Unmanned Systems, Vol. 0, No. 0 (2013) 1–14
© World Scientific Publishing Company

A Particle Fusion Approach for Distributed Filtering and Smoothing

Tony X. Lin[a], Samuel Coogan[a], Donald A. Sofge[b], Fumin Zhang[a]

[a] *Georgia Institute of Technology, Atlanta, Georgia, United States of America*
E-mail: {`tlin339, sam.coogan, fumin`}@gatech.edu

[b] *Naval Research Laboratory, Washington, D.C., United States of America*
E-mail: `donald.sofge`@nrl.navy.mil

Distributed state estimation is an important tool for coordinated team decision-making and typically involves sharing information between robots in order to outperform individual state estimation. The shared information typically takes the form of relative measurements which allow team members to act as virtual sensors for other robots where the virtual sensor uncertainty is corrupted also by the team member's state uncertainty. However, incorporating relative measurements commonly depends on a pointwise product operation in most distributed estimation techniques which is well defined for continuously-valued distributions but ill-defined for particle-based distributions. We propose a drop-in replacement for the pointwise product based on using the generalized Hölder's inequality to upper-bound the product over a series of grid cell sets that discretize the state space. This upper-bound is well-defined for particle-based distributions and allows for tighter approximations by decreasing the volume of the sets. We leverage the approach to realize two distributed estimation strategies that use a pointwise product, the Kullback-Leibler Average and Belief Propagation and use these methods in simulations and experiments with a pair of miniature autonomous blimps.

Keywords: Particle Filtering; Distributed Estimation

1. Introduction

State estimation is a basic necessity to enable higher-level functionality such as feedback control and navigation. Traditionally, filtering and smoothing theories have fulfilled the core needs of effective estimation by providing both point estimates and their associated uncertainties in the form of a marginalized probability density function (PDF) when available information regarding the state(s) of interest are corrupted by noise. For example, the well-known Kalman Filter and Smoother[1, 2] provide optimal state estimates when the system offers linear dynamics and measurements and the corrupting noise is drawn from known Gaussian distributions. In nonlinear and non-Gaussian settings, the particle filter and smoother are used instead and can handle arbitrary noise distributions, dynamics, and measurements at the expense of greater memory requirements.[3] Filtering and smoothing strategies though are ill-defined for dealing with relative measurements between multiple state instances of the system. Such measurements arise in distributed settings, where robots may observe each other, and in simultaneous localization and mapping (SLAM) settings, where robots may observe and re-observe landmarks with unknown position, also known as loop closures. An distributed scenario is shown in Fig. 1 in which a blimp observes both landmarks and another blimp.

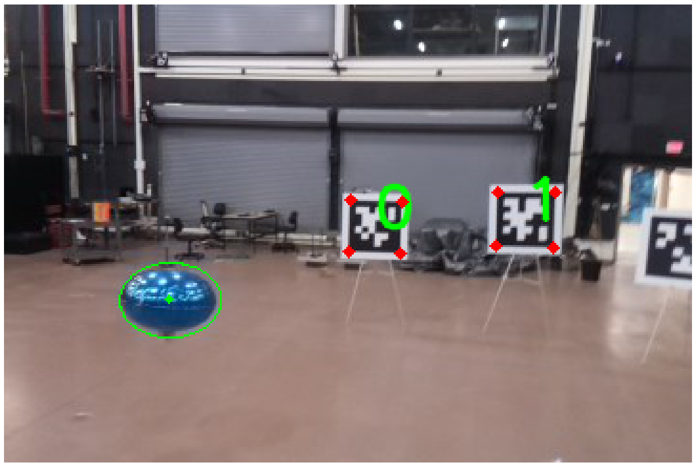


Fig. 1. First person view of a blimp detecting AprilTag landmarks and relative measurements of another blimp.

Traditionally, incorporating this information necessitates some form of distributed Bayesian fusion and allows for team members to act as specialized sensors for each other. However, Bayesian fusion strategies depend on a pointwise product between distributions which are ill-defined for particle-based distributions. In this work, we instead propose the use of an approximate pointwise product for a set of input particle distributions based on Hölder's inequality applied over a discretization of the state space. This upper-bound over each discretization allows us to define a re-weighting scheme across all particles provided in the set of input distributions. We demonstrate the effectiveness of this re-weighting strategy as a drop-in replacement for the true pointwise product by providing comparisons of the true pointwise product between two continuous distributions and the approximate pointwise product between samples of the aforementioned continuous distributions.

We then leverage this strategy to realize existing methods for distributed filtering and smoothing which depend on the pointwise product by substituting our approximate product in place of the true product when the representing distribution is based on particles. In the filtering case, we propose a distributed particle filter based on weighted Kullback-Leibler Averaging[4] (KLA) which computes an average of two distributions. In the smoothing case, we propose to improve trajectory estimates using the Belief Propagation algorithm,[5] which can be used to find approximate marginal PDFs for each state instance.[6] In both scenarios, these algorithms have been proposed for distributions that have well-defined pointwise products. In this work, we demonstrate that these approaches can be used for particle distributions as well by substituting our approximate product instead.

Our primary contributions are as follows: **i)** we propose a novel pointwise product approximation between particle distributions, **ii)** we incorporate this strategy into two well-known algorithms for filtering and smoothing in order to enable particle-based versions of the weighted KLA and Belief Propagation, and **iii)** we validate our proposed strategy using simulations and experiments with real-world miniature autonomous blimps. Different than our previous work,[7] the method proposed in the work uses an explicit upper-bound for the pointwise product in order to compute a re-weighting of particles. The proposed strategy in this work also leverages smaller discretizations of the state space in order to balance between more accurate estimates and faster computation.

This article is organized as follows. In section 2, we discuss closely related works in the domain of distributed particle-based filtering and smoothing. In Section 3, we introduce our approximate pointwise product strategy and provide examples. In Sections 4, we formulate the distributed filtering and distributed smoothing problems respectively and propose solutions that depend on our proposed product. We then validate our approaches in Section 5 using a pair of miniature autonomous blimps and conclude with final thoughts and future works in Section 6.

2. Related Works

Incorporating relative state measurements requires some fusion operation that can account for the uncertainty associated with the observed and observer robot states. In distributed settings, the literature has proposed various distributed Bayesian fusion methods. One method proposed in the SLAM domain proposes transforming relative measurements into single state measurements by treating the state as a growing-length trajectory consisting of all states from the initial time to the current time.[8,9] However, transforming state instances into growing-length trajectories can lead to difficulties due to either an increasingly expensive solution, as in the case of the Kalman Filter, or a loss of information in the earlier states, as in the Particle Filter.

Other strategies for approximating the pointwise product have also been explored in the literature and commonly fall under two categories, interpolation-based or sampling-based. In interpolation-based strategies, an intermediary function is fitted to the sample points in order to produce a continuous function that can be used for the pointwise product. Commonly, the choice of intermediary function is either a Gaussian-Mixture Model (GMM) or a Gaussian-based kernel strategy. These methods can produce good results but are limited in both their representation capabilities and their computation requirements. Gaussians and GMMs are ill-defined for rotations and struggle to represent arbitrary distributions if the distribution parameters are not well tuned (for example determining the number of GMM components). Kernel-based and GMM strategies also suffer from an increasing component count under pointwise product operations, as the pointwise product of N_d input Kernel-based or GMM distributions each with N_c components or bases leads to an output distribution with $N_c^{N_d}$ components or bases. To accommodate the growing memory requirements, pointwise product operations are typically followed by a downsampling procedure in which a new distribution of N_c components is found that minimizes some distance to the true $N_c^{N_d}$ product distribution which can be computationally expensive.[10] By comparison, sampling-based strategies are able to employ very efficient weighted re-sampling techniques to downsample the number of representative particles, although a direct pointwise product is impossible.

In sampling-based strategies, Gibbs sampling methods are typically employed to produce accurate estimates of the pointwise product. These strategies typically produce very accurate results but suffer from requiring an initial convergence period before beginning to sample from the pointwise product distribution. Another particular line of research[11,12] assumes the particle support positions are always identical. This is achieved via either an identical random seed for every distributed node or by maintaining a fixed set of support points common across all agents. While this strategy avoids all of the previously described issues, being unable to modify the particle positions dramatically reduces the tracking capability of the particle methods.

The proposed strategy of this work is most closely described as an interpolation-based strategy via the use of integrals over sets. However, our strategy doesn't require an intermediary representation which allows us to represent rotation and pose spaces and avoid expensive tuning difficulties. Our approach also doesn't require an intermediary sampling step which may be time-expensive. Most closely related to our work is a strategy[13] that proposes a pointwise product through the use of a k-nearest-neighbors (knn) kernel interpolation that produces a re-weighting for each point in the input distributions. However, the knn-based strategy can produce aberrant results when the input distributions have a minimally intersecting support while our approach produces either zero everywhere or a subsampling of the input distributions depending on the desired application. This approach has been faithfully recreated, to the best of our ability, and compared visually in order to demonstrate the differences during the minimally intersecting support case.

Particle-based distributed filtering typically focuses on circumventing the need for a pointwise-product by using some interpolating function[14,15] or by applying a consensus on the likelihood functions provided during sensor measurements.[16] Such strategies suffer from the previously described difficulties with interpolation methods or are only well-suited for distributed tracking of a common state vector, for example when multiple mobile sensors attempt to collaboratively track a human's movement through a house. Other strategies convert the relative measurement into a single state measurement by concatenating all distributed states into a single state[17,18] which grows as time increases. These methods can be effective at incorporating relative measurements but suffer in smoothing problems where initial distributions typically devolve to single sample PDFs.

Particle-based Belief Propagation for factor graphs have also been explored in the literature for estimation problems.[10,19] While these strategies are effective at solving Belief Propagation, these strategies leverage either interpolation-based or sampling-based methods to realize the pointwise-product for the Belief Propagation algorithm. As such, these strategies suffer from similar difficulties such as either being unable to accommodate rotations or requiring an initial sampling convergence time where the first set of samples are not useful.

3. A Particle Fusion Algorithm

We first propose a drop-in replacement strategy for pointwise product operations between particle distributions which we use to represent the marginal PDFs. Particle distributions are defined as

$$p(\boldsymbol{x}) = \sum_{i=1}^{N_p} w_i \delta(\boldsymbol{x} - \boldsymbol{z}_i) \tag{1}$$

in which δ is the dirac delta function and the N_p particles are described by a weight and position pair $(w_i, \boldsymbol{z}_i)$ such that $\sum_i w_i = 1$ and $w_i > 0$. Throughout this paper, we will use the notations $W_i = \{w_j^{(i)}\}_{j=1}^{N_p}$ and $Z_i = \{\boldsymbol{z}_j^{(i)}\}_{j=1}^{N_p}$ to describe the N_p weights and positions of a particle distribution. These distributions can incorporate nonlinear and non-Gaussian information but yield zero everywhere under pointwise product due to the dirac delta function. Nevertheless, particle distributions often suggest a non-zero product due to intermeshing particles.

An example of this can be observed in Fig. 2 in which two particle distributions have intermeshing particle positions but would yield a zero pointwise product everywhere. Noting that particles that are "close enough" should be somehow related in a product operation, we instead propose to discretize the space into sets and compute an approximation of the product over the sets.

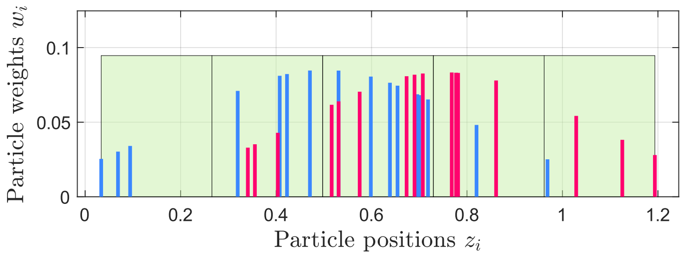


Fig. 2. Example of intermeshing particles in a two particle distribution setting with equal length grid sets computed over the combined set of both particle distributions.

3.1. *Approximate Particle Re-Weighting*

We introduce here a re-weighting strategy for a set of N_d particle distributions $\{(W_i, Z_i)\}_{i=1}^{N_d}$ in which the first distribution (W_1, Z_1) is re-weighted to approximate the pointwise product. As previously mentioned, our approach involves first computing grid cells that completely discretize the space covered by all particles.

We compute these sets as equal volume grid cells with outer boundaries defined by the minimum and maximum of the particle states. We assume a user-provided parameter $N_{sets} \in \mathbb{N}^{N_z}$ where N_z is the number of dimensions for each particle. We define

$$\begin{aligned} Z' &= \{\boldsymbol{z}'_i\}_{i=1}^{N_d \times N_p} = \cup_{i=1}^{N_z} \{\boldsymbol{z}_j^{(i)}\}_{j=1}^{N_p} \\ W' &= \{w'_i\}_{i=1}^{N_d \times N_p} = \cup_{i=1}^{N_z} \{w_j^{(i)}\}_{j=1}^{N_p} \end{aligned} \tag{2}$$

as notation for the set of all particle states and weights from N_d distributions to be multiplied. Then, the maximum and minimum particles are computed as $\bar{\boldsymbol{z}} = \sup\{Z'\}$, $\underline{\boldsymbol{z}} = \inf\{Z'\}$. For an angular dimension with index i, we define fixed boundary values as $\bar{z}_i = \pi$ and $\underline{z}_i = -\pi$ in order to accommodate the wrapping behavior of Euler angles. After computing the grid cell outer boundaries $(\bar{\boldsymbol{z}}, \underline{\boldsymbol{z}})$, we define constant volume grid cells by computing a width $d_i = \frac{\bar{z}_i - \underline{z}_i}{N_{sets,i}}$ along each dimension and use the notation $\mathcal{D}_i$, $i = 1, 2, \ldots, N_{sets,1} \times \ldots \times N_{sets,N_z}$ to indicate each of the grid cells. By defining sets in a grid cell fashion, note that all particles are also members of exactly one set.

Remark 3.1 (Choice of Set Definition). The choice of set here consists of axis-aligned grid cells that are divided evenly to cover the particle locations. While other sets could be defined, such as subdividing sets in regions of higher particle concentrations, or other shape sets as in our previous work,[7] we choose grid cells due to their simplicity and efficiency in computation. We leave the choice of set definition as a component of future work.

After defining grid cell sets $\mathcal{D}_i$ and membership function m, we now discuss the generalized form of Hölder's inequality which for one of the grid cell sets $\mathcal{D}_i$ is defined as

$$\int_{\mathcal{D}_i} \prod_{j=1}^{N_d} p_j(\boldsymbol{x}) d\boldsymbol{x} \leq \prod_{j=1}^{N_d} \Big(\int_{\mathcal{D}_i} p_j^2(\boldsymbol{x}) d\boldsymbol{x} \Big)^{\frac{1}{2}}. \tag{3}$$

Notice in (3) that computing the upper-bound requires computing the product after the integrand instead of before, allowing the upper-bound for particle distributions to evaluate to a non-zero value provided at least one particle from each distribution is a member of set $\mathcal{D}_i$. Hölder's inequality for particle distributions and a set $\mathcal{D}_i$ can be written as

$$\int_{\mathcal{D}_i} \prod_{j=1}^{N_d} p_j(\boldsymbol{x}) d\boldsymbol{x} \leq \prod_{j=1}^{N_d} \Big(\sum_{k:\boldsymbol{z}_k^{(j)} \in \mathcal{D}_i} \big[w_k^{(j)}\big]^2 \Big)^{\frac{1}{2}}. \tag{4}$$

Notice that (4) amounts to summing the square of the weights of particles that are members of $\mathcal{D}_i$ which we can perform quickly and efficiently. This upper-bound also approaches the true integrand as Hölder's inequality yields equivalence if all input functions are proportional.[20] As decreasing the volume of $\mathcal{D}_i$ approaches the evaluation of a single point which is proportional to all other points, decreasing the volume of $\mathcal{D}_i$ allows the upper-bound to approach the true integrand.

Remark 3.2 (Tightness of upper-bound). While increasing the number of discretizations through set subdivision yields a tighter upper-bound, the approximation eventually converges to the true product of 0 everywhere as set subdivisions eventually do not contain at least one particle from each input distribution. We are therefore interested in balancing between increasing the resolution of the sets and supporting the resolution with a larger amount of particles in order to ensure we always maintain an upper-bound on the true product and not the product itself.

We also propose the incorporation of a scalar term $\nu > 0$ in the computation of upper bounds for a set $\mathcal{D}_i$ as

$$I_i = \prod_{j=1}^{N_d} \Big(\sum_{k:\boldsymbol{z}_k^{(j)} \in \mathcal{D}_i} \big[w_k^{(j)}\big]^2 + \nu \Big)^{\frac{1}{2}}. \tag{5}$$

which has practical benefits by allowing us to shape the output product PDF and to choose the behavior of the product when the particle densities do not intermesh as we'll demonstrate later.

Letting I_i be the result of the upper-bound evaluation in (5), we then re-weight particles in the first distribution (W_1, Z_1) to be equal to the upper-bound and normalize all particles to have a valid particle distribution. This operation amounts to setting

$$w_i'^{(1)} \propto I_{k:\boldsymbol{z}_i^{(1)} \in \mathcal{D}_k} \tag{6}$$

which requires computing the membership of particle $\boldsymbol{z}_i^{(1)}$. After computing the new particle distribution $\{(w_i'^{(1)}, \boldsymbol{z}_i^{(1)})\}_{i=1}^{N_p}$, we re-sample the distribution to find the final result. The full algorithm is described in Algorithm 3.3.

Algorithm 3.3 (Approximate Pointwise Product).

```
1:  procedure PPRO($\{(W_i, Z_i)\}_{i=1}^{N_d}$, $N_{sets}$, $\nu$)
2:      $\boldsymbol{\mathcal{D}} \leftarrow$ ComputeSets($\{(W_i, Z_i)\}_{i=1}^{N_d}, N_{sets}$)
3:      $\boldsymbol{I} \leftarrow$ Hölder($\boldsymbol{\mathcal{D}}, \{(W_i, Z_i)\}_{i=1}^{N_d}$, $\nu$)
4:      for $\mathcal{D}_j \in \boldsymbol{\mathcal{D}}$ do
5:          $\boldsymbol{m} \leftarrow$ Membership($Z_1$)
6:          $w_{\boldsymbol{m}}'^{(1)} \leftarrow$ Re-Weight($I_j, \boldsymbol{m}$)
7:      end for
8:      $W' \leftarrow$ Normalize($\{w_i'^{(1)}\}_{i=1}^{N_p}$)
9:      $Z' \leftarrow$ Sample($N_p, W', Z_1$)
10:     return $\{(\frac{1}{N_p}, \boldsymbol{z}_i')\}_{i=1}^{N_p}$
11: end procedure
```

This approach allows for the first particle distribution to be updated according to any information provided by other particle distributions, a common scenario in multi-robot teams in which a robot i may be observed by another robot j according to some relative measurement function g. Even if g provides partial information, robot i may update its weights by fusing $\{(W_i, Z_i)\}$ with $\{(W_j, g(Z_j))\}$ where $g(Z_j)$ are the transformed positions of robot j's state.

However, in some scenarios g may provide full information instead. In such cases, the transformed particles $g(Z_j)$ may be useful for updating the particle positions of robot i. We describe this effect as a type of particle nudging in which particles $g(Z_j)$ can be used to "nudge" Z_i.

3.2. *Weighted Particle Nudging*

Particle nudging is a process used in the literature to improve the tracking performance of a particle filter by optimizing the particles to locations of higher likelihood when given a sensor measurement.[21,22] This process has strong benefits in geophysics problems where the state variable has high dimension and can require impossibly large numbers of particles to achieve good performance. In this work, we instead propose to use particle positions provided by other agents to act as nudging particles. In scenarios where a relative measurement function g provides full state information, we can modify our particle product strategy to "nudge" the positions of the product distribution. This can be achieved by re-weighting all provided particles and adapting the re-sampling phase to incorporate the positions of the other distributions $\{(W_i, Z_i)\}_{i=2}^{N_d}$.

As an example, a robot i with position $\boldsymbol{x}_t^{(i)}$ at time t could make a relative position measurement $\boldsymbol{w}_t^{(i,j)} + \boldsymbol{\epsilon}_t^{(i,j)}$ of another robot j where $\boldsymbol{\epsilon}_t^{(i,j)}$ is some corrupting noise drawn from $p_{noise}(\boldsymbol{x})$. The transformed position $\boldsymbol{x}_t^{(i)} + \boldsymbol{w}_t^{(i,j)}$ can then act as a position measurement of robot j's position corrupted by uncertainty in either robot i's position estimate which is represented as a set of particles $\{(w_k^{(i)}, \boldsymbol{z}_k^{(i)})\}_{k=1}^{N_p}$ or in the relative measurement described by p_{noise}.

We also introduce a weight factor $\alpha_1, \alpha_2, \ldots, \alpha_{N_d}$ such that $\sum_{i=1}^{N_d} \alpha_i = 1$ and $\alpha_i \geq 0$ as well to help control how much the nudging particles influence the final product distribution. For a distribution p_i, we incorporate these weights by modifying particle weights as $\{\big((w_k^{(i)})^{\alpha_i}, \boldsymbol{z}_k^{(i)}\big)\}_{k=1}^{N_p}$ which yields the updated product rule

$$I_i = \prod_{j=1}^{N_d} \Big(\sum_{k:\boldsymbol{z}_k^{(j)} \in \mathcal{D}_i} \big[(w_k^{(j)})^{\alpha_j}\big]^2 + \nu \Big)^{\frac{1}{2}}. \tag{7}$$

Notice here that when $\nu = 0$, this amounts to computing an upper bound on the weighted product between a set of N_d distributions which is equivalent to computing the weighted Kullback-Leibler Average[4] (KLA) over a set $\mathcal{D}_i$. In its original work the weighted KLA is shown to be the geometric average between a set of input distributions and was used to facilitate distributed estimation. In particular, the work discussed using conventional consensus update rules but using the weighted KLA as a notion for distribution consensus. The authors then used this strategy to construct a distributed Kalman Filter which could be used to track a moving target with a sensor network.

For a set of distributions $\{p_i(\boldsymbol{x})\}_{i=1}^{N}$, the weighted KLA with weights α_i is defined as

$$\bar{p} = \operatorname*{arg\,inf}_{p \in \mathcal{P}} \sum_{i=1}^{N} \alpha_i D_{KL}(p, p_i) \tag{8}$$

where D_{KL} is the KL divergence between p and p_i and $\mathcal{P}$ is the family of all valid distributions. Solving (8) amounts to computing

$$\bar{p}(\boldsymbol{x}) \propto \prod_{i=1}^{N_d} p_i^{\alpha_i}(\boldsymbol{x}) \tag{9}$$

in which a scaling factor from re-normalizing the distributions is omitted for brevity.[4] To find the weighted KLA for particle distributions, applying (9) is normally impossible due to the need for a well-defined pointwise-product. However, the weighted nudging strategy defined in this subsection allows us to recover the weighted KLA for particle distributions as an additional application of our strategy. We further use this method to design and construct a distributed particle filter which can incorporate relative measurements between robots which we discuss in greater detail later in this work. The weighted nudged product algorithm is shown in Algorithm 3.4.

Algorithm 3.4 (Nudged Pointwise Product).

```
1: procedure NPPRO($\{\alpha_i, (W_i, Z_i)\}_{i=1}^{N_d}$, $N_{sets}$, $\nu$)
2:     $\boldsymbol{\mathcal{D}} \leftarrow$ ComputeSets($\{(W_i, Z_i)\}_{i=1}^{N_d}$, $N_{sets}$)
3:     $\boldsymbol{I} \leftarrow$ Hölder($\boldsymbol{\mathcal{D}}$, $\{\alpha_i, (W_i, Z_i)\}_{i=1}^{N_d}$, $\nu$)
4:     for $\mathcal{D}_j \in \boldsymbol{\mathcal{D}}$ do
5:         $\boldsymbol{m} \leftarrow$ Membership($\{Z_i\}_{i=1}^{N_d}$)
6:         $w'_{\boldsymbol{m}} \leftarrow$ Re-Weight($I_j, \boldsymbol{m}$)
7:     end for
8:     $W' \leftarrow$ Normalize($\{w'_i\}_{i=1}^{N_p \times N_d}$)
9:     $Z' \leftarrow$ Sample($N_p, W', \{Z_i\}_{i=1}^{N_d}$)
10:    return $\{(\frac{1}{N_p}, \boldsymbol{z}'_i)\}_{i=1}^{N_p}$
11: end procedure
```

3.3. *Approximate Product Examples*

To demonstrate our proposed strategy, we first show an example in which particles are sampled from two Gaussian distributions and our product is used to re-sample particles. Fig. 3 shows the product with a non-zero ν providing a biased distribution based on blue's particles on the left and a re-sampling only from blue on the right since there's no interaction between the distributions.

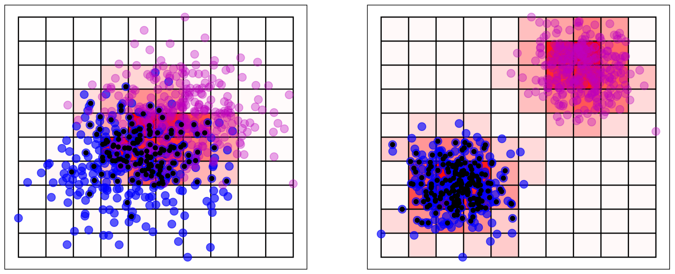


Fig. 3. Gaussian examples of approximate product (highlighted as black particles). Grid cell color indicates upper-bound value. When distributions intermesh (left), product is biased towards the intersecting regions. When distributions are separate (right), product has no effect and maintains the shape of the original distribution.

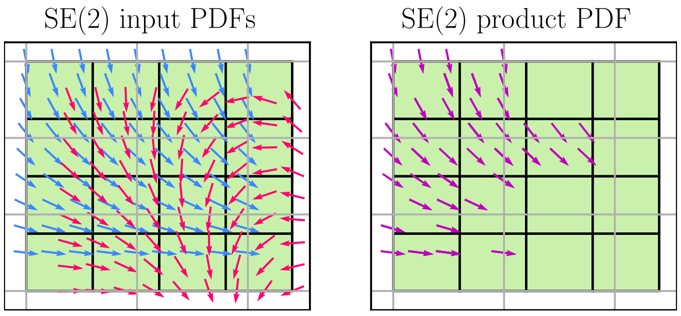


Fig. 4. Examples of the proposed product approximation on $SE(2)$. Input distributions (left) on $SE(2)$ are chosen as two vector fields, only xy sets are shown for convenience. 4 sets along xy dimensions and 5 sets along the heading dimension are used to generate the product approximation (right).

3.4. *Nudged Product Examples*

To demonstrate the benefits of our nudging approach, we use particles sampled from two Gaussian distributions and use our nudged product to re-sample the particles. Fig. 5 shows the effect of the nudged product with a non-zero ν providing a more significant bias between the blue and pink distributions and even achieving a multi-modal shape when distributions do not intermesh. This has the useful quality of retaining non-agreeing information as separate guesses that "nudge" the distribution to another mode.

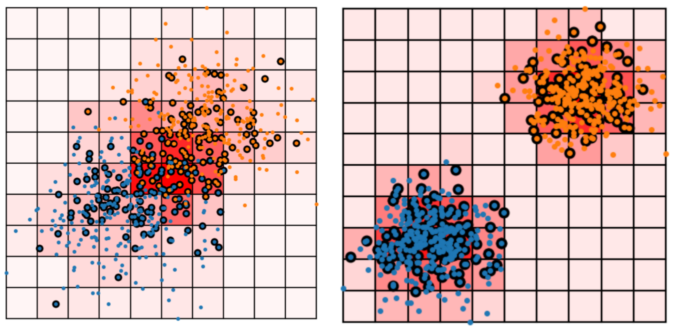


Fig. 5. Gaussian examples of nudged product (highlighted as black particles). Grid cell color indicates upper-bound value. When distributions intermesh (left), product is sampled from both and fuses the distributions. When distributions are separate (right), product subsamples from all distributions and achieves a multimodal shape.

Notice that the results of the nudged product are similar to the regular product when particles perfectly intermesh. The nudged version of our product rule only provides useful information when particle distributions disagree. As previously mentioned, this can be useful in multi-agent settings with relative measurements but can also be detrimental if bad information is provided. For example, in the case where a sensor has been damaged or becomes erroneous, the distribution can be nudged towards worse estimates. In such situations, using the regular proposed product can be used instead to ignore an erroneous information source.

3.5. *Comparison Studies*

We also demonstrate our approach in a comparison study to another product approach based on using local kernels.[13] In the kernel-based strategy, a Gaussian kernel is defined using the k-nearest neighbors of each particle in a single particle distribution in order to compute new weights for each particle in the total distribution. The approach focuses on computing a notion of a product by weighing the effects of nearby particles according to proximity. With an appropriate definition of the proximity, encoded as a kernel, the approach is able to achieve good product results. However, the approach has odd behaviors when particles do not intermesh as to be shown. For two input distributions $\{(w_i^{(1)}, z_i^{(1)})\}_{i=1}^{N_p}$ and $\{(w_i^{(2)}, z_i^{(2)})\}_{i=1}^{N_p}$, the reweighting strategy is given by

$$p_f(x) = c\Big(\sum_{i=1}^{N_p} w_i^{(1)} \hat{f}_2(z_i^{(1)})\delta(x - z_i^{(1)}) + \sum_{k=1}^{N_p} w_k^{(2)} \hat{f}_1(z_k^{(2)})\delta(x - z_i^{(1)})\Big) \tag{10}$$

where c is a normalization factor and $\hat{f}_1$ and $\hat{f}_2$ are local Gaussian kernels re-weighted such that the integrand of the kernel is equal to the total weight of the k-nearest neighbors divided by the volume of a hyper-sphere with radius equal to the distance of the farthest neighbor found in k-nearest neighbors. Fig. 6 shows a comparison of the local kernel strategy to our proposed nudged product strategy when the input distributions (shown as histograms) do not intermesh. When the distributions are not intermeshing, the kernel strategy produces an overly confident distribution centered between the distributions. Our approach, however, can treat this scenario as a case to split the distribution into two separate guesses, and retain information on both until more information is available. Besides the kernel-based strategy,[13] other strategies in the literature use kernel or GMM fitting to find continuous representations of the particle distributions from which a pointwise-product is well-defined.

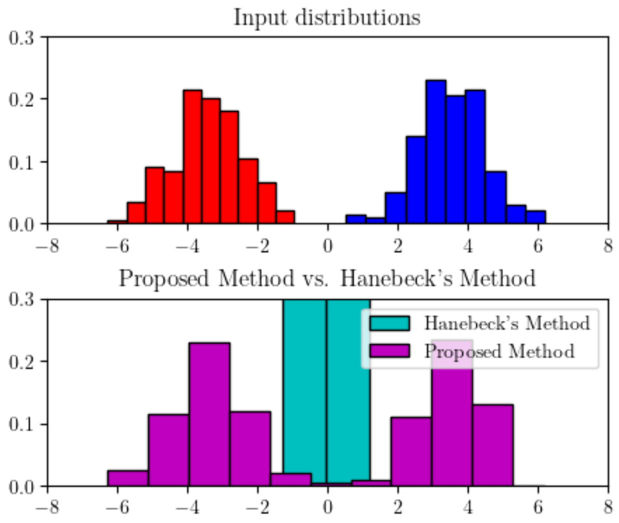


Fig. 6. Comparison to the knn-based fusion method[13] in which local Gaussian kernels are defined to facilitate the pointwise product. Histograms are used to show the shape of the PDF.

Fig. 7 shows the speed of this fitting performance for two particle distributions as the number of particles or the number of dimensions increases. These tests are performed without parallel computing to show exact scaling effects. Our particle-fusion approach scales significantly better than GMM and KDE approaches as the number of particles increases to large numbers. This has inherent benefits in complex multi-modal problems where a large number of particles are needed to represent all modes. Notably, our approach scales poorly as the number of dimensions increases due to the increase in number of sets (in the figure, the total number of sets needed is 5^n where n is the number of dimensions). However, we believe parallel approaches can allow us to scale to larger dimension counts.

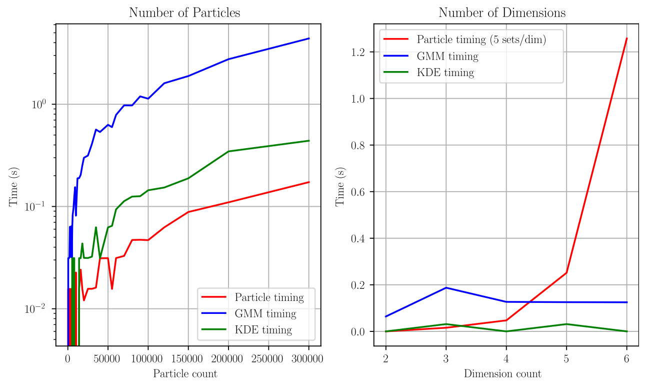


Fig. 7. Computational speed of using direct particle fusion against GMM or KDE fitting. **Left)** time to compute as the number of particles in each input distribution increases. **Right)** time to compute as the number of dimensions increases.

3.6. *Parallel Execution of Product Strategy*

As a final note, we also include details here on the computational performance of the paralellized version of the product strategy, in which we demonstrate the speed improvements associated with using an increasing number of separate processes on an Intel i9-11900H CPU with 8 cores. We split the work for evaluating the product of two 1000 particle distributions over 2500 sets and plot the amount of time it takes to compute the product as we increase the number of workers. Fig. 8 shows the results of splitting the work of an increasing number of sets across an increasing number of processes, with the most significant improvement achieving a five fold speed-up when we split across 7-8 processes (50ms). By increasing the number of cores further, our approach can achieve even larger improvements in computational performance.

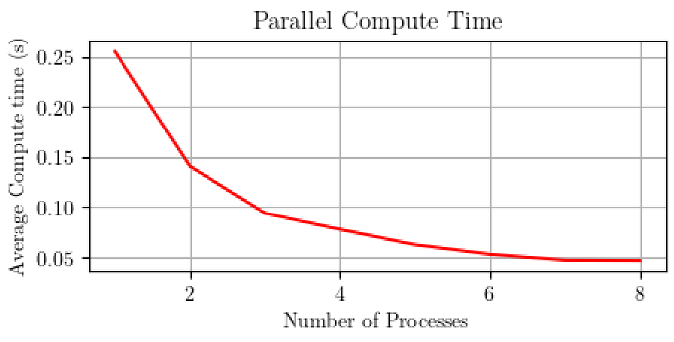


Fig. 8. Effects of splitting the set evaluation work across multiple processes on an 8-core CPU.

4. Distributed Particle Estimation

We discuss now the distributed estimation problems of interest and construct solutions to these problems in the form of distributed particle filtering and distributed particle smoothing. The proposed filtering and smoothing strategies demonstrate the applicability of our proposed product method by realizing existing strategies in the literature that depend on the pointwise product.

4.1. *Distributed Filtering*

We discuss first the distributed filtering problem and describe how our proposed product can be used to solve this problem. For a team of N robots, we model each robot i at time t as a state $\boldsymbol{x}_t^{(i)} \in \mathcal{X}$. Each robot's state evolves forward in time with input $\boldsymbol{u}_t^{(i)} \in \mathcal{U}$ according to

$$\boldsymbol{x}_t^{(i)} = K(\boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{u}_t^{(i)}) + \boldsymbol{s}_t^{(i)} \tag{11}$$

which is corrupted by random noise $\boldsymbol{s}_t^{(i)}$ sampled from known distribution $\boldsymbol{s}_t^{(i)} \sim p_{\boldsymbol{s}_t}^{(i)}(\tilde{\boldsymbol{s}}_t)$.

We assume each robot may make personal measurements on their state of the form

$$\boldsymbol{y}_t^{(i)} = h(\boldsymbol{x}_t^{(i)}) + \boldsymbol{\xi}_t^{(i)} \tag{12}$$

in which $h : \mathcal{X} \to \mathcal{Y}$ is a measurement function corrupted by noise $\boldsymbol{\xi}_t^{(i)} \sim p_{\boldsymbol{y}_t}^{(i)}(\tilde{\boldsymbol{y}}_t)$ drawn from another known distribution describing the corrupting noise of our sensing measurements. We also assume robots are able to make relative measurements with respect to each other and can communicate to help improve each other's estimates.
The relative measurements are described as

$$\boldsymbol{r}_t^{(i,j)} = g(\boldsymbol{x}_t^{(i)}, \boldsymbol{x}_t^{(j)}) + \boldsymbol{\sigma}_t^{(i,j)} \tag{13}$$

in which $g : \mathcal{X} \times \mathcal{X} \to \mathcal{W}$ is a relative measurement function corrupted by noise $\boldsymbol{\sigma}_t^{(i,j)} \sim p_{\boldsymbol{r}_t}^{(i,j)}(\tilde{\boldsymbol{r}}_t)$ drawn from another known distribution describing the corrupting noise of our relative sensing measurements. We use $W_t^{(i)} = \{\boldsymbol{r}_t^{(i,j)}\}_{j \in \mathcal{N}_t^{(i)}}$ to describe the set of observed relative measurements with $\mathcal{N}_t^{(i)}$ containing the indices of observed neighboring robots.

Our principal objective is to identify each robot's marginal PDFs at each time-step using measurements collected by robots individually and relative measurements collected between pairs of robots.

Problem 4.1 (Distributed Filtering). *Assume we have a team of N robots, with each robot i having an initial state estimate $p_{\boldsymbol{x}_0}^{(i)}(\tilde{\boldsymbol{x}}_0)$ and a sequence of T applied inputs $\{\boldsymbol{u}_t^{(i)}\}_{t=1}^T$ perturbed by noise distribution $p_{\boldsymbol{s}_t}^{(i)}(\tilde{\boldsymbol{s}}_t)$. We assume each robot has also collected a history of measurements $\boldsymbol{y}_{0:t}^{(i)}$ and relative measurements $W_{0:t}^{(i)}$. Compute the marginalized PDF $p(\boldsymbol{x}_t^{(i)} \mid \boldsymbol{y}_{0:t}^{(i)}, W_{0:t}^{(i)})$ for each robot i.*

Our approach to solving Problem 4.1 uses the weighted KLA for particle distributions as a fusion operator and modifies the particle filter to compute a weighted KLA between two robots with relative position observations after each iteration of the normal particle filter. The resulting product distribution then acts as the prior distribution for the next particle filter iteration.

We next discuss the distributed smoothing problem which has strong similarities to Problem 4.1 but incorporates information in the future as well.

4.2. *Distributed Smoothing*

Given initial estimates provided by the proposed distributed particle filter, the smoothing problem seeks to use measurements collected in the future to improve earlier estimates along the trajectory. We formulate this problem using dynamics, personal measurements, and relative measurements as in (11), (12), and (13). Our principal objective is once again to identify each robot's marginal PDFs at each time-step but including information provided in the future as well.

Problem 4.2 (Distributed Smoothing). *Assume we have a team of N robots, with each robot i having an initial state estimate $p^{(i)}_{\boldsymbol{x}_0}(\tilde{\boldsymbol{x}}_t)$ and a sequence of T applied inputs $\{\boldsymbol{u}^{(i)}_t\}^T_{t=1}$ perturbed by noise distribution $p^{(i)}_{\boldsymbol{s}_t}(\tilde{\boldsymbol{s}}_t)$. We assume each robot has also collected a history of measurements $\boldsymbol{y}^{(i)}_{0:T}$ and relative measurements $W^{(i)}_{0:T}$. Compute the marginalized PDFs $p(\boldsymbol{x}^{(i)}_t \mid \boldsymbol{y}^{(i)}_{0:T}, W^{(i)}_{0:T})$ for each time instance t and for each robot i.*

The main difference between Problem 4.1 and Problem 4.2 is whether we incorporate future measurements into the marginal inference. We address this problem by formulating a factor graph using (11), (12), and (13) and approximating the marginal PDFs using an algorithm known as Belief Propagation.

4.3. *Factor Graph Formulation*

We formulate Problem 4.2 as an inference problem over a factor graph and design factors based on the collected transformations $\boldsymbol{u}^{(i)}_{1:T}$, personal measurements $\boldsymbol{y}^{(i)}_{0:T}$, and relative measurements $W^{(i)}_{0:T}$. We choose unary factors $\phi^{(i)}_{\boldsymbol{x}_0}(\tilde{\boldsymbol{x}}_0)$ for prior information as

$$\phi^{(i)}_{\boldsymbol{x}_0}(\tilde{\boldsymbol{x}}_0) = p^{(i)}_{\boldsymbol{x}_0}(\tilde{\boldsymbol{x}}_0) \tag{14}$$

and unary factors $\phi^{(i)}_{\boldsymbol{y}_t}(\tilde{\boldsymbol{x}}_t)$ for personal sensor measurements as

$$\phi^{(i)}_{\boldsymbol{y}_t}(\tilde{\boldsymbol{x}}_t) = p^{(i)}_{\boldsymbol{y}_t}(\boldsymbol{y}^{(i)}_t - h(\tilde{\boldsymbol{x}}_t)). \tag{15}$$

We choose binary factors $f^{(i)}_{\boldsymbol{x}_t}(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{x}}_t)$ to encode the local transformations as

$$f^{(i)}_{\boldsymbol{x}_t}(\tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{x}}_t) = p^{(i)}_{\boldsymbol{s}_t}(\tilde{\boldsymbol{x}}_t - K(\tilde{\boldsymbol{x}}_{t-1}, \boldsymbol{u}^{(i)}_t)). \tag{16}$$

We choose binary factors $f^{(i,j)}_{\boldsymbol{r}_t}(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)$ to encode the relative measurements between robots as

$$f^{(i,j)}_{\boldsymbol{r}_t}(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j) = p^{(i,j)}_{\boldsymbol{r}_t}(\boldsymbol{r}^{(i,j)}_t - g(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j)). \tag{17}$$

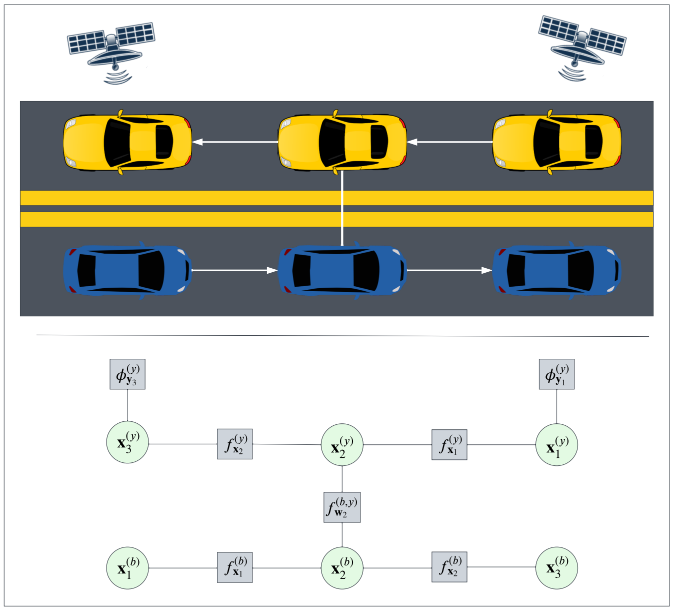


Fig. 9. Example factor graph representation. **Top)** Two vehicles, yellow (y) and blue (b) pass each other on a highway and observe their relative position at the second time-step. Yellow receives GPS data on the first and third time-steps. **Bottom)** The hidden variables (highlighted grey) of each vehicle's position are related by factor nodes encoding the local movements over time $f^{(i)}_{\boldsymbol{x}_t}$, the relative measurement between vehicles at the second time-step $f^{(b,y)}_{\boldsymbol{y}_2}$, and the GPS measurements $\phi^{(y)}_{\boldsymbol{y}_t}$ on yellow.

Defining factors (14), (15), (16), and (17) allows us to describe the hidden variables and their relationships as a factor graph. An example factor graph translated from a distributed autonomous vehicle scenario is shown in Fig. 9 in which two vehicles, yellow and blue, pass each other on the highway. By formulating the factor graph and solving for the marginal PDFs, information collected by yellow via personal GPS measurements can be shared with blue to improve blue's own position estimate, albeit corrupted by yellow's own position uncertainty.

4.4. *Belief Propagation*

Given a factor graph, marginal inference is performed using the Belief Propagation algorithm, a method to efficiently marginalize out variables over every node in the graph by passing messages between nodes. Belief Propagation has a long history in the literature and has been used to efficiently compute marginal PDFs over various types of probabilistic graphs. The approach has been used, more recently, to solve smoothing problem in SLAM settings by using factors to describe the relative measurements created by robots and between robots and other robots or landmarks which are also unknown states to estimate.

Letting $\mathcal{X}_j$ be the set of argument node indices for factor f_j excluding argument i, the Belief Propagation messages from factors to nodes are defined as

$$m_{f_j \to x_i}(x_i) = \int_{\mathcal{X}_j} f_j(\mathcal{X}_j) \prod_{k \in \mathcal{X}_j} m_{x_k \to f_j}(x_j) d\mathcal{X}_j \tag{18}$$

which may be computed using Monte-Carlo sampling[10] after the pointwise product has been computed. Messages from variable nodes x_i to factors f_j are defined as

$$m_{x_i \to f_j}(x_j) = \prod_{k \in \mathcal{N}(x_i) \setminus j} m_{f_k \to x_i}(x_i) \tag{19}$$

in which $\mathcal{N}(\cdot)$ returns the set of indices of all neighboring factor nodes. Finally, the marginal PDF of a variable node, also known as the belief, is computed as

$$b(x_i) = \prod_{j \in \mathcal{N}(x_i)} m_{f_j \to x_i}(x_i) \tag{20}$$

which is simply the pointwise product of all incoming factor messages to the variable node x_i. Unary factors are treated identically to messages (18), i.e., they are incorporated as part of a pointwise product. However, unary factor messages are defined by

$$m_{\phi_j \to x_i}(x_i) = \phi_j(x_i) \tag{21}$$

in which the message is exactly identical to the unary function itself. Commonly, the unary function is a continuous function which does not necessitate the approximate pointwise product strategy proposed here. Instead, we apply the true pointwise product to update the particle weights after applying the approximate pointwise product. On acyclic graphs, the message-passing method defined in Belief Propagation produces exact marginal PDFs using only a single pass through the graph. To compute marginal PDFs on cyclic graphs, Belief Propagation is extended into an iterative form called Loopy Belief Propagation, in which messages are continuously re-computed until convergence at which point the stationary beliefs are used as the marginal PDFs. While loopy Belief Propagation is not guaranteed to converge to the true marginals, the algorithm has been empirically demonstrated to produce good approximations of the marginal PDFs if given good initial belief estimates.[6]

Under certain families of PDFs, such as Gaussians[23, 24] or discrete PMFs,[5] (18), (19), and (20) allow for analytical computation. However, in particle settings these messages cannot be computed analytically, largely in part due to the need for a well-defined pointwise product in the message and belief computations. While other approaches in the literature have circumvented this difficulty either through sampling approaches or via interpolation, these approaches can be slow to execute or require significant fine tuning for good results.[10, 19, 25]

Note here that the defined factors (14), (15), (16), and (17) are actually conditional distributions that arise from the hidden markov model formulation in Problems 4.1 and 4.2. As a result, message computation (18) can be computed using the normal particle filter propagation step.[3] To compute the reverse direction message, we use the backward smoother update to estimate new weights for the previous particles.[3] We next demonstrate a simulation example to show how this algorithm can be used.

4.5. *Simulation Example*

We demonstrate the proposed distributed filtering and smoothing strategies using a four-robot simulation where one of the robots has access to a range-only measurement function but all other robots only have access to relative measurements. In this setting, the personal measurement function in (12) is given by

$$h(\boldsymbol{x}_t^{(1)}) = \left\| \left[x_t^{(1)} \; y_t^{(1)} \right]^\intercal - L \right\|_2 \tag{22}$$

in which L is the 2D position of a known landmark in the environment. All robot inputs are perturbed by zero-mean Gaussian noise with covariance $0.01\boldsymbol{I}_3$ in which the orientation noise is wrapped to $[-\pi, \pi)$. Relative and personal measurements are perturbed by zero-mean Gaussian noise with covariance $0.01\boldsymbol{I}_3$ and variance 0.01, respectively. We run the simulation for 100 iterations and only provide landmark measurements every 3 time-steps. All other robots only have access to relative pose measurements when their distances are within $2.5m$. A visualization of this simulation is available in Fig. 10 which shows that the four robots are only able to make relative measurements with robots following the closest orbits and are able to “share” the uncertainty between robots using KLA updates. This is reflected in both the increasing average error and uncertainty as robots orbit farther away from the landmark.

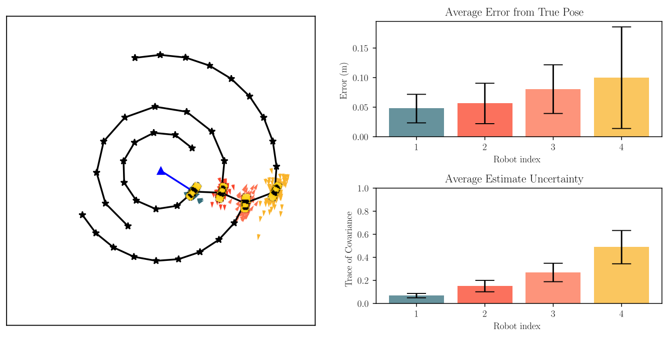


Fig. 10. **Left)** Visualization of the Distributed PF with KLA, black lines show robots making a relative measurement, blue line shows inner-most robot making a personal measurement to the landmark (triangle). **Right)** Average error and uncertainty grows as robots are further from the source of information.

Having collected an initial estimate from the distributed particle filter, we now focus on the smoothing problem in which we incorporate future information to provide estimates over the whole trajectory. We begin by formulating a particle-based factor graph for the four-robot simulation example and use the distributed PF's marginal distributions as our initial beliefs for Belief Propagation. We incorporate three factors: **i)** for odometry related poses, i.e., a robot moving along a trajectory, we use the nudged product to combine messages from previous and future time-steps using a wrapped Gaussian factor, i.e., a Gaussian in SE(2) which wraps the rotation dimension to $[-\pi, \pi)$, **ii)** for personal sensing measurements, we use unary factors to re-weight the particles according to the likelihood associated with detecting the landmark, and **iii)** for relative measurements, we use the nudged product since the relative measurements provides the full state information in the form of a relative pose.

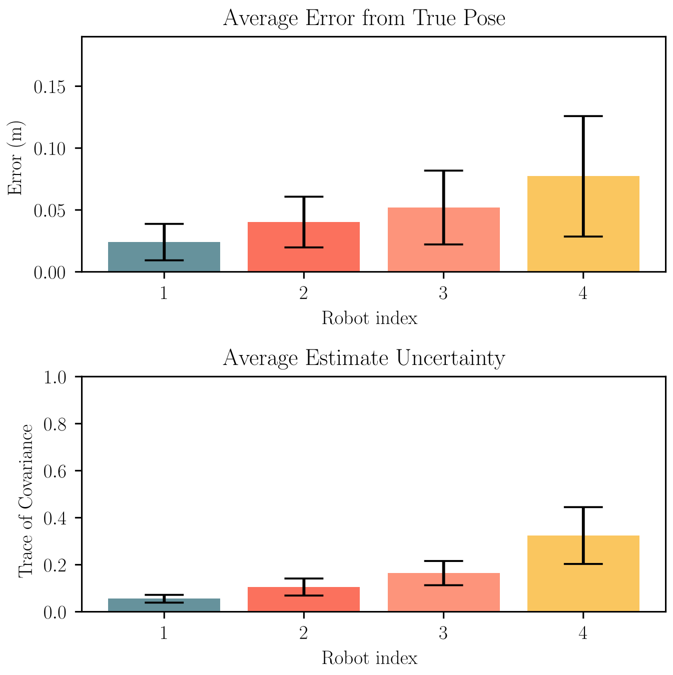


Fig. 11. Average error and uncertainty after smoothing with Belief Propagation. Both error and uncertainty grow the further each robot is from the source of information but improve upon the distributed PF estimates by a maximum of ∼143% in tracking error and ∼152% in uncertainty.

The results of these simulations are shown in Fig. 11 in which the smoothing results for the pair of robots are shown after smoothing with Belief Propagation. We can see that compared to the original average error and uncertainty, the smoothed result achieves both tracking error and uncertainty improvements across all robots with a maximum improvement of ∼143% in tracking error and ∼152% in uncertainty.

5. Experimental Results

Having proposed a distributed filtering and distributed smoothing strategy, we now demonstrate both strategies in an experimental setting using a pair of miniature autonomous blimps. We test the proposed Weighted KLA and Belief Propagation strategies using a pair of experimental miniature autonomous blimps called the Open-Blimp.[26] The Open-Blimp is a lighter-than-air vehicle capable of fully autonomous flight by using 6 rotors arranged in an orthogonal fashion and a helium-filled envelope which has been balanced to equally counter the effects of gravity. We manually pilot two of these platforms in a large open space at the Naval Research Laboratory and allow the platforms to capture the onboard video stream and the applied input data, in the form of recorded accelerations measured by the onboard IMU. We use these images to collect personal and relative position measurements which are available whenever landmarks with known position are observed (realized with Apriltags[27]) and when other blimps are observed (realized using an ellipse tracker of the helium envelope). The onboard IMU is integrated to find relative pose measurements between time-steps.

We adopt state $\boldsymbol{x}_t^{(i)} = (\boldsymbol{p}_t^{(i)}, \theta_t^{(i)})$ which encompasses the 2D position and heading angle of each robot and model the robots using the dynamics (11). We also assume a map of landmarks is available, which in this work is realized as 6 Apriltags with known position. Blimps in these experiments observe these landmarks and other blimps via the forward facing RGB camera using either the Apriltag detector[27] or the previously described ellipse detector. We assume the detection follows the measurement models

$$\begin{aligned} h_j(\boldsymbol{x}_t^{(i)}) &= \boldsymbol{L}_j - \boldsymbol{p}_t^{(i)} \\ g(\boldsymbol{x}_t^{(i)}, \boldsymbol{x}_t^{(j)}) &= \boldsymbol{p}_t^{(i)} - \boldsymbol{p}_t^{(j)} \end{aligned} \tag{23}$$

in which only the position information is used to make measurements. Since the measurements provide only partial information, we use the non-nudged product to update only the weights of the belief distributions. We assume the blimps are perturbed by noise as in 4.1 and 4.2 as well. The input noise function $p_{\boldsymbol{s}_t}^{(i)}$ is split into component noise functions $p_{\boldsymbol{v}_t}^{(i)}$ and $p_{\theta_t}^{(i)}$ which perturb the position and heading of the blimp. $p_{\boldsymbol{v}_t}^{(i)}$, $p_{\theta_t}^{(i)}$, $p_{\boldsymbol{y}_t}^{(i)}$, and $p_{\boldsymbol{r}_t}^{(i)}$ are assumed to be zero-mean Gaussian with variances $1.2\boldsymbol{I}_2$, 0.005, $0.5\boldsymbol{I}_2$, and $0.5\boldsymbol{I}_2$, respectively with heading noises wrapped onto $[-\pi, \pi)$.

5.1. *Experiments with Weighted KLA*

We first apply our distributed particle filter via weighted KLA strategy to our real-life robot experiments. In this set of experiments, we use the collected measurements to compute the regular version of our product in the weighted KLA. This is due to the fact that we are estimating 2D poses but relative information provides only information on relative positions in the local frame of the observing robot. We demonstrate the performance of the proposed method by collecting the 2-norm error between the estimated position of the blimp and the true position of the blimp. Fig. 12 indicates that the average tracking error saw a significant reduction and both robots were able to dramatically improve their performance by using the approximate product to realize the proposed weighted KLA distributed PF.

Fig. 13 shows the first person view of blimp 2 before and after blimp 1 is observed. Blimp 1's PF state is also visualized as a black arrow indicating the mean pose and the black ellipse indicating the covariance of the particle positions. From these snapshots, we can see that blimp 1's initial estimate improves after blimp 2 shares a pose estimate. While the shared information between platforms reduces the tracking error of the blimps, we can reduce the error further by smoothing the trajectory.

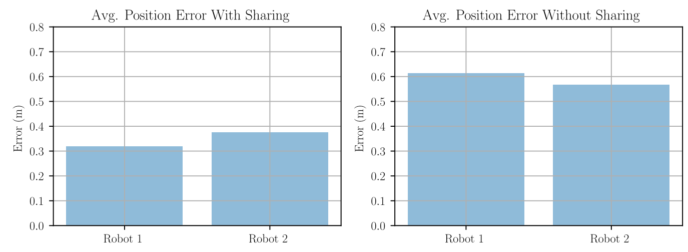


Fig. 12. Average performance of the distributed PF over the course of the experiment for both robots.

5.2. *Experiments with Belief Propagation*

We further improve the estimation quality produced by leveraging Belief Propagation on particle factor graphs to improve the results produced by the proposed distributed PF. We incorporate three factors: **i)** for odometry related poses, i.e., a robot moving along a trajectory, we use the nudged product to combine messages from previous and future time-steps using a wrapped Gaussian factor, i.e., Gaussian noise in SE(2), in which noise on the rotation is wrapped to $[-\pi, \pi)$, **ii)** for personal sensing measurements, we use unary factors to re-weight the particles according to the likelihood of matching the known landmark positions when AprilTags are detected, and **iii)** for relative measurements, we use the regular proposed product to re-weight particles in the relative measurement space, which for this problem is only on position. The factor functions are the input noise functions $p_{\boldsymbol{v}_t}^{(i)}$ and $p_{\theta_t}^{(i)}$ for temporally related poses, the measurement uncertainty $p_{\boldsymbol{y}_t}^{(i)}$, and the relative measurement uncertainty $p_{\boldsymbol{r}_t}^{(i)}$ as stated previously.

In order to refine the initial estimates provided by the distributed PF, we use the marginal PDFs estimated by the distributed PF as our initial belief estimates in Belief Propagation. We see in Fig. 14 that after smoothing, Blimp 1 achieves an average improvement of ∼25.1% and Blimp 2 achieves an average improvement of ∼35.2% over the experiment.

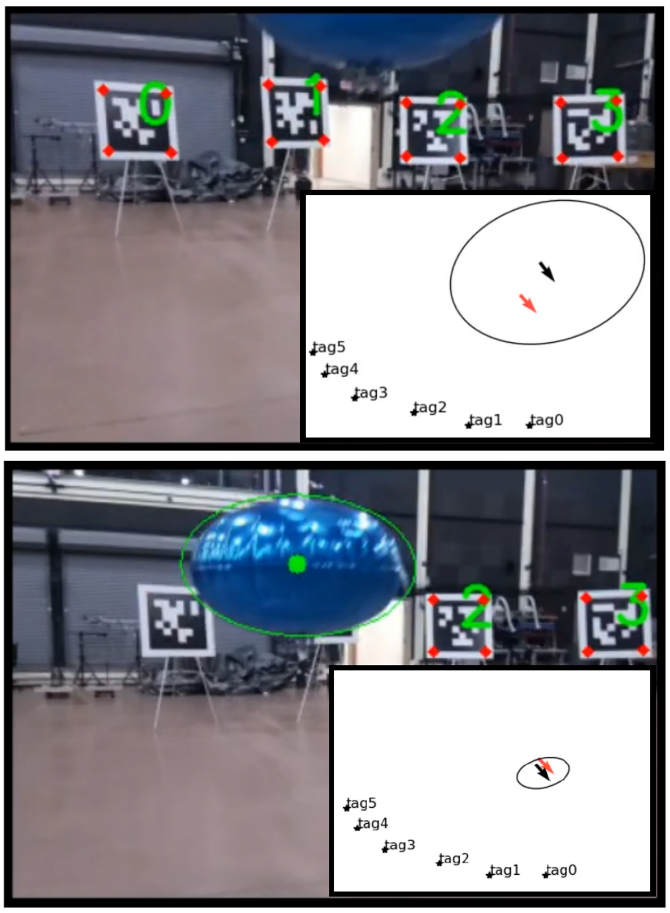


Fig. 13. Pose estimation improvement before and after a relative observation is made available.

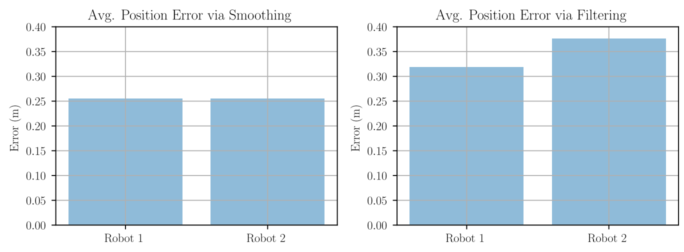


Fig. 14. Average performance of Belief Propagation over the course of the experiment for both robots.

This demonstrates that our proposed product strategy is able to facilitate existing estimation algorithms that require a pointwise product and can be used to produce high accuracy distributed filtering and smoothing.

6. Conclusion

In this work, we proposed a novel product approximation strategy for particle distributions and discussed how this strategy can be used to facilitate distributed filtering, using the weighted Kullback-Leibler Average, and distributed smoothing, using Belief Propagation over Particle Factor Graphs. Our approach is able to achieve accurate results but is limited in the number of dimensions it can handle due to increasing number of sets that must be evaluated. However, through parallel computing, we can achieve significant speed-ups that can help handle higher dimensions. We note that our approach does scale well with number of particles, which may be more useful when trying to model complex multi-modal PDFs.

In future work, we will consider the use of machine learning models to incorporate high-dimensional data into the filtering and smoothing problems. Notably, many robots have access to images or image sequences from which conventional measurement models can not be built. By considering deep neural networks, we can model relationships between images and low-dimensional states for filtering and smoothing frameworks.

Acknowledgments

The research work is supported by ONR grants N00014-19-1-2556 and N00014-19-1-2266; AFOSR grant FA9550-19-1-0283; NSF grants GCR-1934836, CNS-2016582 and ITE-2137798; and NOAA grant NA16NOS0120028; and AFOSR FA9550-22-1-0244.

References

[1] G. Welch, G. Bishop *et al.*, An introduction to the kalman filter (1995).

[2] J. Hartikainen, A. Solin and S. Särkkä, Optimal filtering with kalman filters and smoothers, *Department of biomedica engineering and computational sciences, Aalto University School of Science, 16th August* (2011).

[3] S. Särkkä and L. Svensson, *Bayesian filtering and smoothing* (Cambridge university press, 2023).

[4] G. Battistelli and L. Chisci, Kullback–leibler average, consensus on probability densities, and distributed state estimation with guaranteed stability, *Automatica* **50**(3) (2014) 707–718.

[5] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference* (Morgan kaufmann, 1988).

[6] K. Murphy, Y. Weiss and M. I. Jordan, Loopy belief propagation for approximate inference: An empirical study, *arXiv preprint arXiv:1301.6725* (2013).

[7] T. X. Lin, S. Coogan, D. M. Lofaro, D. A. Sofge and F. Zhang, Monocular vision-based localization and pose estimation with a nudged particle filter and ellipsoidal confidence tubes, *Unmanned Systems* (2022).

[8] A. Torres-González, J. Martinez-de Dios, A. Jiménez-Cano and A. Ollero, An efficient fast-mapping slam method for uas applications using only range measurements, *Unmanned Systems* **4**(02) (2016) 155–165.

[9] A. Rao, W. Han and P. Senarathne, A comparison of slam prediction densities using the kolmogorov smirnov statistic, *Unmanned Systems* **4**(04) (2016) 245–254.

[10] E. B. Sudderth, A. T. Ihler, M. Isard, W. T. Freeman and A. S. Willsky, Nonparametric belief propagation, *Communications of the ACM* **53**(10) (2010) 95–103.

[11] M. Bolic, P. M. Djuric and S. Hong, Resampling algorithms and architectures for distributed particle filters, *IEEE Transactions on Signal Processing* **53**(7) (2005) 2442–2450.

[12] A. Simonetto and T. Keviczky, Recent developments in distributed particle filtering: towards fast and accurate algorithms, *IFAC Proceedings Volumes* **42**(20) (2009) 138–143.

[13] U. D. Hanebeck, Bayesian fusion of empirical distributions based on local density reconstruction, *2015 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE (2015), pp. 277–282.

[14] L.-l. Ong, B. Upcroft, M. Ridley, T. Bailey, S. Sukkarieh and H. Durrant-Whyte, Consistent methods for decentralised data fusion using particle filters, *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, IEEE (2006), pp. 85–91.

[15] D. Gu, J. Sun, Z. Hu and H. Li, Consensus based distributed particle filter in sensor networks, *2008 International Conference on Information and Automation*, IEEE (2008), pp. 302–307.

[16] O. Hlinka, O. Slučiak, F. Hlawatsch, P. M. Djurić and M. Rupp, Likelihood consensus and its application to distributed particle filtering, *IEEE Transactions on Signal Processing* **60**(8) (2012) 4334–4349.

[17] M. Rosencrantz, G. Gordon and S. Thrun, Locating moving entities in indoor environments with teams of mobile robots, *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, (2003), pp. 233–240.

[18] M. Rosencrantz, G. Gordon and S. Thrun, Decentralized sensor fusion with distributed particle filters, *arXiv preprint arXiv:1212.2493* (2012).

[19] A. Ihler and D. McAllester, Particle belief propagation, *Artificial intelligence and statistics*, PMLR (2009), pp. 256–263.

[20] Z. Cvetkovski, *Inequalities: theorems, techniques and selected problems* (Springer Science & Business Media, 2012).

[21] Ö. D. Akyildiz and J. Míguez, Nudging the particle filter, *Statistics and Computing* **30**(2) (2020) 305–330.

[22] H. C. Yeong, R. T. Beeson, N. Namachchivaya and N. Perkowski, Particle filters with nudging in multiscale chaotic systems: with application to the lorenz'96 atmospheric model, *Journal of Nonlinear Science* **30**(4) (2020) 1519–1552.

[23] D. Bickson, Gaussian belief propagation: Theory and aplication, *arXiv preprint arXiv:0811.2518* (2008).

[24] J. Ortiz, T. Evans and A. J. Davison, A visual introduction to gaussian belief propagation, *arXiv preprint arXiv:2107.02308* (2021).

[25] L. Song, A. Gretton, D. Bickson, Y. Low and C. Guestrin, Kernel belief propagation, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings (2011), pp. 707–715.

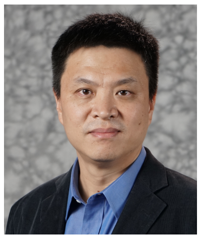
[26] T. X. Lin, T. K. Schuler, D. M. Lofaro, D. Sofge and F. Zhang, The open-blimp: An open-source blimp platform for lighter-than-air research, *AIAA SCITECH 2023 Forum*, (2023), p. 0695.

[27] J. Wang and E. Olson, Apriltag 2: Efficient and robust fiducial detection, *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE (2016), pp. 4193–4198.

Tony X. Lin received a B.S. and M.S. from the University of Virginia, Charlottesville VA, USA, in 2012 and 2016. He is a Ph.D. student in Robotics at the Georgia Institute of Technology in Atlanta GA, USA. Tony's primary research interests involve distributed sensing and estimation for solving localization problems and involve developing coordination strategies for swarms of lighter-than-air vehicles.

Sam Coogan is an associate professor at Georgia Tech in the School of Electrical and Computer Engineering and the School of Civil and Environmental Engineering. Prior to joining Georgia Tech in 2017, he was an assistant professor in the Electrical Engineering Department at UCLA. He received the B.S. degree in Electrical Engineering from Georgia Tech and the M.S. and Ph.D. degrees in Electrical Engineering from the University of California, Berkeley. His research is in the area of dynamical systems and autonomy and focuses on developing scalable tools for verification and control of networked, cyber-physical systems with an emphasis on transportation systems. He received the Donald P Eckman Award from the American Automatic Control Council in 2020, a Young Investigator Award from the Air Force Office of Scientific Research in 2019, a CAREER award from NSF in 2018, and the Outstanding Paper Award for the IEEE Transactions on Control of Network Systems in 2017.

Donald Sofge is a Computer Scientist and Roboticist at the Naval Research Laboratory (NRL) with 35 years of experience (22 at NRL) in Artificial Intelligence, Machine Learning, and Control Systems R&D. He leads the Distributed Autonomous Systems Section in the Navy Center for Applied Research in Artificial Intelligence (NCARAI), where he develops nature-inspired computing paradigms to challenging problems in sensing, artificial intelligence, and control of autonomous robotic systems. He has served as PI/Co-PI on dozens of federally-funded R&D efforts, and has more than 200 refereed publications (including 11 books) in robotics, artificial intelligence, machine learning, planning, sensing, control, and related disciplines, and one patent on virtual state estimation for semiconductor fabrication. His current research focuses on control of autonomous teams or swarms of heterogeneous robotic systems. He has served as an advisor on autonomous systems to DARPA, ONR, OSD, ARL, NSF, and NASA, as well as US representative on international TTCP and NATO technical panels on autonomous systems, and has participated as a member of the following Interagency Working Groups under the National Science and Technology Council (NSTC) Networking and Information Technology Research and Development (NITRD) Program: Intelligent Robotics and Autonomous Systems (IRAS) (formerly Robotics and Intelligent Systems), Machine Learning and Artificial Intelligence (MLAI), and AI R&D Ad Hoc Group. Mr. Sofge is a member of the Maryland Robotics Center Education Advisory Board and also occasionally serves as an Adjunct Faculty Member at the University of Maryland where he has taught graduate-level courses in Robotics. Google Scholar h-index: 24, citations: 3107.



Fumin Zhang is the director of the Hong Kong University of Science and Technology (HKUST) Cheng Kar-Shun Robotics Institute. Prior to joining HKUST, he was a Professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. He received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Maryland, College Park, in 2004. His research interests include marine autonomy, mobile sensor networks, and theoretical foundations for battery supported cyber-physical systems. He received the NSF CAREER Award in 2009, and the ONR YIP Award in 2010