Distributed Gaussian Particle Filtering Using Likelihood Consensus

Ondrej Hlinka[a], *Ondrej Slučiak*[a], *Franz Hlawatsch*[a], *Petar M. Djurić*[b], *and Markus Rupp*[a]

[a]Institute of Telecommunications, Vienna University of Technology, Austria

[b]Department of Electrical and Computer Engineering, Stony Brook University, NY, USA

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

STONY BROOK UNIVERSITY

Contribution

- We propose a distributed implementation of the Gaussian particle filter introduced in [Kotecha & Djurić 2003]

Contribution

- We propose a distributed implementation of the Gaussian particle filter introduced in [Kotecha & Djurić 2003]
- Each sensor computes a global estimate based on the joint (all-sensors) likelihood function

Contribution

- We propose a distributed implementation of the Gaussian particle filter introduced in [Kotecha & Djurić 2003]
- Each sensor computes a global estimate based on the joint (all-sensors) likelihood function
- The (approximate) joint likelihood function is obtained in a distributed way using the likelihood consensus scheme [Hlinka et al. 2010]

Contribution

- We propose a distributed implementation of the Gaussian particle filter introduced in [Kotecha & Djurić 2003]
- Each sensor computes a global estimate based on the joint (all-sensors) likelihood function
- The (approximate) joint likelihood function is obtained in a distributed way using the likelihood consensus scheme [Hlinka et al. 2010]
- A second stage of consensus algorithms can be used to significantly reduce the complexity

Contribution

- We propose a distributed implementation of the Gaussian particle filter introduced in [Kotecha & Djurić 2003]
- Each sensor computes a global estimate based on the joint (all-sensors) likelihood function
- The (approximate) joint likelihood function is obtained in a distributed way using the likelihood consensus scheme [Hlinka et al. 2010]
- A second stage of consensus algorithms can be used to significantly reduce the complexity
- Comparison to other consensus-based distributed particle filters:
  - in [Farahmand et al. 2010], no approximations are needed but the communication requirements can be much higher
  - in [Gu et al. 2008], only local likelihood functions are used, resulting in a performance degradation

Outline

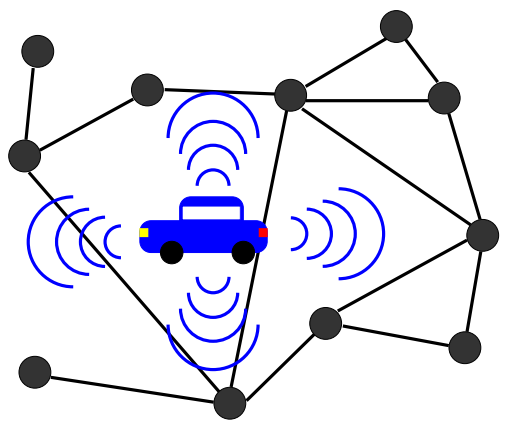
- Distributed estimation in wireless sensor networks
- Distributed Gaussian particle filtering
- Likelihood consensus
- Distributed Gaussian particle filtering with reduced complexity
- Target tracking example and simulation results

Distributed estimation

- We consider a wireless sensor network composed of K sensor nodes jointly estimating a time-varying M-dimensional state vector $\mathbf{x}_n$, $n = 1, 2, \ldots$
- Each sensor $k \in \{1, \ldots, K\}$ obtains a measurement $\mathbf{z}_{n,k}$

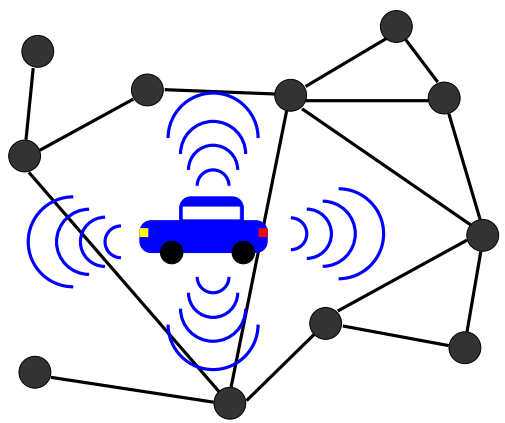
Distributed estimation

- We consider a wireless sensor network composed of K sensor nodes jointly estimating a time-varying M-dimensional state vector $\mathbf{x}_n$, $n = 1, 2, \ldots$
- Each sensor $k \in \{1, \ldots, K\}$ obtains a measurement $\mathbf{z}_{n,k}$
- Example application: localization/tracking based on the sound emitted by a moving target (vehicle)



Distributed estimation

- We consider a wireless sensor network composed of K sensor nodes jointly estimating a time-varying M-dimensional state vector $\mathbf{x}_n$, $n = 1, 2, \ldots$
- Each sensor $k \in \{1, \ldots, K\}$ obtains a measurement $\mathbf{z}_{n,k}$
- Example application: localization/tracking based on the sound emitted by a moving target (vehicle)



- Goals:
  - each sensor node obtains a global state estimate $\hat{\mathbf{x}}_{n,k}$ based on the measurements of all sensors (important for sensor-actuator/robotic networks)
  - only local processing and short-distance communications are used
  - no fusion center, no routing of measurements through the network

Sequential Bayesian estimation

- We wish to perform sequential estimation of the time-varying state $\mathbf{x}_n$ from the current and past measurements of all sensors in the sensor network

Sequential Bayesian estimation

- We wish to perform sequential estimation of the time-varying state $\mathbf{x}_n$ from the current and past measurements of all sensors in the sensor network
- A nonlinear, non-Gaussian state-space model with independent additive Gaussian measurement noises is considered
- The system is described by:
  - state-transition probability density function (pdf) $f(\mathbf{x}_n|\mathbf{x}_{n-1})$
  - joint likelihood function (JLF) $f(\mathbf{z}_n|\mathbf{x}_n)$, where $\mathbf{z}_n = (\mathbf{z}_{n,1}^T \cdots \mathbf{z}_{n,K}^T)^T$

Sequential Bayesian estimation

- We wish to perform sequential estimation of the time-varying state $\mathbf{x}_n$ from the current and past measurements of all sensors in the sensor network
- A nonlinear, non-Gaussian state-space model with independent additive Gaussian measurement noises is considered
- The system is described by:
  - state-transition probability density function (pdf) $f(\mathbf{x}_n|\mathbf{x}_{n-1})$
  - joint likelihood function (JLF) $f(\mathbf{z}_n|\mathbf{x}_n)$, where $\mathbf{z}_n = (\mathbf{z}_{n,1}^T \cdots \mathbf{z}_{n,K}^T)^T$
- Optimal Bayesian estimation amounts to calculation of the posterior pdf $f(\mathbf{x}_n|\mathbf{z}_{1:n})$, where $\mathbf{z}_{1:n} = (\mathbf{z}_1^T \cdots \mathbf{z}_n^T)^T$
- Sequential estimation is enabled by the recursive posterior update

$$f(\mathbf{x}_n|\mathbf{z}_{1:n}) \propto \underbrace{f(\mathbf{z}_n|\mathbf{x}_n)}_{\text{JLF}} \int_{\mathbf{x}_{n-1}} f(\mathbf{x}_n|\mathbf{x}_{n-1}) f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})\, d\mathbf{x}_{n-1}$$

Outline

- Distributed estimation in wireless sensor networks
- Distributed Gaussian particle filtering
- Likelihood consensus
- Distributed Gaussian particle filtering with reduced complexity
- Target tracking example and simulation results

Gaussian particle filtering

- For nonlinear/non-Gaussian systems, optimal Bayesian estimation is typically computationally infeasible
- A computationally feasible approximation is provided by the particle filtering (sequential Monte Carlo) approach

Gaussian particle filtering

- For nonlinear/non-Gaussian systems, optimal Bayesian estimation is typically computationally infeasible
- A computationally feasible approximation is provided by the particle filtering (sequential Monte Carlo) approach
- Gaussian particle filtering [Kotecha & Djurić 2003]:
  - the posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$ is approximated by a Gaussian pdf $\mathcal{N}(\boldsymbol{\mu}_n, \mathbf{C}_n)$
  - the mean $\boldsymbol{\mu}_n$ and covariance $\mathbf{C}_n$ of the Gaussian approximation are obtained from a set of J weighted samples/particles $\{\mathbf{x}_n^{(j)}, w_n^{(j)}\}_{j=1}^{J}$

Distributed Gaussian particle filtering – 1

- We propose a distributed Gaussian particle filter (DGPF)
- Each sensor uses a local Gaussian particle filter to sequentially track the mean $\boldsymbol{\mu}_{n,k}$ and covariance $\mathbf{C}_{n,k}$ of a local Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n,k}, \mathbf{C}_{n,k})$ to the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$

Distributed Gaussian particle filtering – 1

- We propose a distributed Gaussian particle filter (DGPF)
- Each sensor uses a local Gaussian particle filter to sequentially track the mean $\boldsymbol{\mu}_{n,k}$ and covariance $\mathbf{C}_{n,k}$ of a local Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n,k}, \mathbf{C}_{n,k})$ to the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$
- The measurement update at each sensor uses the (global) JLF. This ensures that global estimates are obtained at each sensor

Distributed Gaussian particle filtering – 1

- We propose a distributed Gaussian particle filter (DGPF)
- Each sensor uses a local Gaussian particle filter to sequentially track the mean $\boldsymbol{\mu}_{n,k}$ and covariance $\mathbf{C}_{n,k}$ of a local Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n,k}, \mathbf{C}_{n,k})$ to the global posterior $f(\mathbf{x}_n|\mathbf{z}_{1:n})$
- The measurement update at each sensor uses the (global) JLF. This ensures that global estimates are obtained at each sensor
- The JLF is provided to each sensor in a distributed way using likelihood consensus as proposed in [Hlinka et al. 2010]:
  - the consensus algorithms employed by the likelihood consensus scheme require only local communications and operate without complex routing protocols
  - no measurements or particles need to be exchanged between the sensors

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

  1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

  1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$
  2. At time n, sensor k randomly draws J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$ from the locally available pdf $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

  1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$
  2. At time n, sensor k randomly draws J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$ from the locally available pdf $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$
  3. For each $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, sensor k draws a new, "predicted" particle $\mathbf{x}_{n,k}^{(j)}$ from the state-transition pdf $f(\mathbf{x}_n|\bar{\mathbf{x}}_{n-1,k}^{(j)})$

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

  1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$
  2. At time n, sensor k randomly draws J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$ from the locally available pdf $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$
  3. For each $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, sensor k draws a new, "predicted" particle $\mathbf{x}_{n,k}^{(j)}$ from the state-transition pdf $f(\mathbf{x}_n|\bar{\mathbf{x}}_{n-1,k}^{(j)})$
  4. Likelihood consensus is used to calculate the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ at each sensor – this requires communications with neighboring sensors

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

  1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$
  2. At time n, sensor k randomly draws J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$ from the locally available pdf $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$
  3. For each $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, sensor k draws a new, "predicted" particle $\mathbf{x}_{n,k}^{(j)}$ from the state-transition pdf $f(\mathbf{x}_n|\bar{\mathbf{x}}_{n-1,k}^{(j)})$
  4. Likelihood consensus is used to calculate the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ at each sensor – this requires communications with neighboring sensors
  5. Next, sensor k updates its weights using the JLF: $w_{n,k}^{(j)} \propto f(\mathbf{z}_n|\mathbf{x}_{n,k}^{(j)})$

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:

1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$
2. At time n, sensor k randomly draws J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$ from the locally available pdf $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$
3. For each $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, sensor k draws a new, "predicted" particle $\mathbf{x}_{n,k}^{(j)}$ from the state-transition pdf $f(\mathbf{x}_n|\bar{\mathbf{x}}_{n-1,k}^{(j)})$
4. Likelihood consensus is used to calculate the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ at each sensor – this requires communications with neighboring sensors
5. Next, sensor k updates its weights using the JLF: $w_{n,k}^{(j)} \propto f(\mathbf{z}_n|\mathbf{x}_{n,k}^{(j)})$
6. Finally, sensor k calculates from the particles $\{\mathbf{x}_{n,k}^{(j)}, w_{n,k}^{(j)}\}_{j=1}^{J}$ the new $\boldsymbol{\mu}_{n,k}$ and $\mathbf{C}_{n,k}$

Distributed Gaussian particle filtering – 2

- Each sensor performs local Gaussian particle filtering:
  1. At time $n-1$, sensor k obtained the Gaussian approximation $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$ to the previous global posterior $f(\mathbf{x}_{n-1}|\mathbf{z}_{1:n-1})$
  2. At time n, sensor k randomly draws J particles $\bar{\mathbf{x}}_{n-1,k}^{(j)}$ from the locally available pdf $\mathcal{N}(\boldsymbol{\mu}_{n-1,k}, \mathbf{C}_{n-1,k})$
  3. For each $\bar{\mathbf{x}}_{n-1,k}^{(j)}$, sensor k draws a new, "predicted" particle $\mathbf{x}_{n,k}^{(j)}$ from the state-transition pdf $f(\mathbf{x}_n|\bar{\mathbf{x}}_{n-1,k}^{(j)})$
  4. Likelihood consensus is used to calculate the JLF $f(\mathbf{z}_n|\mathbf{x}_n)$ at each sensor – this requires communications with neighboring sensors
  5. Next, sensor k updates its weights using the JLF: $w_{n,k}^{(j)} \propto f(\mathbf{z}_n|\mathbf{x}_{n,k}^{(j)})$
  6. Finally, sensor k calculates from the particles $\{\mathbf{x}_{n,k}^{(j)}, w_{n,k}^{(j)}\}_{j=1}^{J}$ the new $\boldsymbol{\mu}_{n,k}$ and $\mathbf{C}_{n,k}$
- The global state estimate $\hat{\mathbf{x}}_{n,k}$ is given by $\boldsymbol{\mu}_{n,k}$

Outline

- Distributed estimation in wireless sensor networks
- Distributed Gaussian particle filtering
- Likelihood consensus
- Distributed Gaussian particle filtering with reduced complexity
- Target tracking example and simulation results

Joint likelihood function

- Sensor k acquires a measurement vector $\mathbf{z}_{n,k}$ according to

$$\mathbf{z}_{n,k} = \mathbf{h}_{n,k}(\mathbf{x}_n) + \mathbf{v}_{n,k} \, ,$$

  where all $\mathbf{v}_{n,k}$ are independent and $\mathbf{v}_{n,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,k})$

Joint likelihood function

- Sensor k acquires a measurement vector $\mathbf{z}_{n,k}$ according to

$$\mathbf{z}_{n,k} \;=\; \mathbf{h}_{n,k}(\mathbf{x}_n) + \mathbf{v}_{n,k}\;,$$

where all $\mathbf{v}_{n,k}$ are independent and $\mathbf{v}_{n,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,k})$

- The JLF is obtained as

$$f(\mathbf{z}_n|\mathbf{x}_n) \;=\; \prod_{k=1}^{K} f(\mathbf{z}_{n,k}|\mathbf{x}_n) \;\propto\; \exp\!\left(-\frac{1}{2} S_n(\mathbf{z}_n, \mathbf{x}_n)\right),$$

with

$$S_n(\mathbf{z}_n, \mathbf{x}_n) \;\triangleq\; \sum_{k=1}^{K} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]^T \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]$$

Joint likelihood function

- Sensor k acquires a measurement vector $\mathbf{z}_{n,k}$ according to

$$\mathbf{z}_{n,k} \;=\; \mathbf{h}_{n,k}(\mathbf{x}_n) + \mathbf{v}_{n,k}\,,$$

where all $\mathbf{v}_{n,k}$ are independent and $\mathbf{v}_{n,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{n,k})$

- The JLF is obtained as

$$f(\mathbf{z}_n|\mathbf{x}_n) \;=\; \prod_{k=1}^{K} f(\mathbf{z}_{n,k}|\mathbf{x}_n) \;\propto\; \exp\!\left(-\frac{1}{2} S_n(\mathbf{z}_n, \mathbf{x}_n)\right),$$

with

$$S_n(\mathbf{z}_n, \mathbf{x}_n) \;\triangleq\; \sum_{k=1}^{K} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]^T \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \mathbf{h}_{n,k}(\mathbf{x}_n)]$$

- Direct calculation of $S_n(\mathbf{z}_n, \mathbf{x}_n)$ requires that each sensor knows the measurements $\mathbf{z}_{n,k}$ and measurement functions $\mathbf{h}_{n,k}(\cdot)$ of all other sensors

- However, an approximation of $S_n(\mathbf{z}_n, \mathbf{x}_n)$ can be calculated at each sensor by means of consensus algorithms

Approximation of the measurement function

- We approximate the measurement function of sensor k, $\mathbf{h}_{n,k}(\mathbf{x}_n)$, by a multivariate vector-valued polynomial of degree R:

$$\mathbf{h}_{n,k}(\mathbf{x}_n) \approx \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n) \triangleq \sum_{\mathbf{r}=\mathbf{0}}^{R} \boldsymbol{\alpha}_{\mathbf{r},n,k}\, p_{\mathbf{r}}(\mathbf{x}_n)\,,$$

where $\mathbf{r} \triangleq (r_1, \ldots, r_M)$, $p_{\mathbf{r}}(\mathbf{x}_n) \triangleq x_{n,1}^{r_1} x_{n,2}^{r_2} \ldots x_{n,M}^{r_M}$, and $\sum_{\mathbf{r}=\mathbf{0}}^{R}$ is short for $\sum_{r_1=0}^{R} \cdots \sum_{r_M=0}^{R}$ with $\sum_{m=1}^{M} r_m \leq R$

Approximation of the measurement function

- We approximate the measurement function of sensor k, $\mathbf{h}_{n,k}(\mathbf{x}_n)$, by a multivariate vector-valued polynomial of degree R:

$$\mathbf{h}_{n,k}(\mathbf{x}_n) \approx \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n) \triangleq \sum_{\mathbf{r}=\mathbf{0}}^{R} \boldsymbol{\alpha}_{\mathbf{r},n,k}\, p_{\mathbf{r}}(\mathbf{x}_n)\,,$$

  where $\mathbf{r} \triangleq (r_1, \ldots, r_M)$, $p_{\mathbf{r}}(\mathbf{x}_n) \triangleq x_{n,1}^{r_1} x_{n,2}^{r_2} \ldots x_{n,M}^{r_M}$, and $\sum_{\mathbf{r}=\mathbf{0}}^{R}$ is short for $\sum_{r_1=0}^{R} \cdots \sum_{r_M=0}^{R}$ with $\sum_{m=1}^{M} r_m \leq R$

- The coefficients $\boldsymbol{\alpha}_{\mathbf{r},n,k}$ of the polynomial approximation can be obtained using least squares polynomial fitting

- The polynomial approximation $\tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)$ is calculated locally at each sensor

Approximation of the joint likelihood function

- Substituting $\tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)$ for $\mathbf{h}_{n,k}(\mathbf{x}_n)$ yields the approximation

$$S_n(\mathbf{z}_n, \mathbf{x}_n) \approx \tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n) \triangleq \sum_{k=1}^{K} [\mathbf{z}_{n,k} - \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)]^T \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)]$$

Approximation of the joint likelihood function

- Substituting $\tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)$ for $\mathbf{h}_{n,k}(\mathbf{x}_n)$ yields the approximation

$$S_n(\mathbf{z}_n, \mathbf{x}_n) \approx \tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n) \triangleq \sum_{k=1}^{K} [\mathbf{z}_{n,k} - \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)]^T \mathbf{Q}_{n,k}^{-1} [\mathbf{z}_{n,k} - \tilde{\mathbf{h}}_{n,k}(\mathbf{x}_n)]$$

- $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)$ can be rewritten as

$$\begin{aligned} \tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n) &= \sum_{k=1}^{K} \left[\sum_{\mathbf{r}=\mathbf{0}}^{2R} \beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})\, p_{\mathbf{r}}(\mathbf{x}_n) \right] \\ &= \sum_{\mathbf{r}=\mathbf{0}}^{2R} T_{\mathbf{r},n}(\mathbf{z}_n)\, p_{\mathbf{r}}(\mathbf{x}_n)\,. \end{aligned}$$

This is a polynomial in $\mathbf{x}_n$, of degree $2R$, with coefficients

$$T_{\mathbf{r},n}(\mathbf{z}_n) = \sum_{k=1}^{K} \beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})$$

Likelihood consensus

- The coefficients $T_{\mathbf{r},n}(\mathbf{z}_n)$ can be viewed as a sufficient statistic describing $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)$ and, in turn, the approximate JLF

$$\tilde{f}(\mathbf{z}_n|\mathbf{x}_n) \propto \exp\left(-\frac{1}{2}\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)\right) = \exp\left(-\frac{1}{2}\sum_{\mathbf{r}=\mathbf{0}}^{2R} T_{\mathbf{r},n}(\mathbf{z}_n)\, p_{\mathbf{r}}(\mathbf{x}_n)\right)$$

Likelihood consensus

- The coefficients $T_{\mathbf{r},n}(\mathbf{z}_n)$ can be viewed as a sufficient statistic describing $\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)$ and, in turn, the approximate JLF

$$\tilde{f}(\mathbf{z}_n|\mathbf{x}_n) \propto \exp\left(-\frac{1}{2}\tilde{S}_n(\mathbf{z}_n, \mathbf{x}_n)\right) = \exp\left(-\frac{1}{2}\sum_{\mathbf{r}=\mathbf{0}}^{2R} T_{\mathbf{r},n}(\mathbf{z}_n)\, p_{\mathbf{r}}(\mathbf{x}_n)\right)$$

- All $T_{\mathbf{r}}(\mathbf{z}_n) = \sum_{k=1}^{K} \beta_{\mathbf{r},k}(\mathbf{z}_{n,k})$ can be computed at each sensor using a distributed, iterative consensus algorithm that requires only communications between neighboring sensors:
  - sensor k computes $\beta_{\mathbf{r},n,k}(\mathbf{z}_{n,k})$ from locally available data ($\mathbf{z}_{n,k}$, $\boldsymbol{\alpha}_{\mathbf{r},n,k}$, $\mathbf{Q}_{n,k}$)
  - the sum over all sensors ($\sum_{k=1}^{K}$) is computed in a distributed way using a consensus algorithm (requires transmission of partial sums to neighbors)

Outline

- Distributed estimation in wireless sensor networks
- Distributed Gaussian particle filtering
- Likelihood consensus
- Distributed Gaussian particle filtering with reduced complexity
- Target tracking example and simulation results

DGPF with reduced complexity

- Each of the K local Gaussian particle filters uses a reduced number of particles, $J' = J/K$, to calculate a partial mean and a partial covariance of the global posterior

DGPF with reduced complexity

- Each of the K local Gaussian particle filters uses a reduced number of particles, $J' = J/K$, to calculate a partial mean and a partial covariance of the global posterior
- The partial means and covariances calculated at the individual sensors are combined by means of a second stage of consensus algorithms

DGPF with reduced complexity

- Each of the K local Gaussian particle filters uses a reduced number of particles, $J' = J/K$, to calculate a partial mean and a partial covariance of the global posterior
- The partial means and covariances calculated at the individual sensors are combined by means of a second stage of consensus algorithms
- If the second-stage consensus algorithms are sufficiently converged, the performance (estimation accuracy) of the DGPF with reduced complexity is effectively equal to that of the original DGPF

DGPF with reduced complexity

- Each of the K local Gaussian particle filters uses a reduced number of particles, $J' = J/K$, to calculate a partial mean and a partial covariance of the global posterior
- The partial means and covariances calculated at the individual sensors are combined by means of a second stage of consensus algorithms
- If the second-stage consensus algorithms are sufficiently converged, the performance (estimation accuracy) of the DGPF with reduced complexity is effectively equal to that of the original DGPF
- The computational complexity is substantially reduced at the cost of some increase in local communications

Outline

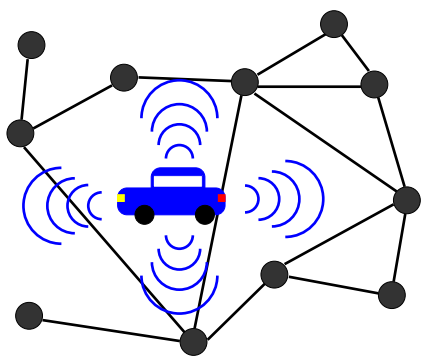
- Distributed estimation in wireless sensor networks
- Distributed Gaussian particle filtering
- Likelihood consensus
- Distributed Gaussian particle filtering with reduced complexity
- Target tracking example and simulation results

Application to target tracking

- The state $\mathbf{x}_n = (x_n\ y_n\ \dot{x}_n\ \dot{y}_n)^T$ represents the 2D position and 2D velocity of a target in the x-y plane at time n
- The state evolves according to the state-transition equation $\mathbf{x}_n = \mathbf{G}\mathbf{x}_{n-1} + \mathbf{u}_n$, where $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_u)$

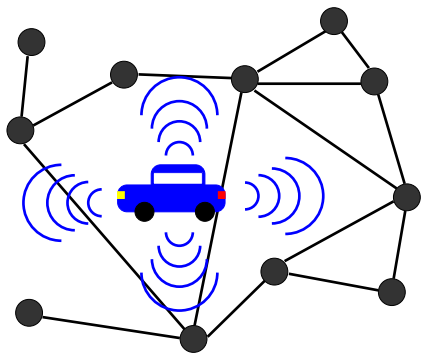
Application to target tracking

- The state $\mathbf{x}_n = (x_n\ y_n\ \dot{x}_n\ \dot{y}_n)^T$ represents the 2D position and 2D velocity of a target in the x-y plane at time n
- The state evolves according to the state-transition equation $\mathbf{x}_n = \mathbf{G}\mathbf{x}_{n-1} + \mathbf{u}_n$, where $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_u)$
- Randomly deployed acoustic amplitude sensors sense the sound emitted by the target



Application to target tracking

- The state $\mathbf{x}_n = (x_n\ y_n\ \dot{x}_n\ \dot{y}_n)^T$ represents the 2D position and 2D velocity of a target in the x-y plane at time n
- The state evolves according to the state-transition equation $\mathbf{x}_n = \mathbf{G}\mathbf{x}_{n-1} + \mathbf{u}_n$, where $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_u)$
- Randomly deployed acoustic amplitude sensors sense the sound emitted by the target



- Sensor measurement model:

$$z_{n,k} = h_{n,k}(\mathbf{x}_n) + v_{n,k} = \frac{A}{\|(x_n\ y_n)^T - \boldsymbol{\xi}_{n,k}\|} + v_{n,k}\,,$$

where $\boldsymbol{\xi}_{n,k}$ is the position of sensor k at time n and $v_{n,k} \sim \mathcal{N}(0, \sigma_v^2)$

Simulation settings

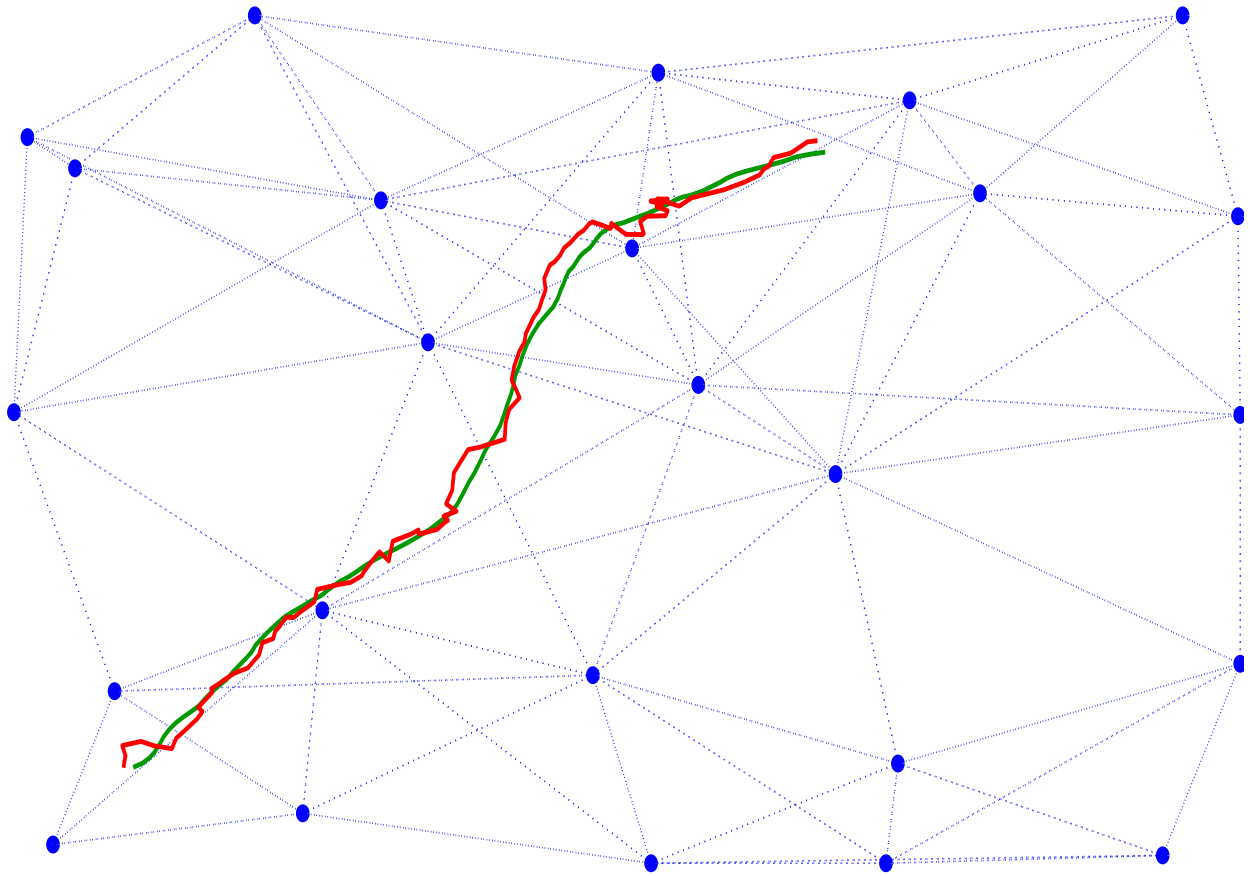
- Sensor network:
  - deployed over a field of dimensions 200m $\times$ 200m
  - consists of 25 acoustic amplitude sensors
  - each sensor communicates with sensors within a 90m range

Simulation settings

- Sensor network:
  - deployed over a field of dimensions 200m $\times$ 200m
  - consists of 25 acoustic amplitude sensors
  - each sensor communicates with sensors within a 90m range

- Particle filtering and likelihood consensus:
  - the proposed distributed Gaussian particle filter (DGPF) and its reduced complexity version (DGPF-R) are compared with a centralized Gaussian particle filter (CGPF)
  - number of particles $J = 1000$
  - the measurement functions $h_{n,k}(\mathbf{x}_n)$ are approximated by a polynomial of degree $R = 2 \;\Rightarrow\;$ 14 consensus algorithms executed in parallel
  - consensus using 8 iterations is compared with exact sum calculation

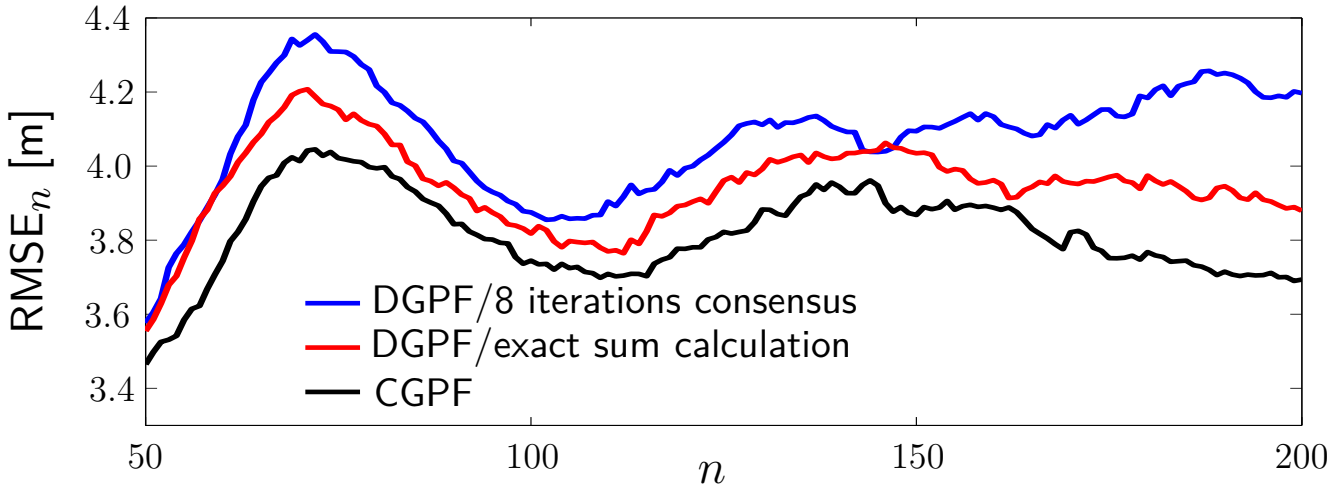
Simulation results



- Target trajectory generated by means of the state-space model (green) and trajectory estimated by the DGPF (red).

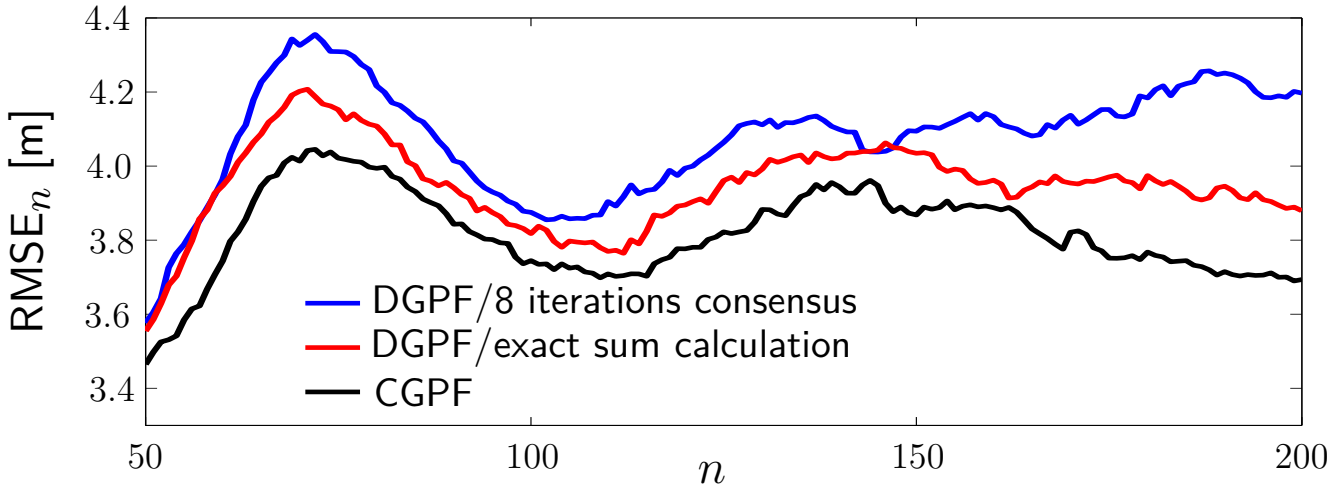
Simulation results

- Root mean square error (RMSE) versus time n:

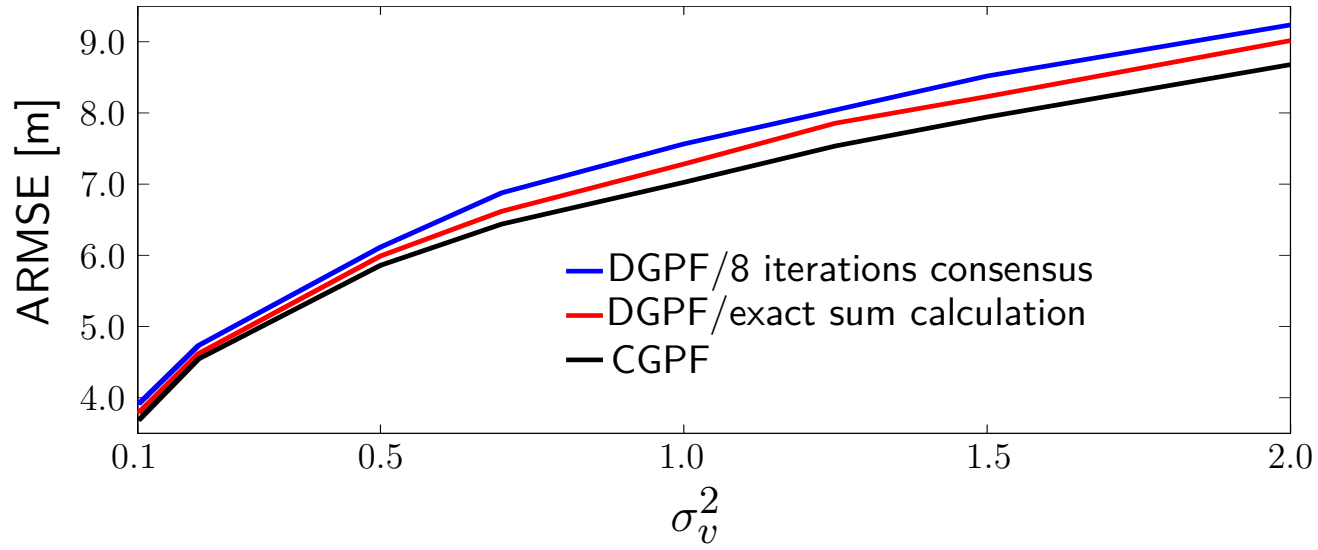


Simulation results

- Root mean square error (RMSE) versus time n:

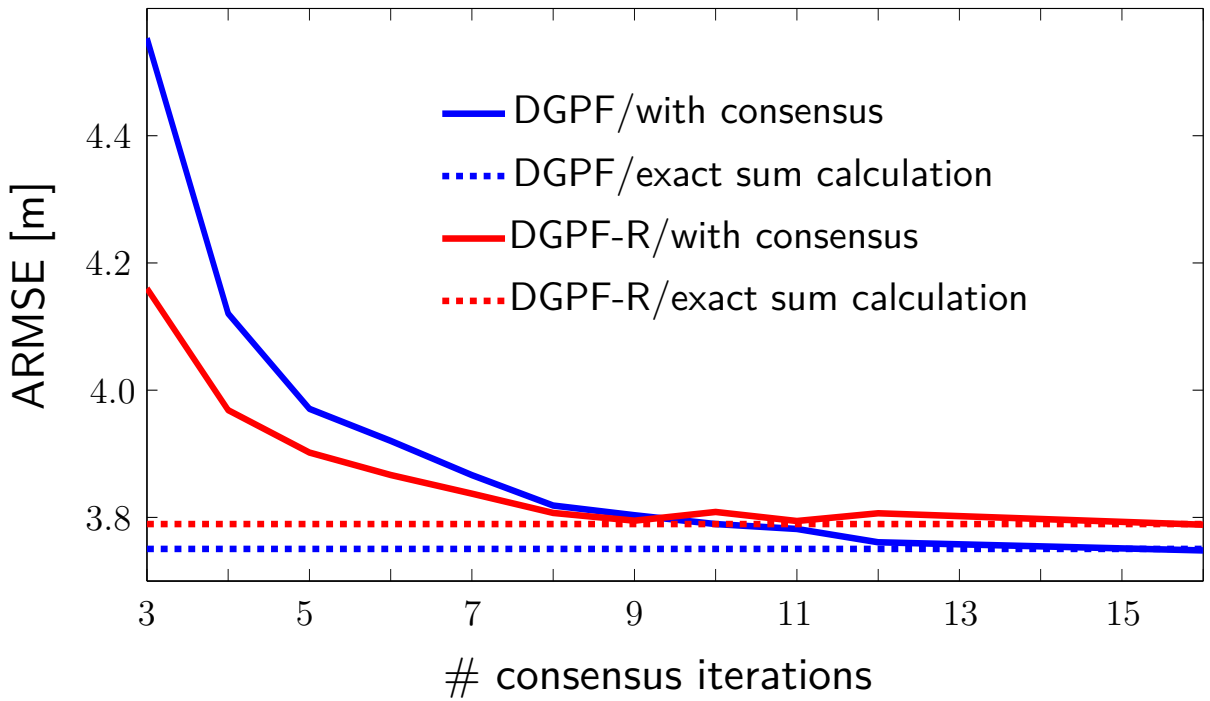


- Average RMSE (ARMSE) versus measurement noise variance:



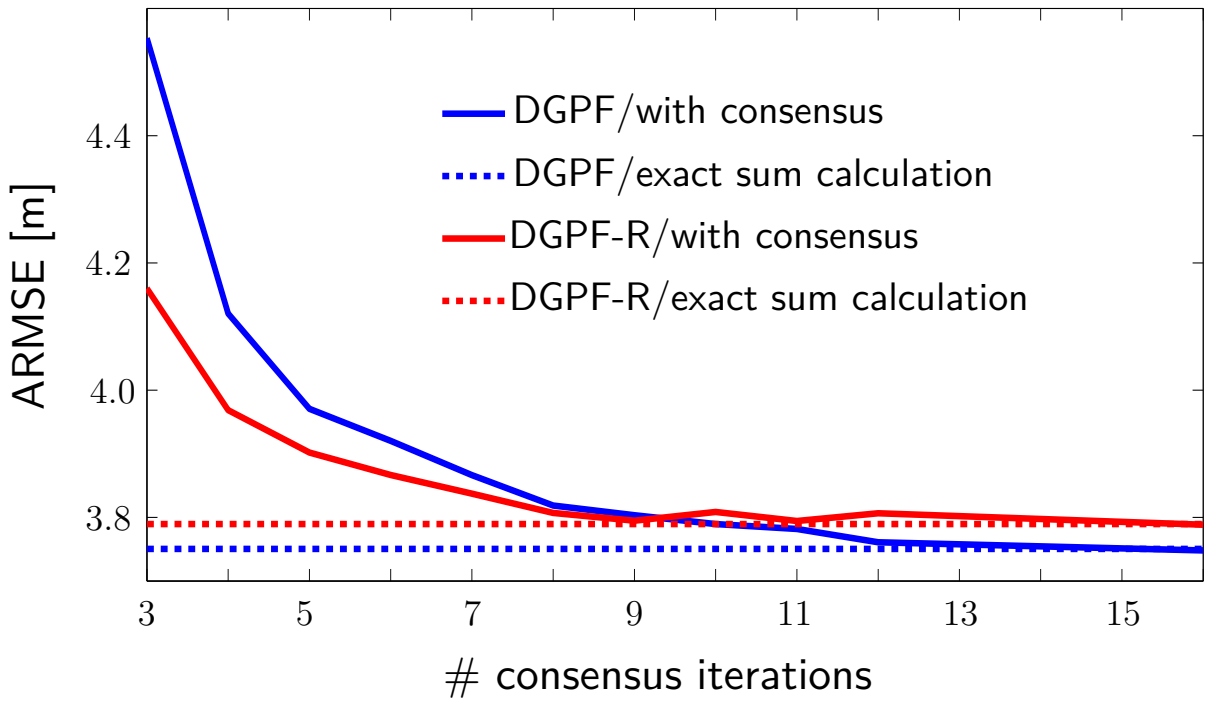
Simulation results

- ARMSE versus number of consensus iterations:



Simulation results

- ARMSE versus number of consensus iterations:



- As the number of consensus iterations increases, the performance of the DGPF approaches the performance of a DGPF with exact sum calculation
- For a small number of consensus iterations, the DGPF-R outperforms the DGPF

Conclusion

- We proposed a distributed Gaussian particle filtering scheme, where each sensor runs a local Gaussian particle filter that computes a global state estimate reflecting the measurements of all sensors
- The particle weights at each sensor are updated using the joint likelihood function, which is obtained in a distributed way via likelihood consensus

Conclusion

- We proposed a distributed Gaussian particle filtering scheme, where each sensor runs a local Gaussian particle filter that computes a global state estimate reflecting the measurements of all sensors
- The particle weights at each sensor are updated using the joint likelihood function, which is obtained in a distributed way via likelihood consensus
- Likelihood consensus requires only local communications of "sufficient statistics." No measurements or particles need to be communicated
- Suitable for dynamic sensor networks

Conclusion

- We proposed a distributed Gaussian particle filtering scheme, where each sensor runs a local Gaussian particle filter that computes a global state estimate reflecting the measurements of all sensors
- The particle weights at each sensor are updated using the joint likelihood function, which is obtained in a distributed way via likelihood consensus
- Likelihood consensus requires only local communications of "sufficient statistics." No measurements or particles need to be communicated
- Suitable for dynamic sensor networks
- We also proposed a reduced-complexity variant of the distributed Gaussian particle filter in which the number of particles is substantially reduced

Conclusion

- We proposed a distributed Gaussian particle filtering scheme, where each sensor runs a local Gaussian particle filter that computes a global state estimate reflecting the measurements of all sensors
- The particle weights at each sensor are updated using the joint likelihood function, which is obtained in a distributed way via likelihood consensus
- Likelihood consensus requires only local communications of "sufficient statistics." No measurements or particles need to be communicated
- Suitable for dynamic sensor networks
- We also proposed a reduced-complexity variant of the distributed Gaussian particle filter in which the number of particles is substantially reduced
- Simulation results indicate good performance even in comparison with a centralized Gaussian particle filter

Thank you