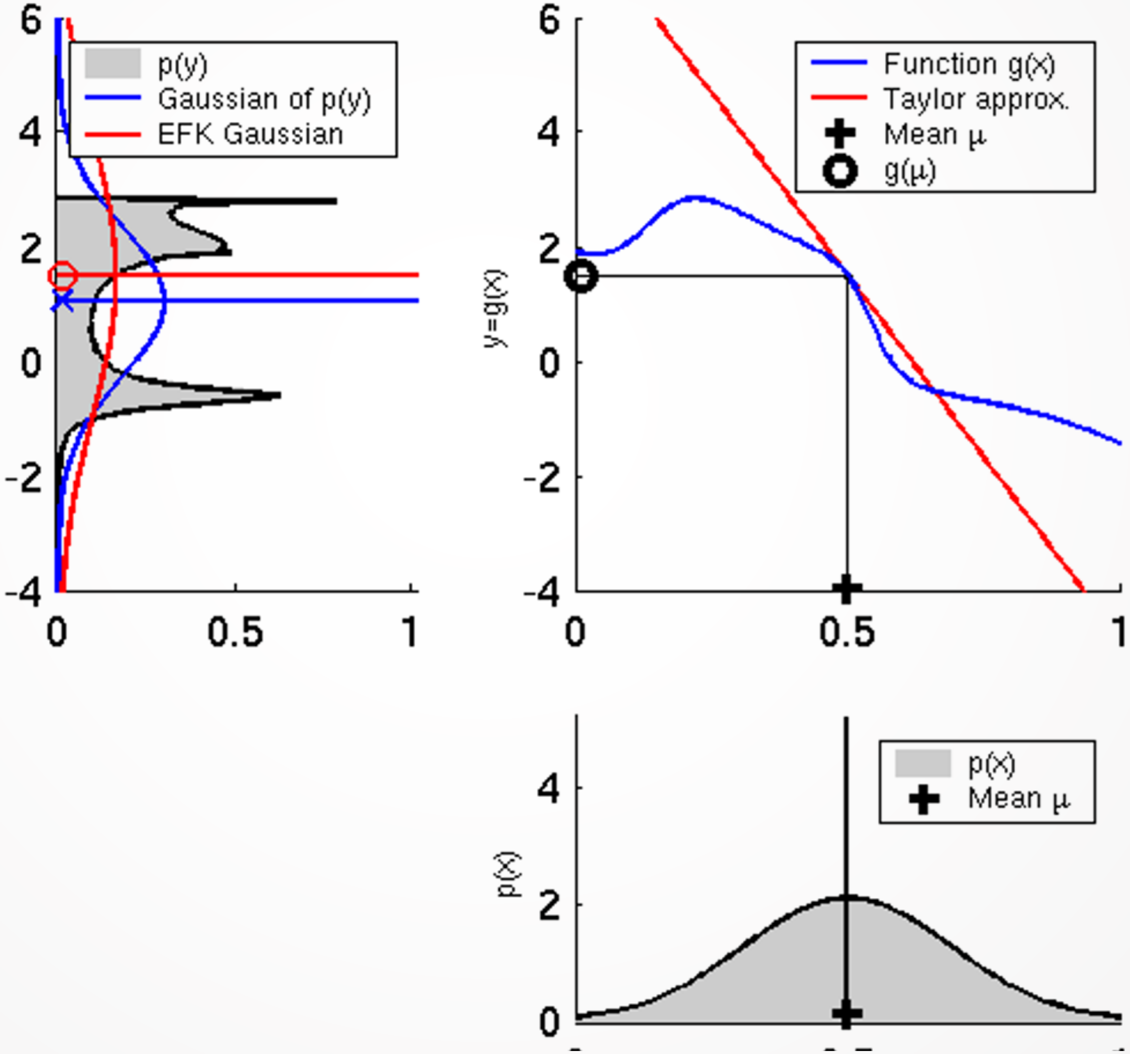
Nonlinear Function



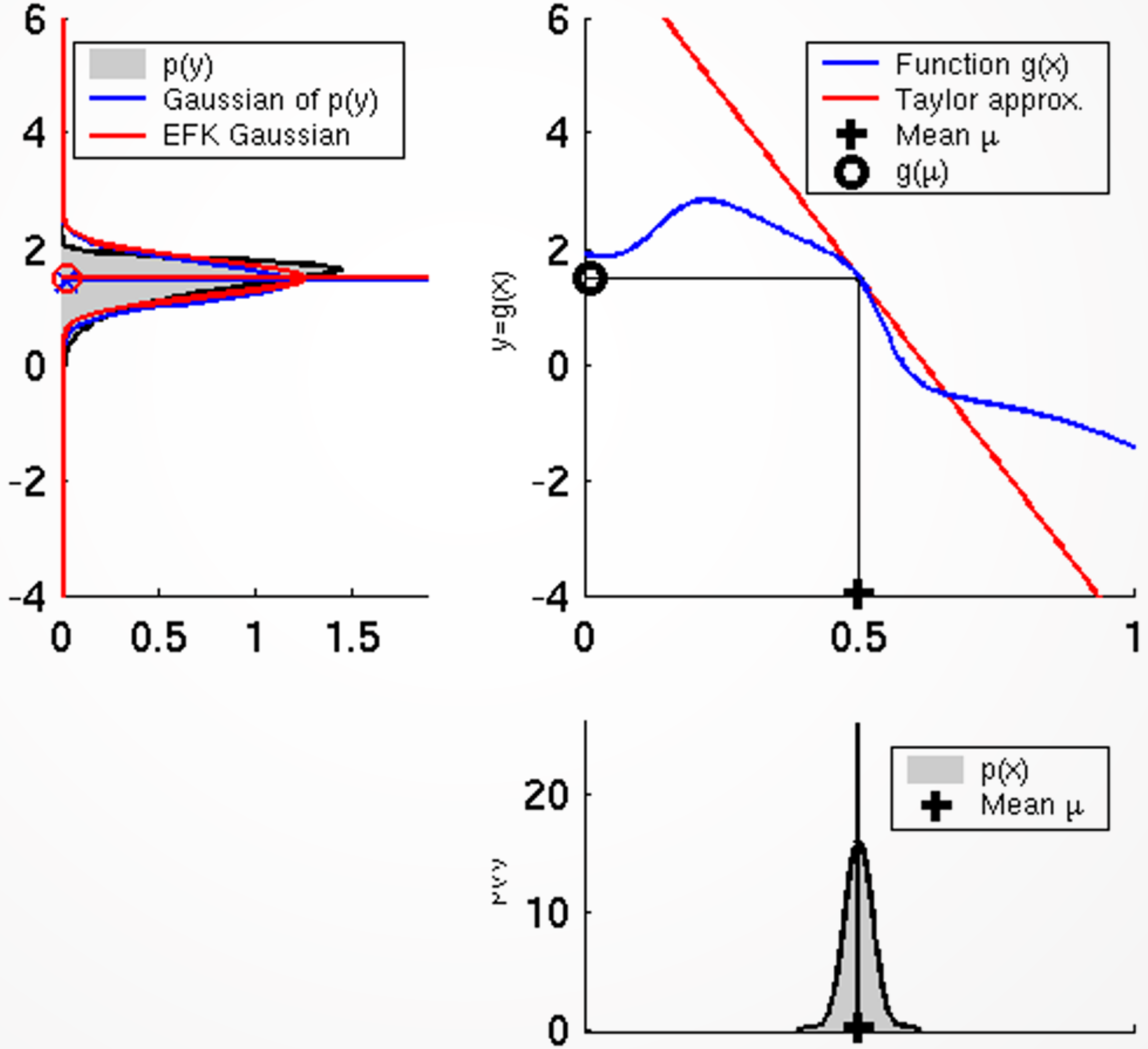
EKF Linearization (1)



EKF Linearization (2)



EKF Linearization (3)



Linearized Motion Model

- The linearized model leads to:

$$p(x_t \mid u_t, x_{t-1}) \approx \det\left(2\pi R_t\right)^{-\frac{1}{2}}$$
$$\exp\Big(-\frac{1}{2}\left(x_t - g(u_t, \mu_{t-1}) - G_t\left(x_{t-1} - \mu_{t-1}\right)\right)^T$$
$$R_t^{-1}\left(x_t - \underbrace{g(u_t, \mu_{t-1}) - G_t\left(x_{t-1} - \mu_{t-1}\right)}_{\text{linearized model}}\right)\Big)$$

- R_t describes the noise of the motion.

Linearized Observation Model

- The linearized model leads to:

$$p(z_t \mid x_t) = \det\left(2\pi Q_t\right)^{-\frac{1}{2}}$$
$$\exp\left(-\frac{1}{2}\left(z_t - h(\bar{\mu}_t) - H_t\left(x_t - \bar{\mu}_t\right)\right)^T\right.$$
$$\left. Q_t^{-1}\left(z_t - \underbrace{h(\bar{\mu}_t) - H_t\left(x_t - \bar{\mu}_t\right)}_{\text{linearized model}}\right)\right)$$

- Q_t describes the noise of the motion.

EKF Algorithm

1: $\textbf{Extended\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

2: $\bar{\mu}_t = \underline{g(u_t, \mu_{t-1})}$

3: $\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + R_t$

4: $K_t = \bar{\Sigma}_t \, H_t^T (H_t \, \bar{\Sigma}_t \, H_t^T + Q_t)^{-1}$

5: $\mu_t = \bar{\mu}_t + K_t(z_t - \underline{h(\bar{\mu}_t)})$

6: $\Sigma_t = (I - K_t \, H_t) \, \bar{\Sigma}_t$

7: return μ_t, Σ_t

$A_t \leftrightarrow G_t$

$C_t \leftrightarrow H_t$

KF vs EKF

EKF Summary

- Extension of the Kalman Filter.
- One way to deal with nonlinearities.
- Performs local linearizations.
- Works well in practice for moderate nonlinearities.
- Large uncertainty leads to increased approximation error.

EKF: Indicative Questions

- Describe the Extended Kalman Filter for a linear process of the form $\dot{x} = f(x, u)$
- Explain the statistical role of the Kalman Gain for nonlinear systems

CS491/691: Introduction to Aerial Robotics

Topic: Autonomous Exploration

Dr. Kostas Alexis (CSE)

The Exploration path planning problem

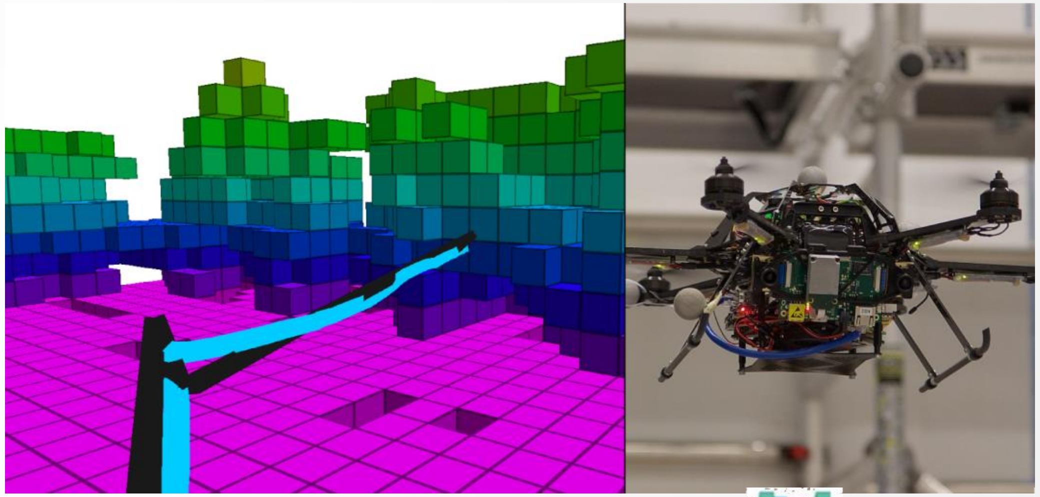
Problem Definition: Volumetric Exploration

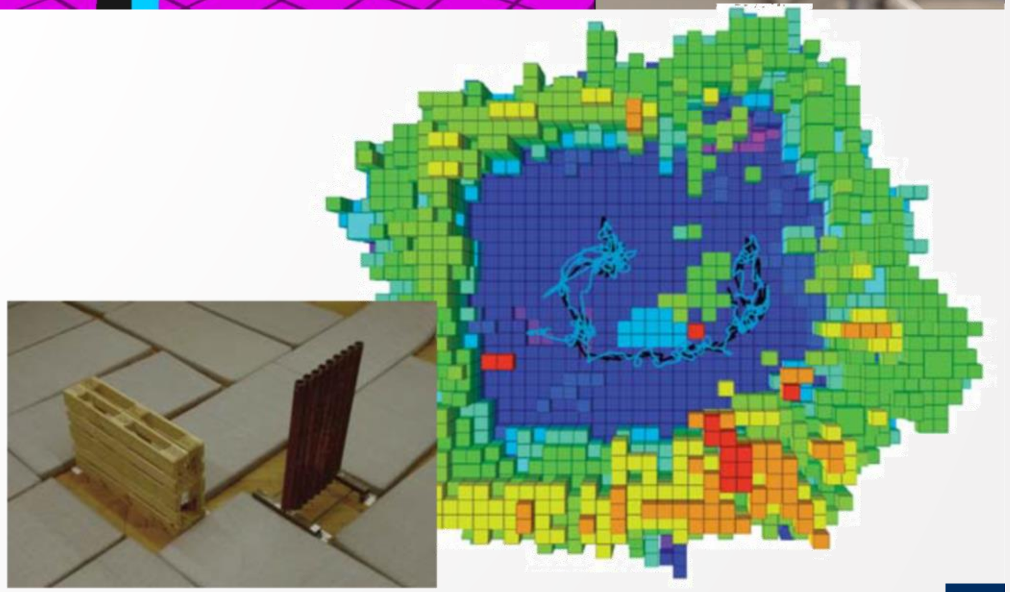
The exploration path planning problem consists in **exploring a previously unknown bounded 3D space** $V \subset \mathbb{R}^3$. This is to determine which parts of the initially unmapped space $V_{unm} = V$ are free $V_{free} \subset V$ or occupied $V_{occ} \subset V$. The operation is subject to vehicle kinematic and dynamic constraints, localization uncertainty and limitations of the employed sensor system with which the space is explored.

- As for most sensors the perception stops at surfaces, hollow spaces or narrow pockets can sometimes not be explored with a given setup. This residual space is denoted as V_{res}. The problem is considered to be fully solved when $V_{free} \cup V_{occ} = V \setminus V_{res}$.
- Due to the nature of the problem, a suitable path has to be computed online and in real-time, as free space to navigate is not known prior to its exploration.

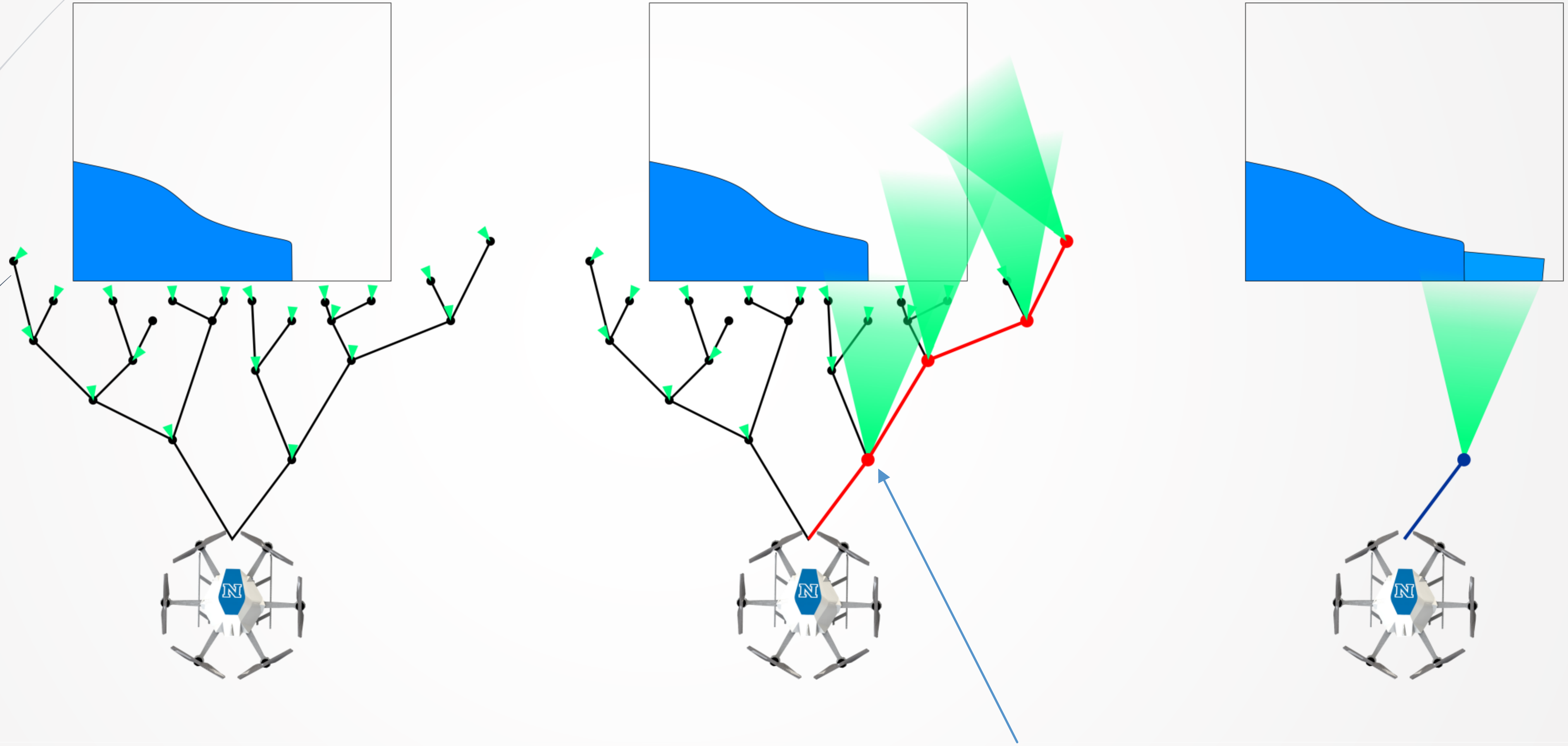
Receding Horizon Next-Best-View Exploration

- **Goal:** Fast and complete exploration of unknown environments.
- Define **sequences of viewpoints based on vertices sampled using random trees**.
- Select the path with the best sequence of best views.
- Execute only the first step of this best exploration path.
- Update the map after each iteration.
- Repeat the whole process in a receding horizon fashion.





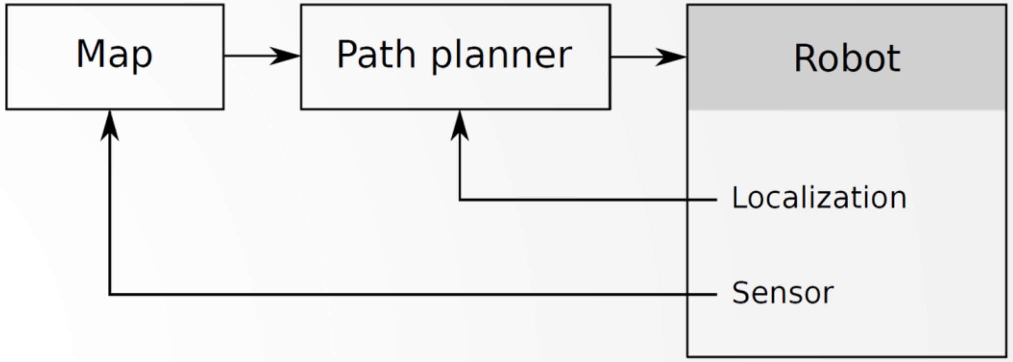
Exploration Planning (**`nbvplanner`**)



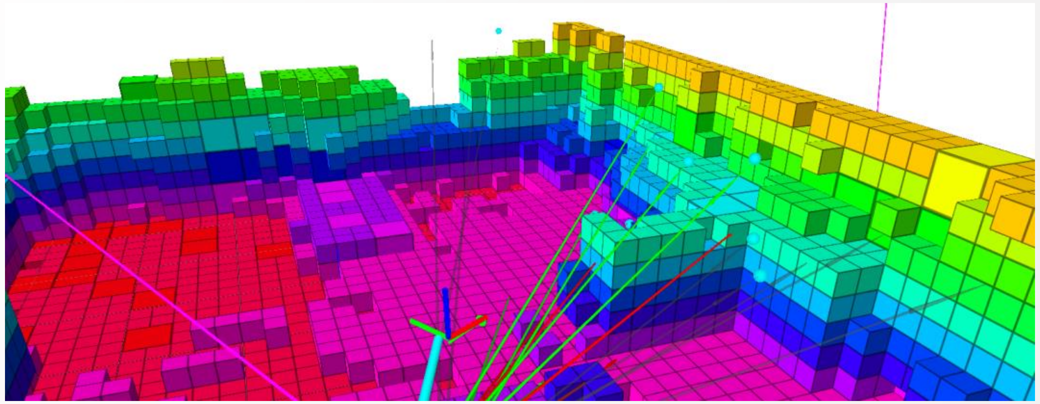
$$\mathbf{Gain}(n_k) = \mathbf{Gain}(n_{k-1}) + \mathbf{Visible}(\mathcal{M}, \xi_k)e^{-\lambda c(\sigma_{k-1}^k)}$$

Exploration Planning (`nbvplanner`)

- **Environment representation:** Occupancy Map dividing space V into $m \in M$ cubical volumes (voxels) that can be marked either as free, occupied or unmapped.
- Use of the `octomap` representation to enable computationally efficient access and search.
- Paths are planned only within the free space V_{free} and collision free point-to-point navigation is inherently supported.
- At each viewpoint/configuration of the environment ξ, the amount of space that is visible is computed as $Visible(M, \xi)$



The Receding Horizon Next-Best-View Exploration Planner relies on the real-time update of the 3D map of the environment.

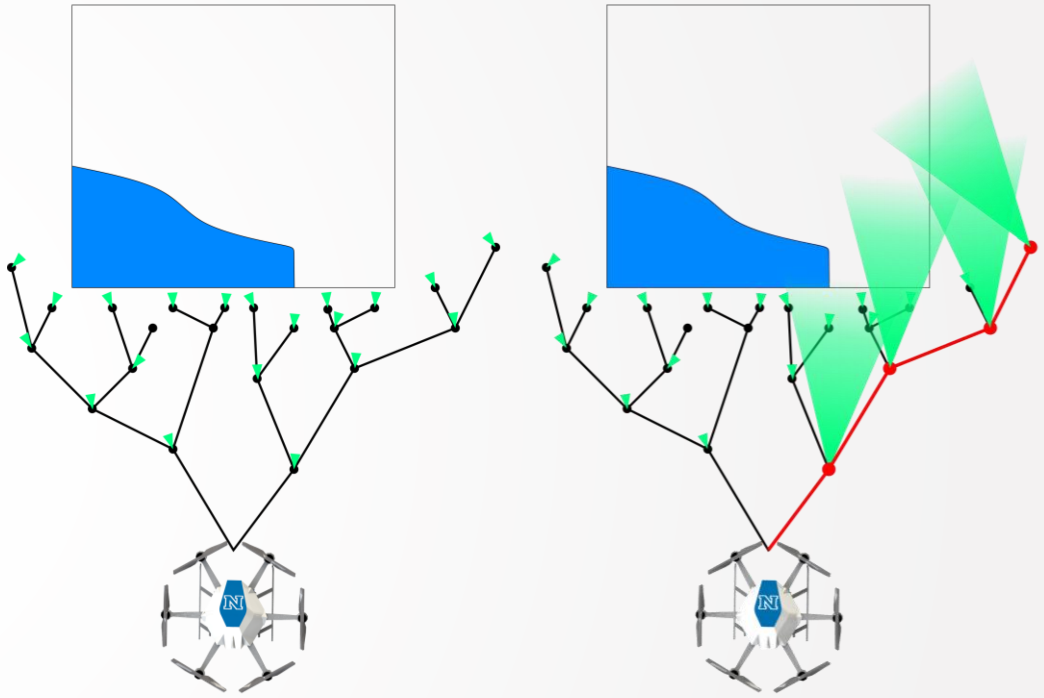


Exploration Planning (`nbvplanner`)

- **Tree-based exploration:** At every iteration, `nbvplanner` spans a random tree of finite depth. Each vertex of the tree is annotated regarding the collected Information Gain – a metric of how much new space is going to be explored.

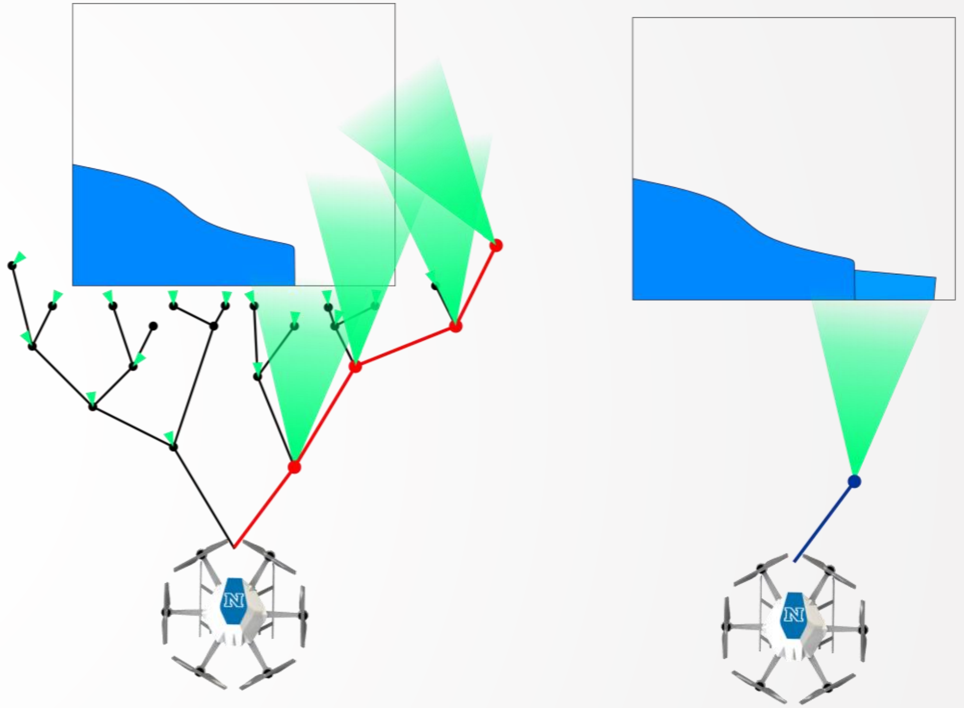
$$\mathbf{Gain}(n_k) = \mathbf{Gain}(n_{k-1}) + \mathbf{Visible}(\mathcal{M}, \xi_k) e^{-\lambda c(\sigma_{k-1}^k)}$$

- Within the sampled tree, evaluation regarding the path that overall leads to the highest information gain is conducted. This corresponds to the **best path** for the given iteration. It is a sequence of next-best-views as sampled based on the vertices of the spanned random tree.



Exploration Planning (`nbvplanner`)

- **Receding Horizon:** For the extracted best path of viewpoints, only the first viewpoint is actually executed.
- The system moves to the first viewpoint of the path of best viewpoints.
- The map is subsequently updated.
- Subsequently, the whole process is repeated within the next iteration. This gives rise to a receding horizon operation.



Exploration Planning (**nbvplanner**) Algorithm

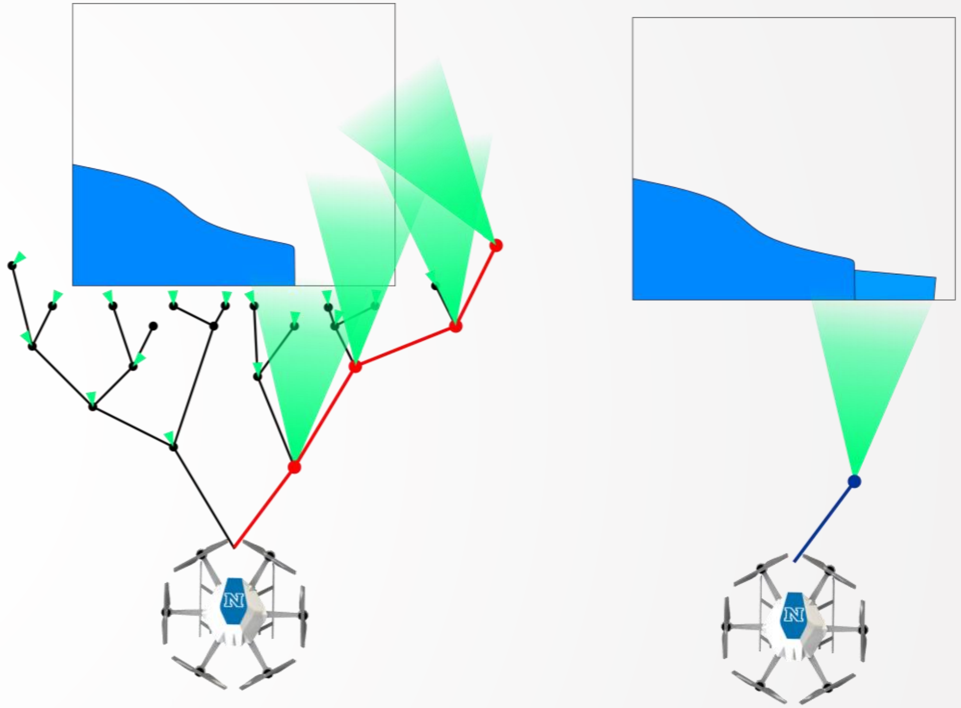
nbvplanner Iterative Step

- $\xi_0 \leftarrow$ current vehicle configuration
- Initialize $\boldsymbol{T}$ with ξ_0 and, unless first planner call, also previous best branch
- $g_{best} \leftarrow 0$ // Set best gain to zero
- $n_{best} \leftarrow n_0(\xi_0)$ // Set best node to root
- $N_T \leftarrow$ Number of nodes in $\boldsymbol{T}$
- **while** $N_T < N_{max}$ or $g_{best} == 0$ **do**
  - Incrementally build $\boldsymbol{T}$ by adding $n_{new}(\xi_{new})$
  - $N_T \leftarrow N_T + 1$
  - **if** $Gain(n_{new}) > g_{best}$ **then**
    - $n_{best} \leftarrow n_{new}$
    - $g_{best} \leftarrow Gain(n_{new})$
  - **if** $N_T > N_{TOT}$ **then**
    - Terminate exploration
- $\sigma \leftarrow \boldsymbol{ExtractBestPathSegment}(n_{best})$
- Delete $\boldsymbol{T}$
- **return** σ

Exploration Planning (`nbvplanner`) Remarks

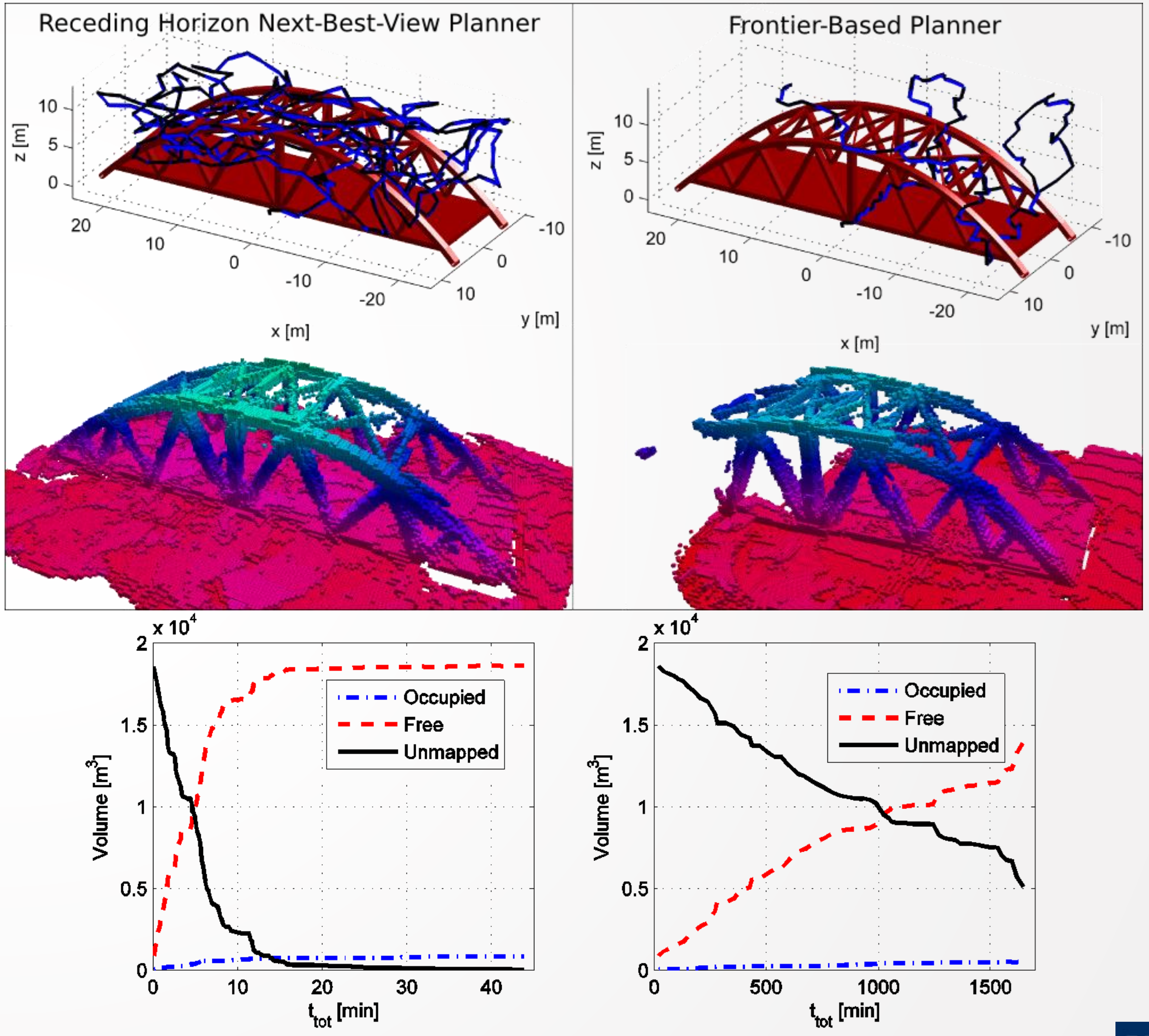
- **Inherently Collision-free:** As all paths of `nbvplanner` are selected along branches within RRT-based spanned trees, all paths are inherently collision-free.
- **Computational Cost:** nbvplanner has a thin structure and most of the computational cost is related with collision-checking functionalities. The formula that expresses the complexity of the algorithm takes the form:

$$\mathcal{O}(N_{\mathbb{T}} \log(N_{\mathbb{T}}) + N_{\mathbb{T}}/r^3 \log(V/r^3) + N_{\mathbb{T}}(d_{\max}^{\text{planner}}/r)^4 \log(V/r^3))$$



`nbvplanner` Evaluation (Simulation)

- **Simulation-based evaluation:** Explore a bridge.
- Comparison with Frontier-based exploration.



Extension to Surface Inspection

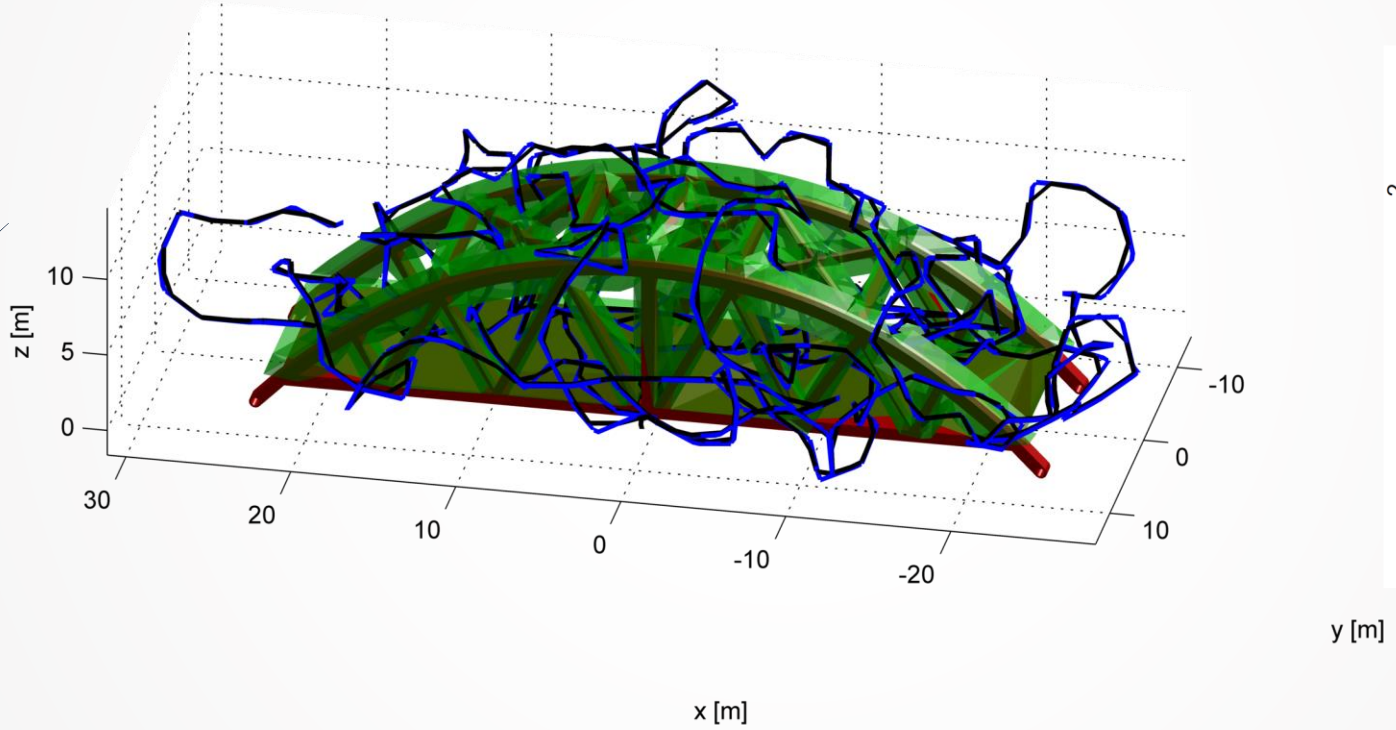
Problem Definition: Surface Inspection

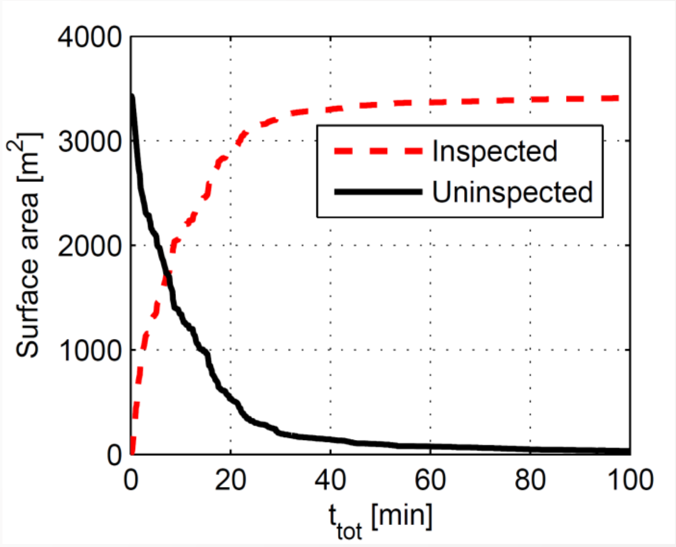
Given a surface S, find a collision free path σ starting at an initial configuration $\xi_{init} \in \Xi$ that leads to the inspection of the part S_{insp}, when being executed, such that there does not exist any collision free configuration from which any piece of $S \backslash S_{insp}$ could be inspected. Thus, $S_{insp} = S \backslash S_{res}$.

- Let $\overline{V_s} \subseteq \Xi$ be the set of all configurations from which the surface piece $s \subseteq S$ can be inspected. Then the residual surface is given as $S_{res} = \cup_{s \in S} (s | \overline{V_s} = 0)$

`nbvplanner` Evaluation (Simulation)

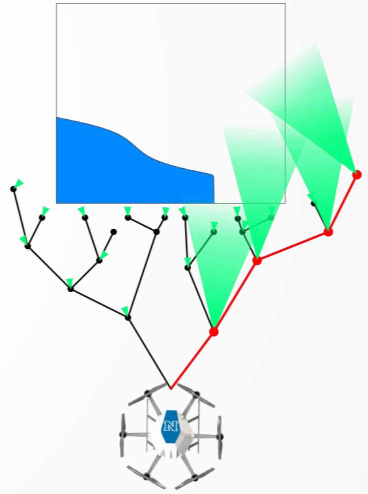
- **Extension to surface inspection:** The robot identifies trajectories that locally ensure maximum information gain regarding surface coverage.

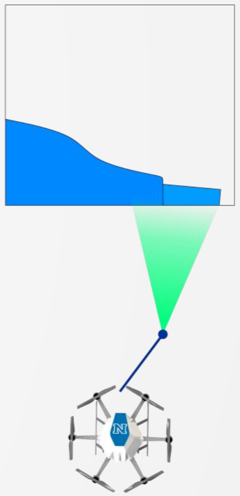




EPP: Indicative Questions

- Describe a path planning algorithm for volumetric exploration
- What is the role of receding horizon path planning for the problem of unknown area exploration





Thank you!

Please ask your question!