FAST AND ACCURATE COOPERATIVE LOCALIZATION IN WIRELESS SENSOR NETWORKS

Fabian Scheidt, Di Jin, Michael Muma and Abdelhak M. Zoubir

Signal Processing Group
Technische Universität Darmstadt
Darmstadt, Germany
{fscheidt, djin, mmuma, zoubir}@spg.tu-darmstadt.de

ABSTRACT

Cooperative localization capability is a highly desirable characteristic of wireless sensor networks. It has attracted considerable research attention in academia and industry. The sum-product algorithm over a wireless sensor network (SPAWN) is a powerful method to cooperatively estimate the positions of many sensors (agents) using knowledge of the absolute positions of a few sensors (anchors). Drawbacks of the SPAWN, however, are its high computational complexity and communication load. In this paper we address the complexity issue, reformulate it as convolution problem and utilize the fast Fourier transform (FFT), culminating in a fast and accurate localization algorithm, which we named SPAWN-FFT. Our simulation results show SPAWN-FFT's superiority over SPAWN regarding the computational effort, while maintaining its full flexibility and localization performance.

Index Terms— Cooperative localization, SPAWN, FFT, Kernel bandwidth, Efficient computation

1. INTRODUCTION

Accurate and low-cost sensor position information is a critical requirement for a wide variety of applications. In cooperative localization, sensors work together to share and process measurements in order to build a map of the network [1, 2]. Due to their ability to incorporate prior information in form of a distribution, Bayesian methods have become popular to tackle localization problems. Their main drawback is often the intractable analytical integration. One possible solution is to express the problem as a global joint distribution, map it on a factor graph and apply the sum-product algorithm (SPA) to calculate efficiently the marginals. Its use in the context of wireless sensor networks gives rise to SPAWN as detailed in [3, 4]. While this algorithm suffered from high computational complexity and communication load, [5] provided some efficient solutions to this problem. In this paper, we propose a new approach that scales down the message multiplication's complexity of the SPAWN from $\mathcal{O}(R^2)$ to $\mathcal{O}(R)$, where R is the number of particles. This is achieved by reformulating the kernel density estimation (KDE) of the SPAWN such it can be computed via an FFT and interpolation, culminating into a new algorithm, the SPAWN-FFT.

In the ensuing paper, Section 2 serves as problem formulation, Section 3 shortly recapitulates the SPAWN's essentials and Section 4 introduces the SPAWN-FFT. In Section 5 simulation results are presented and Section 6 concludes this work.

2. PROBLEM FORMULATION

A typical localization problem in a wireless sensor network involves N sensors, divided into a subset of anchors and agents. The anchors have exact location knowledge either obtained from a global positioning system or hard-programmed during the network deployment phase. The aim is to localize the agents using communication between the sensor nodes that are within communication range. This paper considers only the $2d$ scenario, but the $3d$ extension is straightforward. A sensor's location will be denoted by $\mathbf{x} = [x, y]^T$. The sensor's communication range is bounded to R_c. Furthermore $\Gamma_{\rightarrow i}$ denotes node i's set of neighbors. The localization problem is modeled stochastically in the Bayesian framework and our statistical signal model is

$$z_{j,i} = d_{j,i} + \eta_{j,i}, \tag{1}$$

where $z_{j,i}$ denotes the noisy distance measurement, $d_{j,i}$ the true Euclidean distance and $\eta_{j,i}$ the measurement noise, for nodes i and j, respectively. The complete set of measurements is denoted by $\mathbf{z}$. Given the set of measurements from (1) along with the assumptions that (i) all node's prior are mutually independent $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{N} f_i(\mathbf{x}_i)$, (ii) the measurements are mutually conditionally independent, $f(\mathbf{z}|\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{i=1}^{N} \prod_{j \in \Gamma_{\rightarrow i}} f(z_{j,i}|\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and (iii) that the relative measurements $z_{j,i}$ depend only on the receivers and transmitters respective position, i.e., $f(z_{j,i}|\mathbf{x}_1, \ldots, \mathbf{x}_N) = f(z_{j,i}|\mathbf{x}_i, \mathbf{x}_j)$, the global joint posterior distribution can be formulated as,

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_N|\mathbf{z}) \propto \prod_{i=1}^{N} f_i(\mathbf{x}_i) \prod_{j \in \Gamma_{\rightarrow i}} f(z_{j,i}|\mathbf{x}_i, \mathbf{x}_j). \tag{2}$$

The aim is to marginalize (2), using the SPA and obtain $f(\mathbf{x}_i|\mathbf{z})$. The final point estimates are easily obtained from the posterior mean, the posterior mode or posterior median.

This work was supported by the project HANDiCAMS which acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 323944.

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

3. THE SUM-PRODUCT ALGORITHM OVER A WIRELESS SENSOR NETWORK

The SPA's application to the localization problem is conducted iteratively, due to unavoidable cycles in the graph. The algorithm runs for a predefined number of iterations with iteration index $\ell = 1, \ldots, N_{\text{iter}}$. Based on range measurements SPAWN calculates for each sensor, during each iteration, the so-called internal message and belief update via,

$$\mathcal{I}_{j,i}^{(\ell)}(\mathbf{x}_i) \propto \int f(z_{j,i}|\mathbf{x}_i, \mathbf{x}_j)\mathcal{B}_j^{(\ell)}(\mathbf{x}_j)d\mathbf{x}_j, \tag{3}$$

$$\mathcal{B}_i^{(\ell+1)}(\mathbf{x}_i) \propto f_i(\mathbf{x}_i) \prod_{j\in\Gamma_{\to i}} \mathcal{I}_{j,i}^{(\ell)}(\mathbf{x}_i). \tag{4}$$

The internal message (3) is computed locally at node i and contributes to the belief update (4). The belief indicates node i's uncertainty regarding its true location $\mathbf{x}_i$.

3.1. The Classical (particle based) SPAWN

The non-linear relationship in $f(z_{j,i}|\mathbf{x}_i, \mathbf{x}_j)$ and the non-Gaussian uncertainty of $\mathcal{B}_j^{(\ell)}(\mathbf{x}_j)$ gives rise to Monte Carlo solutions using particles and associated weights, known as importance sampling. This culminates into the particle based SPAWN, inspired by [6]. In the ℓ-th iteration node i broadcasts its belief, expressed as particles $\mathbf{x}_i^{r,(\ell)}$ and weights $w_i^{r,(\ell)}$, forming the set $\{\mathbf{x}_i^{r,(\ell)}, w_i^{r,(\ell)}\}_{r=1}^R$. Receiving the neighbors beliefs $\mathcal{B}_j^{(\ell)}(\mathbf{x}_j)\ \forall j \in \Gamma_{\to i}$, node i filters them with its own measurements forming the internal message $\mathcal{I}_{j,i}^{(\ell)}(\mathbf{x}_i)$, being approximated as $\{\mathbf{x}_{j,i}^{r,(\ell)}, w_{j,i}^{r,(\ell)}\}_{r=1}^R$. As an approximation to the updated belief $\mathcal{B}_i^{(\ell+1)}(\mathbf{x}_i)$, a set of particles $\{\mathbf{x}_i^{r,(\ell+1)}\}_{r=1}^{\alpha R}$ is drawn from a proposal distribution $q^{(\ell)}(\mathbf{x}_i)$. In line with [6] this could be the incoming messages sum,

$$q^{(\ell)}\left(\mathbf{x}_i^{r,(\ell+1)}\right) = \sum_{j\in\Gamma_{\to i}} \mathcal{I}_{j,i}^{(\ell)}\left(\mathbf{x}_i^{r,(\ell+1)}\right). \tag{5}$$

Then, the associated importance weights are computed as,

$$w_i^{r,(\ell+1)} \propto \frac{f_i\left(\mathbf{x}_i^{r,(\ell+1)}\right)\prod_{j\in\Gamma_{\to i}} \mathcal{I}_{j,i}^{(\ell)}\left(\mathbf{x}_i^{r,(\ell+1)}\right)}{q^{(\ell)}\left(\mathbf{x}_i^{r,(\ell+1)}\right)}. \tag{6}$$

The $\mathcal{I}_{j,i}^{(\ell)}(\mathbf{x}_i)$ is approximated via KDE as,

$$\mathcal{I}_{j,i}^{(\ell)}\left(\mathbf{x}_i^{r,(\ell+1)}\right) = \sum_{\kappa=1}^{R} w_{j,i}^{r,(\ell)} \mathcal{K}_{\mathbf{H}}\left(\mathbf{x}_i^{r,(\ell+1)} - \mathbf{x}_{j,i}^{\kappa,(\ell)}\right). \tag{7}$$

In this context, $\mathcal{K}_{\mathbf{H}}(\cdot)$ denotes the kernel and $\mathbf{H}$ the bandwidth matrix which is either chosen as diagonal or unconstrained. While the kernel choice is of minor impact and is often chosen as Gaussian, bandwidth selection is crucial and offers performance gains for the algorithm. To avoid the impoverishment, resampling of $\{\mathbf{x}_i^{r,(\ell+1)}, w_i^{r,(\ell+1)}\}_{r=1}^R$ should be conducted, leading to a new set of particles with equal weights. The calculation of (7) is of complexity $\mathcal{O}(R^2)$ for a given bandwidth matrix. An automatic data driven calculation of $\mathbf{H}$ involves a further computation of complexity $\mathcal{O}(R^2)$ [7], so the rule-of-thumb technique should be utilized, if speed is of major concern. The communication load is $2R$ real numbers.

4. THE SPAWN-FFT

The main drawback of SPAWN is its message multiplication's complexity, driven by the involved KDE. A key observation regarding (7) is, that this operation is a convolution between the data point masses and the chosen kernel function. As such, the discrete convolution theorem can be applied along with the FFT for efficient calculation. In the following we only describe the technique introduced by [7–9] which is limited to product kernels. Even though this is sufficient for SPAWN, recent improvements by [10] made unconstrained bandwidth matrices available, for the price of increased computational effort. In SPAWN this procedure involves the following steps: first, the particle data are quantized. Second, the problem is reformulated as discrete convolution and solved using the two-dimensional FFT. Third, the internal message is sampled using an interpolation technique. An optional fourth step uses a Gaussian mixture clustering and expresses the node's belief in parameters.

4.1. Data Quantization

In a first step the set $\{\mathbf{x}_{j,i}^{r,(\ell)}, w_{j,i}^{r,(\ell)}\}_{r=1}^R$ is used to determine an appropriate bandwidth matrix, where a diagonal one is sufficient. In a subsequent step, the particles $\{\mathbf{x}_{j,i}^{r,(\ell)}\}_{r=1}^R$ are bounded into a box with lower and upper bounds

$$\mathbf{a} = \left[\min\left(x_{j,i}^{r,(\ell)}\right) - 3h_x; \min\left(y_{j,i}^{r,(\ell)}\right) - 3h_y\right], \tag{8}$$

$$\mathbf{b} = \left[\max\left(x_{j,i}^{r,(\ell)}\right) + 3h_x; \max\left(y_{j,i}^{r,(\ell)}\right) + 3h_y\right]. \tag{9}$$

The cash of $\pm 3h$ is proposed in [8] to avoid problems with the FFT's wrap-around edge conditions and guard the target density from numerical corruption, as illustrated in Figure 1. Then the data are quantized into $\mathcal{M}$ levels for each dimension. Classical techniques are the simple and linear procedure from [11].

For instance, the following simple rule determines a quantization step size as $\delta_d = (b_d - a_d)/\mathcal{M}$ and discretizes the data as,

$$k_d = \text{floor}(1 + (r_d - a_d)/\delta_d), \tag{10}$$

where k_d represents the index and r_d the r-th particle in the d-th dimension, such as x and y. For each data point r_d, a grid count is determined as,

$$\nu_{k_x,k_y} = \nu_{k_x,k_y} + 1. \tag{11}$$

This delivers a matrix of grid-counts $\boldsymbol{\mathcal{C}}_{\mathcal{M}\cdot\mathcal{M}}$.

4.2. Two-Dimensional Fourier Transformation

Given the grid counts, the next step is to determine the kernel values. For this purpose one must determine $\delta_d = (b_d - a_d)/(\mathcal{M}-1)$ for each dimension. The next step is to introduce an effective support parameter τ. For the Gaussian kernel this allows a truncation and all kernel evaluations for $|r_d\delta_d/h_d| \geq \tau$ can be omitted. Furthermore it is sufficient to define some $L_d = \min\{\mathcal{M}-1, \text{floor}(\tau h_d/\delta_d)\}$ for each dimension and subsequently define some $l_d = (0 : L_d)\delta_d$ for each dimension as well. The last quantity represents the sample values for the kernel function. Given these quantities a highly composite number of 2 is determined as $P_d = 2^{\text{ceil}(\log_2(L_d+\mathcal{M}))}$ and $P_x = P_y$ should be ensured. Then two convolution matrices are generated. The grid-count matrix $\mathbf{C}_{P_x \cdot P_y}$ is

$$\mathbf{C} = \begin{bmatrix} \boldsymbol{\mathcal{C}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{12}$$

with $\boldsymbol{\mathcal{C}}$ in its first $1, \ldots, \mathcal{M}$ rows and columns. Second the kernel matrix $\mathbf{K}_{P_x \cdot P_y}$ is given by,

$$\mathbf{K} = \begin{bmatrix} k_{0,0} & \cdots & k_{0,L_2} & & k_{0,L_2} & \cdots & k_{0,1} \\ \vdots & \ddots & \vdots & \mathbf{0} & \vdots & \ddots & \vdots \\ k_{L_1,0} & \cdots & k_{L_1,L_2} & & k_{L_1,L_2} & \cdots & k_{L_1,1} \\ \mathbf{0} & & & \mathbf{0} & & \mathbf{0} & \\ k_{L_1,0} & \cdots & k_{L_1,L_2} & & k_{L_1,L_2} & \cdots & k_{L_1,1} \\ \vdots & \ddots & \vdots & \mathbf{0} & \vdots & \ddots & \vdots \\ k_{1,0} & \cdots & k_{1,L_2} & & k_{1,L_2} & \cdots & k_{1,1} \end{bmatrix}. \tag{13}$$

The quantities (12) and (13) are zero-padded and for the latter one, the wrap-around ordering is utilized. Furthermore it is remarkable that only $2L_d$ kernel evaluations are necessary, regardless of the value of R, delivering enormous computational savings compared to (7). Subsequently (12) and (13) are transformed into the frequency domain using the FFT and the convolution problem is easily solved as

$$\mathbf{C}_{\mathcal{F}} = \mathcal{F}(\mathbf{C}), \tag{14}$$

$$\mathbf{K}_{\mathcal{F}} = \mathcal{F}(\mathbf{K}), \tag{15}$$

$$\mathbf{P}_{\mathcal{F}} = \mathbf{C}_{\mathcal{F}} \odot \mathbf{K}_{\mathcal{F}}. \tag{16}$$

In this context the matrix (16) is the Hadamard-Schur product of (14) and (15). Since P_d is already a highly composite number, further point adjustment for the FFT is superfluous. The searched density is obtained via IFFT applied to (16) as,

$$\mathbf{P} = \mathcal{F}^{-1}(\mathbf{P}_{\mathcal{F}}), \tag{17}$$

and a normalization step is carried out as $\mathbf{P} = \mathbf{P}/\prod_{d=1}^{2} P_d$. For unconstrained bandwidth support, equations (12) and (13) must be adjusted as described in [10]. However, this results in the loss of symmetries and increased computational effort.

4.3. Internal Message Interpolation

From the entries of (17) the searched density can now be sampled from $\mathbf{P}$ via its first $1, \ldots, \mathcal{M}$ rows and columns respectively, delivering a sub-matrix $\tilde{\boldsymbol{P}}$. It is noteworthy that due to numerical rounding errors, introduced from the FFT, several entries may be negative. A logical check can replace them with the machine precision number. The next step concerns the actual evaluation of the kernel at the temporal belief particles $\{\mathbf{x}_i^{r,(\ell)}\}_{r=1}^{\alpha R}$. We use *bilinear interpolation* for this task with the aforementioned particles as input. Its application requires the creation of a rect-linear $2d$ interpolation grid. This can be done by creating a linear mesh between (8) and (9) using $\mathcal{M}$ points. Given the value of $\tilde{\boldsymbol{P}}$ at four grid points, $\boldsymbol{\Omega}_{11} = [x_1, y_1]$, $\boldsymbol{\Omega}_{12} = [x_1, y_2]$, $\boldsymbol{\Omega}_{21} = [x_2, y_1]$ and $\boldsymbol{\Omega}_{22} = [x_2, y_2]$, interpolation in the x-direction delivers

$$\mathcal{I}_{j,i}^{(\ell)}([x, y_1]) \approx \frac{x_2 - x}{x_2 - x_1}\tilde{\boldsymbol{P}}(\boldsymbol{\Omega}_{11}) + \frac{x - x_1}{x_2 - x_1}\tilde{\boldsymbol{P}}(\boldsymbol{\Omega}_{21}),$$

$$\mathcal{I}_{j,i}^{(\ell)}([x, y_2]) \approx \frac{x_2 - x}{x_2 - x_1}\tilde{\boldsymbol{P}}(\boldsymbol{\Omega}_{12}) + \frac{x - x_1}{x_2 - x_1}\tilde{\boldsymbol{P}}(\boldsymbol{\Omega}_{22}).$$

Subsequently proceeding in the y-direction delivers the desired estimate,

$$\mathcal{I}_{j,i}^{(\ell)}(\mathbf{x}_i) \approx \frac{y_2 - y}{y_2 - y_1}\mathcal{I}_{j,i}^{(\ell)}([x, y_1]) + \frac{y - y_1}{y_2 - y_1}\mathcal{I}_{j,i}^{(\ell)}([x, y_2]). \tag{18}$$

For particles outside the box, we extrapolate by the machine precision number (eps) in order to avoid numerical instabilities, guarantee matching output vectors and prevent the proposal distribution from destruction. Therefore, with (18) an efficient solution to (7) is found and from a high level perspective the message multiplication's complexity scales down from $\mathcal{O}(R^2)$ to $\mathcal{O}(P\log(P) + R)$ in the worst case. Since $R \gg P\log(P)$ the complexity is approximately $\mathcal{O}(R)$ as compared in Table 1. For dimensional analysis see [6].

Algorithm	Complexity
SPAWN	$\mathcal{O}(R^2)$
SPAWN-FFT	$\mathcal{O}(R)$

Table 1: Comparison of message multiplication's complexity.

4.4. Belief Clustering

With the intention to reduce the algorithm's communication overhead of $2R$ belief particles, [12] demonstrated that clustering using a Gaussian mixture, allows to approximate this belief as,

$$\mathcal{B}_i^{(\ell)}(\mathbf{x}_i) \approx \sum_{k=1}^{K} \omega_i^{k,(\ell)} \mathcal{N}\left(\mathbf{x}_i; \boldsymbol{\mu}_i^{k,(\ell)}, \boldsymbol{\Sigma}_i^{k,(\ell)}\right), \tag{19}$$

where $\mathcal{N}$ represents a Gaussian with mean $\boldsymbol{\mu}_i^{k,(\ell)}$ and covariance $\boldsymbol{\Sigma}_i^{k,(\ell)}$ and $\omega_i^{k,(\ell)}$ are the mixing coefficients. The parameters can be determined via the EM-algorithm. Along with the proposal in [5] a further extension can be found utilizing an adaptive procedure such as the greedy EM. Therewith, instead of a large set of particles, only the parameters $\{\omega_i^{k,(\ell)}, \boldsymbol{\mu}_i^{k,(\ell)}, \boldsymbol{\Sigma}_i^{k,(\ell)}\}_{k=1}^{K}$ are required to transmit.

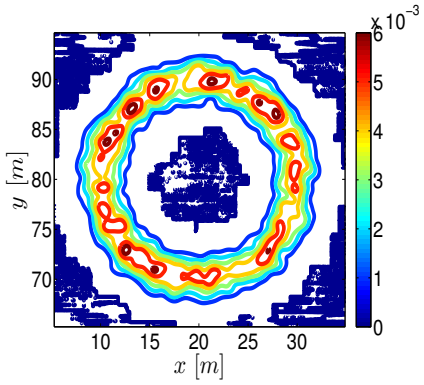

Fig. 1: Internal message with rounding errors and $3h$ buffer.

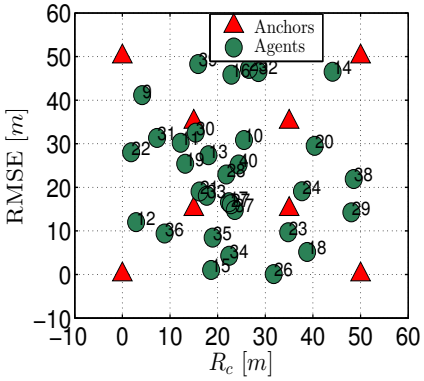

Fig. 2: Sensor network.

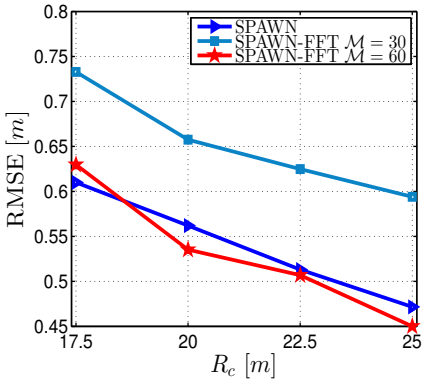

Fig. 3: RMSE localization performance over communication range R_c.

5. SIMULATION RESULTS

5.1. Simulation Setup

The subsequent simulation part examines the performance of the SPAWN-FFT algorithm and compares it with the classical SPAWN regarding their localization performance in terms of root-mean-square error performance (RMSE) and computational time for the full algorithm. Our simulation scenario is $50\ m\ \times\ 50\ m$ indoor with 32 agents and 8 anchors. Anchors are placed as nested squares, and agents are uniformly randomly distributed over the full deployment area, as depicted in Figure 2. The agent's belief is initialized as uniform over the full area and the communication range R_c ranging from 17.5 m to 25 m, with an increment of 2.5 m. Our measurement scenario is line-of-sight with $\eta_{j,i} \sim \mathcal{N}(0, (0.4)^2)$. As kernel function a standard Normal $\mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{H})$ is chosen and the bandwidth matrix is designed as diagonal with $h_d = \mathrm{std}\left(\mathbf{x}_{j,i}^{r,(\ell)}\right) R^{-1/3}$ and $\mathbf{H} = \mathrm{diag}(h_x^2, h_y^2)$.

The utilized quantization or binning strategy is Scott's simple binning from [11]. As point estimate the posterior mean is chosen. Furthermore, no message censoring scheme is utilized. For each of the following curve's points 100 Monte Carlo runs were conducted and the results were averaged over them. Those runs using the additional Gaussian mixture clustering are marked with GM. Table 2 lists additional simulation parameters.

Parameter	Value	Description
R	500	Number of particles
α	2	Sampling parameter
K	2	Number of mixture components
N_{iter}	10	Number of iterations
τ	3	Effective support parameter
$\mathcal{M}$	30 & 60	Level number

Table 2: Simulation parameters.

5.2. Simulation Results

At first we examined the localization performance of SPAWN and SPAWN-FFT. In this first step we did not utilize the Gaussian mixture clustering. From the results of Figure 3 it is obvious that the quantization level's choice $\mathcal{M}$ is of crucial impact. For $\mathcal{M} = 30$ SPAWN-FFT is inferior regarding its localization performance. With increased communication range, this becomes more evident. The reason can be found in the level number, as it determines the degree of approximation for the internal message and the details which can be stored. So in this case, the quantization error is large. The second observation is, that for $\mathcal{M} = 60$, SPAWN-FFT performs similarly as the SPAWN. With an increased level number, the quantization error becomes smaller, more details can be stored and according to Figure 3, SPAWN-FFT is even slight superior for communication ranges greater than around 18 m, which is most evident for 20 m. Nevertheless, one should be aware of numerical rounding errors and should conclude, that for increasing communication range, both algorithms tend to perform similarly.

Next to the algorithm's localization performance the simulation time was one of our major interests. For this purpose the execution time for the full algorithm was measured. The simulations were run on an Intel $i3\ 350m$ with 2.27 GHz and 8 GB memory. Figure 4's results reveal that SPAWN requires the largest amount of simulation time, as one could expect from (7)'s complexity, where a remarkable fact is, that the sample size is fixed and its complexity increases with a scaling parameter related to the increased communication range. Furthermore, as expected SPAWN-FFT clearly outperforms SPAWN regarding the simulation time. The fastest solution can be obtained for $\mathcal{M} = 30$, while increasing $\mathcal{M}$ to 60 only modest influences it. The results for localization performance and simulation time demonstrate SPAWN-FFT's capability to outperform SPAWN regarding the computational effort while preserving SPAWN's flexibility and localization performance.

In a final step, we compared the localization performance and simulation time of SPAWN-FFT with belief clustering. According to Figure 5 the localization performance is slightly improved, resulting from an additional amount of smooth-

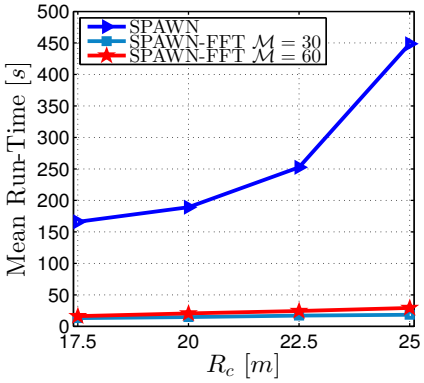

Fig. 4: Mean Run-Time performance over communication range R_c.

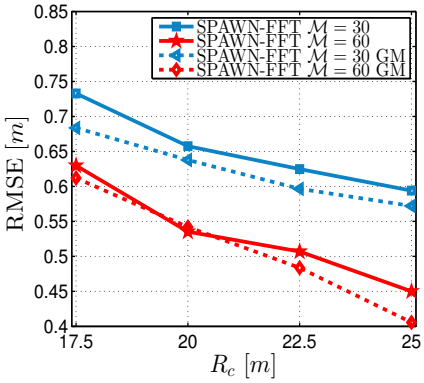

Fig. 5: RMSE localization performance over communication range R_c with Gaussian mixture GM.

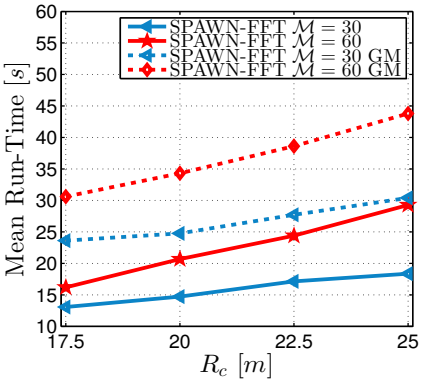

Fig. 6: Mean Run-Time performance over communication range R_c with Gaussian mixture GM.

ing, introduced by the Gaussian mixture. While the mixture model's order is fixed here, note that localization results are still improvable by a more appropriate model order selection strategy. The EM-algorithm's use increases the simulation time as demonstrated in Figure 6, but for a fixed component model the operation can be carried out in linear complexity. This demonstrates that belief clustering potentially improves the localization performance and effectively reduces the communication load at the cost of additional computational effort for clustering. Concluding it should be noted, that SPAWN and SPAWN-FFT tend to improve their performance as the number of particles increases. Nevertheless, for SPAWN this is more inconvenient than for SPAWN-FFT, first from the complexity of (7) as well as from the communication load.

6. CONCLUSIONS

With our work we proposed an extension to the SPAWN framework, the SPAWN-FFT algorithm. The proposed SPAWN-FFT algorithm reduces the computational complexity associated with message multiplication in the SPAWN while preserving its flexibility. As shown in our simulation results, the SPAWN-FFT significantly reduces the simulation time, and retains a similar localization performance as the SPAWN. Additionally, Gaussian mixture belief clustering reduces the communication load to the mixture's parameters.

References

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O Hero III, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, July 2005.

[2] M. Z. Win, A. Conti, S. Mazuelas, Y. Shen, W. M. Gifford, D. Dardari, and M. Chiani, "Network localization and navigation via cooperation," *IEEE Commun. Mag.*, vol. 49, no. 5, pp. 56–62, 2011.

[3] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[4] J. Lien, U. J. Ferner, W. Srichavengsup, H. Wymeersch, and M. Z. Win, "A comparison of parametric and sample-based message representation in cooperative localization," *International Journal of Navigation and Observation*, vol. 2012, Jul. 2012.

[5] D. Jin, F. Yin, C. Fritsche, A. M. Zoubir, F. Gustafsson, and AB Ericsson, "Efficient cooperative localization algorithm in LOS/NLOS environments," in *Signal Processing Conference (EUSIPCO), 2015 23rd European*. IEEE, 2015, pp. 185–189.

[6] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, 2005.

[7] M. P. Wand and M. C. Jones, *Kernel smoothing*, CRC Press, 1994.

[8] B. W. Silverman, *Density estimation for statistics and data analysis*, vol. 26, CRC Press, 1986.

[9] M. P. Wand, "Fast computation of multivariate kernel estimators," *J Comput. Graph. Stat.*, vol. 3, no. 4, pp. 433–445, 1994.

[10] A. Gramacki and J. Gramacki, "FFT-based fast computation of multivariate kernel estimators with unconstrained bandwidth matrices," *arXiv preprint arXiv:1508.02766*, 2015.

[11] D. W. Scott, *Multivariate density estimation: theory, practice, and visualization*, John Wiley & Sons, 2015.

[12] P.-A. Oikonomou-Filandras and K.-K. Wong, "Hybrid non-parametric belief propagation for localization in wireless networks," in *IEEE Sensor Signal Processing Defence (SSPD 2011)*. IET, 2011, pp. 1–5.