ARTICLE IN PRESS

Ad Hoc Networks xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Real-time collaborative tracking for underwater networked systems

D. Mirza [a,*], P. Naughton [a], C. Schurgers [b], R. Kastner [a]

[a] Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, United States
[b] Qualcomm Institute, Calit2, University of California, San Diego, United States

A R T I C L E I N F O

Article history:
Received 3 March 2014
Received in revised form 12 October 2014
Accepted 13 October 2014
Available online xxxx

Keywords:
Underwater networks
Real-time tracking
Acoustic networks

A B S T R A C T

Localization is a crucial requirement for mobile underwater systems. Real-time position information is needed for control and navigation of underwater vehicles, in early warning systems and for certain routing protocols. Past research has shown that the localization accuracy of networked underwater systems can be significantly improved using inter-vehicle collaboration. More specifically the Maximum Likelihood (ML) position estimates of a mobile collective can be computed from measurements of relative positions and motion, albeit in a non-real-time fashion. In this work we extend this solution to compute the position estimates of a network in real-time and in a distributed fashion. We first describe a centralized approach to identify key factors that fundamentally limit the performance of a real-time solution. Using the centralized approach as a benchmark, we arrive at a real-time distributed solution that additionally computes the location of vehicles using information obtained locally by them. We address practical considerations in the implementation of our algorithm and propose solutions to mitigate computational errors. With this proposed implementation, we provide insight on how to appropriately plan a deployment of nodes when collaborative tracking is to be utilized. Lastly, we shed light on situations where implementing collaborative tracking can hinder the localization performance of the network so that these can be avoided.

1. Introduction

There has been a growing interest in operating groups of autonomous underwater vehicles in a networked fashion, using short to medium range acoustic modems [1]. Location information is critical to such networked mobile underwater systems for correctly annotating data samples, control and navigation and in certain data routing protocols. However, since GPS is not available underwater, the position of vehicles has to be estimated over time.

Most existing solutions based on Kalman or particle filters are designed for tracking vehicles individually [2,3]. This is achieved by combining measurements of relative position with respect to known landmarks or beacons together with estimates of the vehicle's motion obtained from on board sensors. These *non-collaborative* tracking methods perform poorly in underwater networks where vehicles have short to medium acoustic communication range. This is due to the fact that a vehicle may not be within the communication range of a sufficient number of beacons or it may move out of range of beacons from time to time. A solution to this problem involves using additional measurements for localization, which can be obtained from communication between vehicles. Such a *collaborative* approach has been employed by a number

* Corresponding author.
E-mail addresses: dimirza@eng.ucsd.edu (D. Mirza), pnaughto@eng.ucsd.edu (P. Naughton), cschurgers@ucsd.edu (C. Schurgers), kastner@cs.ucsd.edu (R. Kastner).

http://dx.doi.org/10.1016/j.adhoc.2014.10.008
1570-8705/© 2014 Elsevier B.V. All rights reserved.

of localization techniques such as multi-dimensional scaling and iterative multi-lateration. However, these techniques have been proposed for stationary networks. They typically use relative position information (distance, proximity or direction) as geometric constraints on the unknown location of devices. Since devices are stationary, the constraints can be considered concurrently to solve for the unknown locations. In principle these algorithms can be extended to mobile underwater networks by localizing time snapshots of the network as suggested in [4–6]. However this requires measurements of relative position to be obtained concurrently. Further, position estimates for each snapshot must be computed before there is significant displacement in the vehicle positions. In reality, due to practical communication constraints, the use of such techniques for underwater networks would be severely limited when deployments are sparse and vehicles are moving.

Our prior work has shown that in sparse mobile underwater networks, localization accuracy can be significantly improved when vehicles collaborate [7], compared to both individual tracking and collaborative static localization. We proposed a collaborative solution that tracks a group of vehicles as a collective by jointly computing the trajectories of all vehicles using non-concurrent estimates of inter-node distances together with measurements of motion from on-board navigational sensors. While there exist a few collaborative tracking methods that also loosely follow the same idea, our solution is optimal as it jointly computes the Maximum Likelihood (ML) positions of all vehicles over time.

The drawback of this optimal ML solution is that it is centralized and can only be applied offline, after all measurements are available at a single location. It serves applications where position information is only needed post-facto, but cannot solve the problem of real-time localization. Creating a real-time tracking solution for mobile underwater networks is a difficult task due to the power required for inter-node communication and the band limited acoustic channel of the ocean. Despite these challenges, our work in [8] modified our existing ML solution to compute positions in real-time. In this work we first propose a centralized real-time algorithm which computes all position estimates at a central location albeit in real-time. We use this solution to highlight the inherent loss in localization performance when we go from a non-real-time to a real-time solution. More practically, we propose a localized distributed solution for our real-time tracking that minimizes the communication overhead by strategically sharing information between vehicles, yet achieves localization accuracies close to the best case. The essential problem that we address to this end is to determine what information to share between vehicles and when, as well as how to encode information to minimize the communication overhead.

While the work of [8,7] only considered one set of trajectories in their analysis, in this work we provide additional analysis of these techniques by applying them to different motion models. We find that there are cases where the collaborative framework for localization does not perform best, but often it can alleviate many different sources of error in an underwater network and provide results that are superior to the non-collaborative case. Ultimately, we propose that careful consideration of the motion models of the vehicles as well as the positioning of the beacons must be taken into account when planning a deployment of vehicles who are collaborating to determine their position. We provide insight on how these parameters affect the location accuracy of each vehicle so that an informed decision can be made on whether or not to collaboratively localize as well as how to set up a network to reduce localization error when collaborating. Before we can delve into this analysis, we will describe our prior work on non-real-time centralized tracking as well as centralized and distributed real-time tracking. This knowledge forms the basis of later analysis on deployment considerations.

2. Related work

A detailed discussion of how our factor-graph framework relates to other estimation techniques that are traditionally used for underwater navigation, in robotics as well in terrestrial sensor networks can be found in our prior work [7]. A number of real-time and distributed localization techniques have been proposed for underwater networks [4,9,5,6]. Among these DNRL [4] and LSL [6] use multi-lateration while USP [5] applies iterative bilateration. To improve the localization accuracy, DNRL proposes the use of mobile beacons. In LSL a hierarchical (infrastructured) approach is proposed for localization [6]. These techniques essentially estimate positions for time snapshots of the network using only spatial constraints (estimates of inter-vehicle distances) and do not account for vehicle motion. The main drawback compared to our solution is that they would not work well in sparse networks and need more frequent signaling. A more in depth survey is available in [10]. A comparative performance evaluation of some of these schemes has been done in [11].

AUV navigation using a single mobile beacon is proposed in [2,3]. Here measurements of vehicle motion are combined with non-concurrent distance estimates to the beacon. In addition path planning is done to improve the tracking performance [3]. A collaborative solution has been proposed for under-ice AUV tracking where inter-vehicle estimates of distance and relative velocities (from measurements of Doppler shifts) are used for localization [12]. Vehicle positions are estimated using a least squares approach. To minimize the position uncertainty, a subset of neighbor vehicles are selected as position references via an exhaustive search method.

3. Non-real-time centralized solution

Our centralized ML solution estimates the position of vehicles by simultaneously capturing a number of spatial and temporal constraints. These constraints are the result of the measurements that the vehicles collect to help them determine their positions. A first set of measurements, yielding the 'spatial' constraints, are inter-vehicle distance estimates with neighbors within communication range, collected at different times by sending an acoustic ping and measuring the time-of-flight, as described by Eq. (1). The so-called 'temporal' constraints are estimates of vehicle velocity measurements as gathered by the navigational sensors on-board each vehicle. The temporal constraints are mathematically shown in Eq. (2). Calculating the collection of vehicle positions over time within the set of spatial and temporal constraints is a complex multi-dimensional estimation problem. To solve it optimally, we have proposed the framework of factor-graphs. Our factor-graph solution efficiently computes the individual pdfs of the position of vehicles at discrete times from the joint pdf of all the unknown positions given measurements (of inter-vehicle distances and motion). The factor-graph itself is a graphical representation of this joint pdf. A detailed discussion of the complete solution and how these constraints are represented is available in our prior work [7]. The theoretical foundations of this technique have been laid out by Kschischang et. al.[13]. Here we only highlight the key ideas behind our centralized and offline algorithm. To do this we use a simplified notation and describe the factor-graph in terms of a *dependency graph*. The complete factor-graph can be reconstructed from the dependency-graph by replacing each vertex in the dependency graph by its internal representation as discussed in Section 3.1.

Fig. 1(a) shows the dependency graph for a network of four vehicles. We have shown the graph for only four vehicles to not overload the figure. The structure of the graph can be viewed in terms of four chains stacked vertically. Each chain captures the evolution of the unknown position of a vehicle over time. Correspondingly, each vertex in a chain (shown as a hexagon) represents the unknown states (position and velocity) of the vehicle at a particular time instance. A link between any two vertices exists if there is a measurement (or a set of measurements) that relates/constrains those states. Vertices within a chain are linked by measurements of motion, i.e. the temporal constraints. Vertices across chains are linked by measurements of inter-node distance, i.e. the spatial constraints. The triangles represent the position of beacon nodes at any particular time instance. A link with a triangular vertex exists if a measurement of distance was made with the beacon at that time. Note that measurements (of distance and motion) are obtained in real-time and stored on the memory banks of the vehicles. However, the dependency-graph described above is constructed in non-real-time, after all the measurements are available at a central location.

The links in the dependency graph shown in Fig. 1(a) establish pathways of information flow across vehicles in the network as well as back and forth along the temporal dimension of each chain in the graph. Initially, position information is only available at the triangular beacon-vertices (we assume beacons are, for example, buoys on the surface equipped with a GPS receiver). The sum-product algorithm that runs on the above described graph generates this information. When we run the sum-product algorithm on the graph, information from the beacon vertices eventually percolates to all the unknown states (vertices) of the factor-graph along the graph edges as depicted in Fig. 1(a).

The dependency graph shown above is a simplified notation used for the actual factor-graph. Next, we describe how the factor-graph is obtained from the dependency

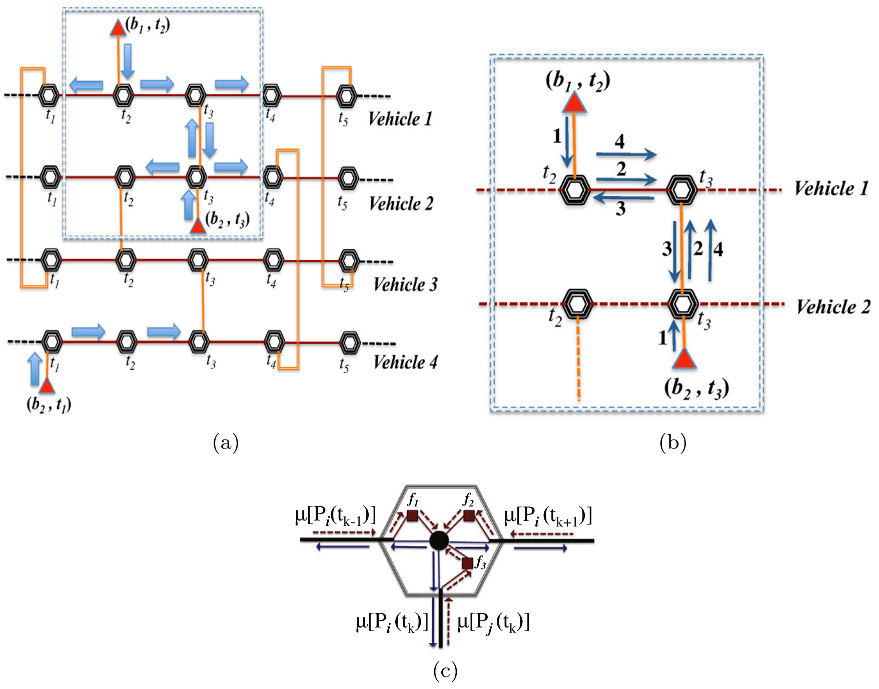

Fig. 1. Centralized non-real-time solution. (a) Dependency graph. (b) Iterative message passing for position estimation. (c) Internal structure of a vertex in the dependency graph for any vehicle i at time t_k.

graph and present the operation of the sum-product algorithm to estimate vehicle positions.

3.1. Constructing the factor-graph from the dependency graph

The unknown positions are computed by running the sum-product (SP) algorithm (in non-real-time) on the factor graph. The factor-graph representation is obtained from the dependency graph shown in Fig. 1(a) by replacing each hexagonal vertex by its corresponding internal structure shown in Fig. 1(c).

To describe the internal structure of each vertex we denote the unknown position of any vehicle i at time t_k as $P_i(t_k)$. A vertex in the dependency graph is a simplified notation for a state-variable, $P_i(t_k)$ and a set of transformation functions (f_1,$f_2$and f_3), as shown in Fig. 1(c). By replacing each vertex in the dependency graph with these elements, the equivalent factor graph is obtained.

The transformation functions f_1,$f_2$and f_3 are computed from the statistical characterization of distance and velocity measurements. We derive these functions from equations relating the unknown positions of vehicles to measurements of distance and velocity. A distance measurement obtained at time t_k as a result of communication between two vehicles with positions $P_i(t_k)$ and $P_j(t_k)$ at that time relates the unknown positions as follows:

$$z_d(t_k) = \|P_i(t_k) - P_j(t_k)\|_2 + \varepsilon_k^R \tag{1}$$

where $z_d(t_k)$ is the distance measurement, ε is the error in distance estimates with any known pdf.

Next, a velocity measurement, $z_v(t_k)$ is related to the unknown position from first principles:

$$z_v(t_k) = \frac{1}{|t_{k+1} - t_k|}(P(t_{k+1}) - P(t_k)) + \varepsilon_k^V \tag{2}$$

where $\varepsilon_k^V = [\varepsilon_k^{v_x}, \varepsilon_k^{v_y}]^T$ is the error in the velocity measurement with known statistics.

The transformation functions f_1,$f_2$and f_3 are defined as follows:

$$f_1 = p(P_i(t_k)|P_i(t_{k-1}), z_v(t_k)) \tag{3}$$

$$f_2 = p(P_i(t_k)|P_i(t_{k+1}), z_v(t_k)) \tag{4}$$

$$f_3 = p(P_i(t_k)|P_j(t_k), z_d(t_k)) \tag{5}$$

The exact definitions of the above conditional distributions are obtained from Eqs. (1) and (2) and a statistical characterization of the error terms in these equations. We have characterized the errors in our simulations to be uniformly distributed, however any other distributions can be used to define the above conditionals. For completeness we describe the error in distance estimates are Gaussian: $\varepsilon_k^R \sim U(-\sigma_{Rmax}, \sigma_{Rmax})$. Further, the error in velocity measurements are also Gaussian $\varepsilon_j^{v_x} \sim U(-\sigma_{Vmax}, \sigma_{Vmax})$, $\varepsilon_j^{v_y} \sim U(-\sigma_{Vmax}, \sigma_{Vmax})$.

In Fig. 1(c), the incoming and outgoing messages are shown by dotted and solid arrows respectively. When the sum-product algorithm runs on the FG constructed using the above notation, the functions, f_1,$f_2$and f_3, operate on the incoming messages $\mu[P_i(t_{k-1})]$, $\mu[P_i(t_{k+1})]$ and $\mu[P_j(t_k)]$, respectively, to compute independent estimates of the pdf of $P_i(t_k)$, as per Eq. (6). The outgoing message $\mu[P_i(t_k)]$ is computed as per Eq. (7).

3.2. Operation of the sum-product algorithm

The sum-product algorithm is essentially an *iterative* message passing algorithm. At each iteration, every vertex in the graph estimates the pdf of its own states based on incoming messages. (A vertex encapsulates the unknown states and a set of transformation functions that operate on incoming messages to compute these estimates as described in Section 3.1). The pdfs computed for each vertex are then passed on as messages to all neighbor vertices.

At each time instance the SP algorithm does two things. First, the state-variables are estimated by intersecting the individual estimates from its function node neighbors, Eq. (7). Next the SP estimates the function nodes by marginalizing the local likelihood function of each state-variable, Eq. (6). This is repeated until convergence. Note that there are two very distinct notions of time here: the time represented by states in the graph (which is the time evolution of the network we are trying to estimate) and the time post-facto during which we run our algorithm on a central location (the running time).

$$\mu_{f-x}(x) = \sum_{\sim\{x\}} \left(f(X) \prod_{y \in (f) \setminus \{x\}} \mu_{y-f}(y) \right) \tag{6}$$

where the summary operation is defined as:

$$\sum_{\sim x_1} f(x_1, x_2, x_3) = \sum_{x_2 \in A_2} \sum_{x_3 \in A_3} f(x_1, x_2, x_3)$$

$$\mu_{x-f}(x) = \prod_{h \in n(x) \setminus f} \mu_{h-x}(x) \tag{7}$$

where X is the set of arguments of function node $f; n(w) \setminus \{x\}$ denotes the set of neighbors of a given node w on the graph excluding the node x.

The key point in the above described factor-graph solution is that to effectively fuse position information, messages have to be passed a number of times across each of the links of the graph as depicted for a small section of the graph in Fig. 1(b). The solution above runs offline and centrally after the actual sensing mission is over. Next, we will explore if and how we can modify this solution to tackle the problem of real-time tracking in underwater networks.

4. Real-time tracking

4.1. Centralized real-time tracking

In this first subsection, we try to answer the question:*what is the fundamental difference between a non-real-time and a real-time estimation of vehicle positions?* Here we still assume that computation can run in a central location. While in our offline non-real-time solution (discussed in Section 3) the factor-graph was constructed in one shot using all measurements made when vehicles were underwater, here we propose a dynamic factor-graph solution that evolves in time to (a) incorporate new states (position and velocity), (b) incorporate estimates of motion and distance with neighbor vehicles as they become available, and (c) limits the use of future information in estimating positions.

We are interested in knowing the positions of vehicles at discrete times δ apart. At any time t, the FG consists of N chains of vertices representing the unknown states of all vehicles until that time. A new vertex is added to the tail of each chain every δ(s) as depicted in Fig. 2. In addition, when a measurement of distance becomes available between two vehicles i and j, due to a communication event at time t, a new vertex is inserted to the i^{th} and j^{th} chain in the graph corresponding to time t and a link is created between them. Each new state is also appropriately linked to previous states in a chain using measurements of motion. Whenever the FG is updated with a new state or measurement, the SP algorithm is run on the graph to compute updated estimates of all states. To obtain position estimates in real-time, the pdf of any state in the FG corresponding to time t is recorded at that time, before future measurements become available. In principle we can estimate the vehicle positions semi real-time, by limiting the extent of future information to a time window τ. Essentially, estimates corresponding to any time t are obtained from the FG at $t+\tau$. This is generally referred to as *smoothing*. Such a solution would be suitable for applications that can tolerate some delay in obtaining position estimates.

We implemented our centralized real-time tracking solution in the NS2 simulator [14]. The simulation set up is described later in Section 4.3 with parameters given in Table 1. In this section we evaluate the localization performance as a function of the smoothing delay, τ. Fig. 3 shows the cdf of the RMS error in position estimates for different values of τ. In the figure, the performance of the offline solution corresponds to $\tau=\infty$ and that of the real-time tracking corresponds to $\tau=0$. Notice the loss in performance as we go from a non-real-time solution to a real-time one. This is due to the fact that the real-time solution does not use any future measurements for estimating the unknown states. However, as time elapses information flows backward (along the horizontal links of the graph) to refine past position estimates. Fig. 3 also shows the gain in localization performance as the smoothing parameter, τ is increased. We observe that the performance of the semi-real-time tracking coincides with that of the non-real-time solution for $\tau=75$ min which is a fraction of the total simulation time of 8 h.

While the simulation results suggest that we can get better localization accuracy by using estimates slightly in the past, in reality we would still need to take into account the fact that the vehicle has moved during the interval τ. Therefore, past (refined) estimates have to be interpolated to the present time. However, this is what our tracking algorithm does in an optimal way using navigational measurements. Therefore, applications that are interested in knowing where a vehicle is ‘now’ in order to make location-based decisions cannot leverage from the parameter τ. This is true for most applications such as navigation and control, routing and so on. However, refining past estimates would be useful for annotating data samples with position information and batch processing them in semi-real time. The smoothing delay is one of the parameters that inherently limits the performance of a real-time and distributed solution. In Section 4.2 we will discuss other such factors.

As shown in Fig. 3, the improvement in the accuracy as a function of the smoothing delay diminishes for large

Table 1
Simulation parameters.

Tracking granularity, δ	2 min
Localization period, T_{LOC}	8 min
Simulation duration, T_{sim}	8 h
Number of vehicles, N_U	8
Number of beacons, N_B	1
Area of deployment, A	2 km × 2 km
Maximum depth of deployment, D	100 m
Maximum transmission range, R	500 m
Average speed of vehicles, $\overline{v}$	0.5 m/s
Bytes used to represent pdfs internally, $N_{internal}$	512
Bytes used to communicate pdfs, N_{comm}	18

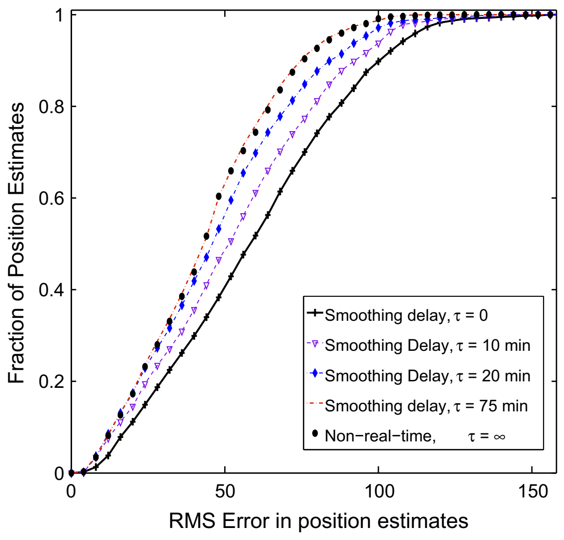

Fig. 3. Cumulative Distribution Function (CDF) of the RMS error as a function of the smoothing parameter, τ. Each curve in the plot shows the fraction of position estimates (indicated by the y-axis), computed for all vehicles for the entire simulation duration, with RMS error less than a given threshold (indicated by the x-axis).

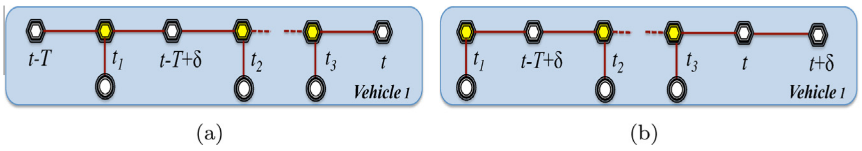

Fig. 2. Evolution of dynamic FG. (a) FG at time t. (b) FG at time $t+\delta$.

enough delays. This is because current updates do not affect past estimates if we go far back enough in time. We therefore limit the number of past states in the dynamic FG to be within a time window T. As new states are added to the FG, states older than $t - T$ are removed from it to limit the computation.

4.2. Distributed real-time tracking

In this section we address the problem of performing the tracking in a distributed fashion, i.e. *How can we split the computation on different vehicles in order to jointly estimate the vehicle positions?*

Our factor-graph solution actually lends itself quite naturally to a distributed solution. For a network with N vehicles, we can view the centralized graph (shown in Fig. 1(a) for $N = 4$) as N *communicating* subgraphs. This decomposed view of the centralized factor-graph is presented in Fig. 4. Since each vehicle is primarily interested in its own position, it would naturally implement the subgraph corresponding to its own states as highlighted in the figure. A vehicle also maintains copies of the position states of other vehicles corresponding to times when a distance measurement was obtained with them as depicted in Fig. 2. If at a later time, more accurate estimates of these states are communicated to the vehicle, it incorporates this information into its FG.

A direct consequence of implementing the subgraphs on different vehicles is that the message-passing algorithm has to now operate over the actual underwater acoustic network and information needs to be transferred between vehicles. This poses a number of challenges on the operation of the sum-product algorithm compared to the centralized case. To achieve the best case performance in a distributed setting, a vehicle has to share updates about all its states (past and current) with those vehicles that had obtained a distance measurement to it. This is equivalent to passing messages over all links of the centralized FG. However, in a distributed setting, messages can only be passed over the active communication links of the physical acoustic network. The problem is that since vehicles are moving, past states cannot be shared with those vehicles that have gone out of communication range.[1] We have depicted the problem that arises in the distributed case in Fig. 5(a). The figure shows the network topology for three vehicles A, B and C at times t_1 and t_2. At time t_1, B communicates with A and shares its current position state, $P_B(t_1)$. Vehicle A also estimates its distance to B at the same time and incorporates this information in its factor-graph as shown in Fig. 5(c). Later at time t_2, B gets new position information from vehicle C, which improves its past estimate at t_1. Now in a centralized solution, this updated information would have been passed to the subgraph of vehicle A, over the link created at t_1 as shown in Fig. 5(b). However, in the distributed scenario this is not possible because A and B are no longer in communication range.

To achieve the best localization performance in the distributed case, we propose that vehicles share their states with those that are multiple hops away. To do so they periodically broadcast their current and past position states as well as forward the states of other vehicles. This addresses the problem described in Fig. 5 because when B improves its past estimate, $P_B(t_1)$ at a later time t_2, it routes this improved estimate to A. The limitation of this scheme compared to the centralized solution is that updates cannot be shared with vehicles that are not reachable via multihop. Nevertheless it gives the best possible real-time distributed localization performance (by construction) and serves as a performance baseline. In the remainder of the paper we will propose ways to minimize the energy overhead of the distributed tracking.

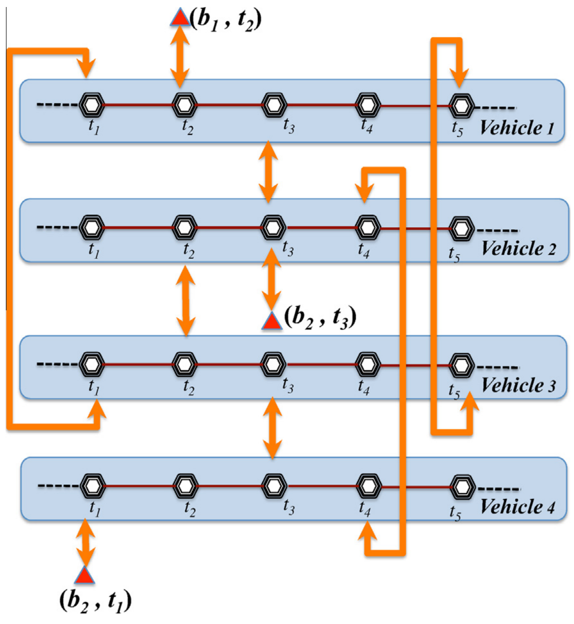

Fig. 4. Decomposed view of centralized dependency graph.

4.3. Distributed tracking with local information sharing

In the previous section, we presented the best distributed real-time solution. However, since underwater acoustic communication is resource intensive and the vehicles are energy-constrained, we now consider ways to make our distributed solution more practical. To limit the communication, we propose schemes that only use local information obtained from communication with immediate neighbors. In the first scheme, each vehicle periodically shares its current and all past position states with its immediate neighbors. In the second scheme we further reduce the amount of information transmitted by vehicles by having them periodically communicate only their current position state to their immediate neighbors. We next compare the performance of the tracking solutions presented so far using the simulation setup described in [8].

[1] Note that a similar problem exists in the centralized formulation because all measurements have to be routed to a single location, while there may be times when no route is available. However, in our simulations we assume that measurements are always available at a single location for centralized tracking. This was done to carry out a logical study of the factors that affect the performance of the distributed solution.

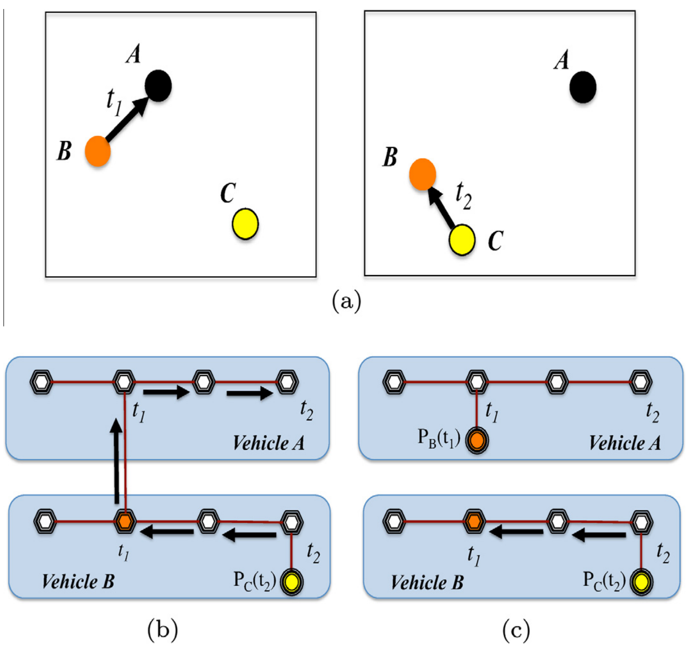

Fig. 5. (a) Network topology at times t_1 and t_2. FG at time t_2. (b) Centralized tracking. (c) Distributed tracking.

Simulation setup: We implemented all the elements of our real-time, centralized and distributed FG tracking in NS2 using the NSMiracle extensions for simulating underwater network protocols [14]. We simulated a network with 8 mobile vehicles and 1 beacon. The vehicles were placed in a 2 km $\times$ 2 km area. The beacons resided on the surface, while the vehicles were at different depths. We used the mobility model proposed in [7] where vehicles follow smooth curves. Vehicles transmitted localization updates every $T_{LOC}(s)$. While our proposed information sharing schemes can be used with any MAC protocol, we chose a TDMA MAC. The simulation parameters are summarized in Table 1.

Using the above described simulation set up we compared the performance of our offline solution (see Section 3), real-time centralized (see Section 4.1), real-time distributed (best case) (see Section 4.2) and the real-time distributed tracking solutions with local information sharing (described earlier in this section). The CDF of RMS error in position estimates for the different tracking solutions is shown in Fig. 6. As the results show the maximum loss in performance occurs when we go from an offline solution to a real-time one.

While there is some benefit in forwarding updates over multiple hops, the energy overhead of doing so is substantial. However, we can come very close to the best case distributed performance by sharing past estimates with immediate neighbors. For these simulations, the results suggest that if performance is the key requirement with nominal constraints on energy a good compromise solution may be to share current and past states with immediate neighbors. However, if energy is of primary concern, then periodically sharing only the current state with immediate neighbors is sufficient.

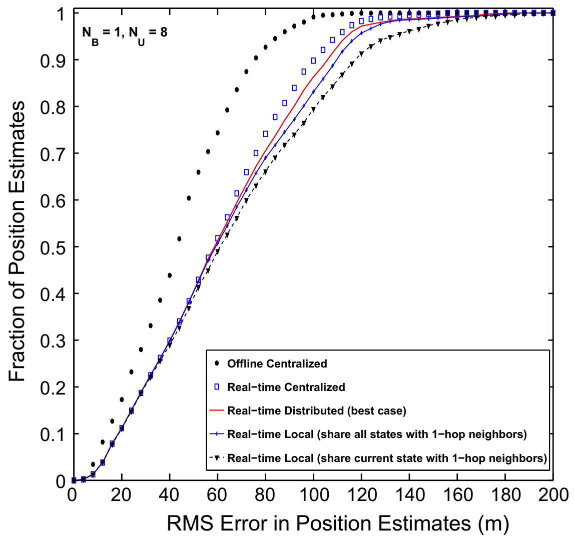

Fig. 6. CDF of RMS error for proposed information schemes.

To get a better understanding of the relative benefit of the different schemes, we evaluated their performance when the number of beacons, N_B and number of vehicles N_U were varied. Later in Section 6, we provide a more in-depth discussion of how this affects the localization performance.

Fig. 7 shows the CDF of the RMS error in position estimates for the different tracking solutions as well for the non-collaborative case where vehicles only use measurements with beacons for position estimation. These results show that there is significant gain in localization accuracy from inter-vehicle collaboration. Further, this gain increases with the number of vehicles because of

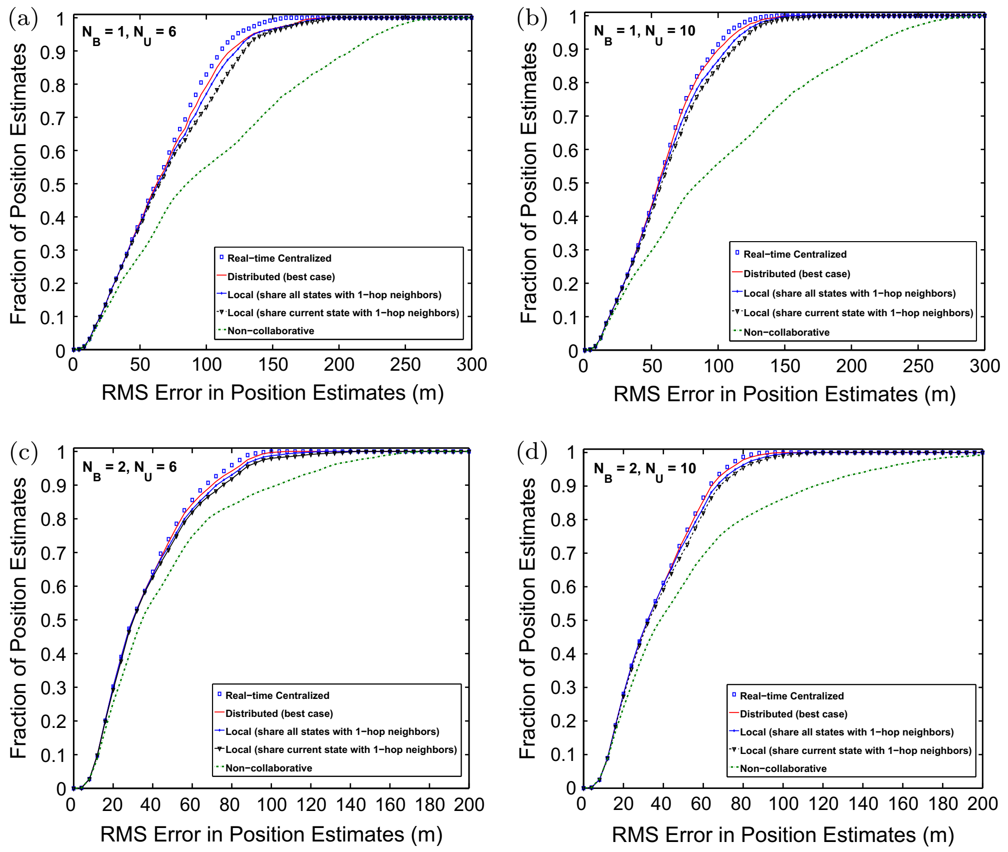

Fig. 7. Performance of real-time centralized, real-time distributed (best case) and real-time distributed tracking solutions with local information sharing for networs with (a) one beacon and 6 unknown nodes, (b) one beacon and 10 unknown nodes, (c) two beacons and 6 unknown nodes, (d) two beacons and 10 unknown nodes.

the additional constraints obtained from inter-vehicle distance estimates. Among the collaborative schemes, sharing past (refined) position states with other vehicles improves localization performance when there is only one beacon in the network. For the networks with one beacon, when the network connectivity is very low ($N_U = 6$), sharing past states with only immediate neighbors almost achieves the best case performance. However, when the connectivity improves ($N_U = 10$), past states have to be shared over multiple hops for the best case performance. This is because under conditions of low network connectivity vehicles that would benefit from past position updates are not reachable via multihop. This also explains the performance difference between the real-time centralized and the best case distributed solutions for $N_B = 1$ and $N_U = 6$. However, we observed that when the number of beacons is increased ($N_B = 2$), the performance of the different information sharing schemes almost overlap, showing little gain in sharing past estimates over single or multiple hops. Nevertheless we still observe a significant gain over the non-collaborative case.

There may be times when the best performance is needed, which requires vehicles to share updates about past position estimates over single and multiple hops. However, we observed that in most cases, it is sufficient to periodically share only the current position estimate with neighbors.

5. Implementation details

5.1. Message representation

The sum-product algorithm operates efficiently on the pdf of discrete random variables while both position and velocity are most naturally defined over continuous space. To address this problem, we approximate the true pdf of states by piece-wise constant distributions that are uniformly weighted over 2D grids [7]. We have shown in our prior work that if these distributions are binary weighted, the entire sum-product algorithm can be implemented in binary logic which significantly reduces the computation overhead [7].

To enable vehicles to control their energy consumption without significantly deteriorating their localization accuracy we propose to decouple the representation used when communicating pdfs over the wireless channel from that used internally by vehicles. A more in depth description of the proposed solution and how pdfs of different granularities can be compared can be found in our prior work

[8]. The key point is that we can adapt the number of bytes used to communicate pdfs over the wireless channel, without significantly deteriorating the performance of the algorithm. This is summarized in Fig. 8 where we compare the localization performance for various values of N_{comm}, the number of bytes used to represent communicated pdfs and $N_{internal}$, the number of bytes used to represent pdfs local to the nodes. The conclusion we can draw from this figure is that we maintain sufficient accuracy when only transmitting 18 bytes of data for each pdf and maintaining an internal pdf of 512 bytes. This low communication overhead is key for a practical implementation of this system. There is a trade off between the number of grid points used to represent pdfs internally, $N_{internal}$, and the computation overhead of the algorithm. The number of grid points is inversely related to the size of each grid. In the next section we will discuss why and how the grid size adversely affects the accuracy of the algorithm and propose an implementation to counter this effect as much as possible under practical constraints.

5.2. Implementation of the factor graph for effective information fusion

While uniformly weighted piecewise constant distributions minimize the computation and energy overhead of the sum-product algorithm, they deteriorate the accuracy of position estimates by introducing rounding errors. In this section we present why these errors are introduced and provide an approach that minimizes their effect.

The local factor graph of each vehicle essentially fuses distance estimates to neighbor vehicles, obtained at different times, to compute the pdf of all the states specified by the vertices of the graph. Each distance measurement (in combination with an estimate of the position of the neighbor vehicle) provides an independent estimate of the position of the vehicle at the time that it was obtained. In principle the measurements of the vehicle's motion allow these estimates to be translated to any past or future time, t where they are fused. The translation in time is mathematically given by

$$P(t|z_i) = P(t_i|z_i) + \overline{v}(t_i,t)|t_i - t| + \varepsilon_v|t_i - t| \quad (8)$$

where z_i denotes the distance estimate obtained at time t_i, $P(t_i|z_i)$ denotes the estimate of the position of the vehicle at time t_i due to the distance measurement obtained at t_i alone, $\overline{v}(t_i,t)$is the estimate of the vehicle's average velocity in the interval (t_i,t) and $\varepsilon_v$is the error in velocity measurements.

As the error term in the above equation shows there is additional uncertainty in the translated position estimate $P(t|z_i)$ since measurements of the vehicle's motion are not exact.

In practice, the translation step given by Eq. (8) is performed numerically by the SP algorithm. This introduces a rounding error every time a translation is performed because the true pdf is approximated by the binary weighted piecewise constant pdfs (that were described in the previous section). In effect the true pdf of any state is smeared every time information is passed from one state in the graph to an adjacent state. The extent of smearing at each stage depends on the grid size of the representation. Further, the rounding errors introduced due to grid size have a cumulative effect when information available in one state has to pass through many intermediate states (that do not provide additional information) in order to be merged with information available in a different state in the graph.

To further elucidate the effect of intermediate states and grid size on rounding errors, consider the example factor-graph shown in Fig. 9 where distance estimates are obtained at two time instances t_1 and t_6. Each of these measurements provides an independent estimate of the position of the vehicle at the time that it was obtained denoted as $P(t_1|z_1)$, and $P(t_6|z_6)$ respectively. Now to obtain the final estimates of the positions at times t_1 and t_6 position information obtained at time t_1 has to be translated to t_6 and vice versa. In essence we need to compute $P(t_6|z_1)$ and $P(t_1|z_6)$. These estimates can be obtained as per Eq. 8. However, the factor-graph in Fig. 9 shows that there are 4 intermediate states between t_1 and t_6. These states do not have any measurements of distance associated with them and are solely present because the application requires an estimate of the vehicle's position at these times. Since the SP algorithm computes the pdf of any state using only that of its neighbor states, rounding errors are added at every intermediate state when $P(t_1|z_1)$ is translated to t_6. Therefore, these errors have a cumulative effect.

To minimize the cumulative effect of rounding errors the size of the grids used to represent pdfs must be very small and far more grid points are required which increases the computation overhead. This problem becomes specially pronounced when distance measurements become available very intermittently compared to the tracking granularity, δ. To address this issue we propose an alternate implementation of the factor-graph that was earlier presented in Section 4. Note that logically our solution still follows the one presented in Section 4 to estimate positions in real time.

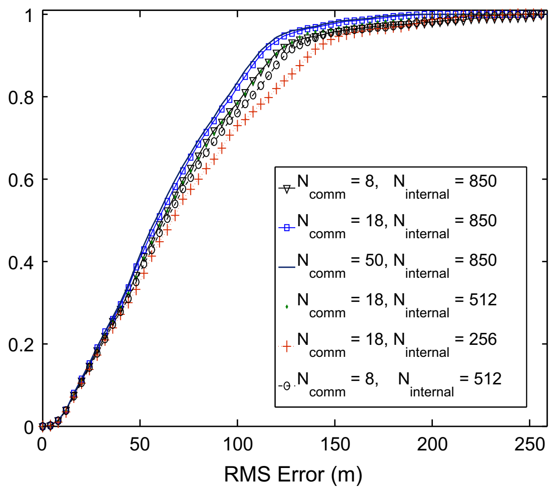

Fig. 8. CDF of RMS error in position estimates as a function of the number of bytes used to communicate each message, N_{comm} and the number of bytes used to represent the message internally, $N_{internal}$.

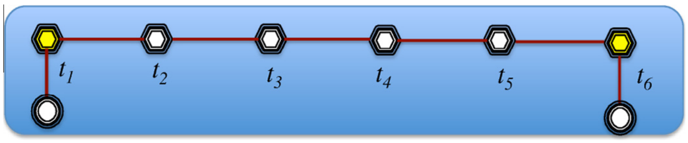

Fig. 9. Factor graph with 4 intermediate states between t_1 and t_6.

We refer to states corresponding to times when at least one distance measurement is available as informative states. All other states are referred to as empty states. In our proposed implementation the pdfs of the unknown states are computed in two steps as shown in Fig. 10. We first define the factor-graph over all informative states. The implementation of this step for the original factor graph shown in Fig. 2(b) is presented in Fig. 10(a). Note that in this step aggregating the motion measurements over the intervals between the informative states eliminates the empty states. The sum-product algorithm is then executed on this factor-graph to estimate the pdf of all informative states. By eliminating the intermediate empty states, the effect of rounding errors are minimized when combining distance estimates obtained at different times.

Next we interpolate between the informative states (using motion measurements) to compute the pdf of all other states. The factor-graph corresponding to this step (for the original FG, Fig. 2(b)) is shown in Fig. 10(b). As shown in the figure, the pdf of each state in the interval (t_1, t_2) is computed using the estimates at t_1 and t_2 and the appropriate aggregated motion measurements. Similarly, states in the interval (t_2, t_3) are computed from the position estimates at t_2 and t_3 and those later than t_3 are computed using the estimate at t_3 alone.

We now evaluate the performance of our proposed FG implementation versus the one where all the unknown states are estimated in one shot. In order to isolate the effect of intermediate empty states on the localization accuracy from all other parameters that affect performance, we simulated a scenario where only one vehicle was tracked using distance estimates from beacons. A measurement of distance was made with a beacon every 36s. Three beacons were used for tracking. The vehicle position was estimated every 3s. Keeping the maximum number of grid points used to represent the pdf of position estimates the same, we compared the RMS error in position estimates obtained over 20 min of simulation time using the FG implementation shown in Fig. 2 and the proposed implementation shown in Fig. 10. The cumulative distribution function (cdf) of the RMS error in estimates for the two implementations is shown in Fig. 11(a) and the RMS error versus time is shown in Fig. 11(b). From these results we observe that the localization accuracy improves significantly using our proposed implementation.

In summary, for real-time tracking the FG implemented by each vehicle will conceptually evolve as described in Section 4 however we implement it as explained in this section. In addition, the grid size used to represent pdfs internally is kept small by using signifcantly larger number of bytes to represent these pdfs internally compared to the number of bytes used to communicate these pdfs between vehicles. This essentially translates to choosing a much larger value for the parameter $N_{internal}$ compared to the parameter N_{comm}.

6. Deployment considerations

For the simulations in this section we compare non-collaborative localization to sharing the current state with a 1-hop neighbor (shortened to “Collaborative” for the remainder of this section) because it was shown in our previous work [8] that the localization accuracy obtained by this scheme was the same as sharing all information with all reachable nodes. We specifically investigate how key factors, namely, direct communication with beacons, network connectivity, and inaccuracies in the position estimates limit the scenarios where the proposed collaborative algorithm outperforms a non-collaborative approach.

6.1. Dependence on proximity to beacon

Since position information always percolates into the network starting at a beacon, one of the factors that naturally determines the error in position estimates is the proximity of nodes to beacons. In this section we quantify the impact of direct communication with beacons on the localization (relative to other factors). To do this we simulated a scenario with 6 unknown nodes and one beacon. The trajectory of one of the unknowns is shown in Fig. 12(a). The trajectory of other nodes followed a similar pattern, however each track was generated independent of the others. The trajectory shown in Fig. 12(a) is color coded into three segments to indicate the level of connectivity of the node to the beacon and to other nodes

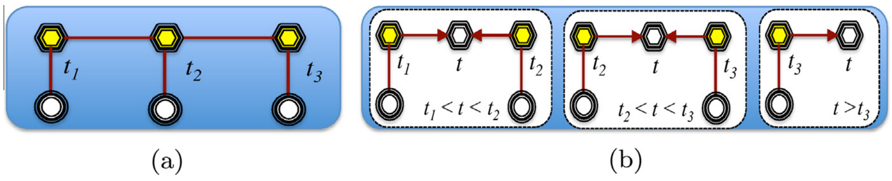

Fig. 10. Proposed implementation. (a) FG for estimating informative states. (b) FG for estimating empty states from informative states.

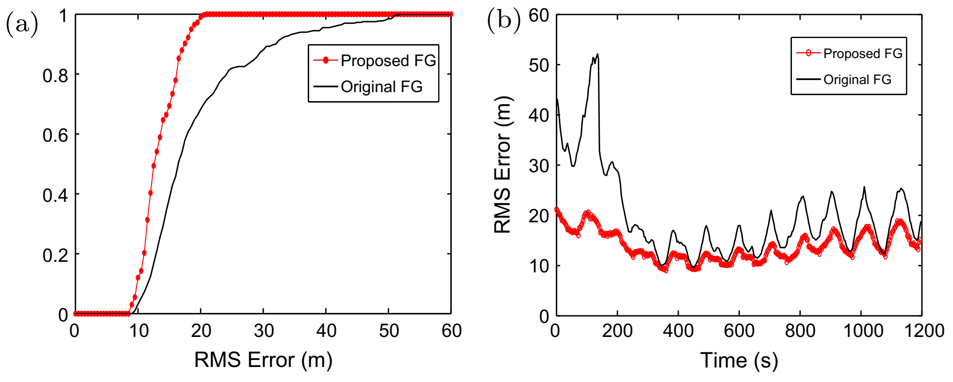

Fig. 11. Performance of proposed FG implementation versus original. (a) Cumulative distribution function of RMS error. (b) RMS error versus time.

in the network. The segments of the trajectory in red correspond to times when the node was within the communication range of the beacon, the segments in green correspond to times when it was within the communication range of at least another node and the segments in black correspond to times when it was not within the communication range of any node.

For the above described scenario we tracked the location of all unknowns in real-time using both collaborative and non-collaborative localization. To quantify the impact of direct communication with a beacon on the localization accuracy, we separated the position estimates obtained during an entire simulation into two sets. The first set corresponded to estimates obtained at times when a node communicated directly with a beacon (the red segments of the trajectory in Fig. 12(a)) and the second set consisted of times when the node could not receive any beacon signal (the green and black segments of the trajectory in Fig. 12(a)). The cdf of the RMS error in estimates for both sets are shown in Fig. 12(b).

Fig. 12(b) shows the relative benefit of collaborative tracking to non-collaborative tracking based on direct proximity to beacons. We observe that there is a much greater disparity in the localization performance between the non-collaborative and collaborative frameworks when nodes are not in the range of the beacon compared to when they are. If a node is in the range of beacons, there is not much gain from using collaboration. However, if it is out of range of beacons significant gains can be obtained by leveraging the localization information of other nodes. Notice that this gain can only occur when collaboration provides the nodes good information, i.e. the collaboration nodes are in communication with a beacon. While the specific trajectory is not important, what is important is whether or not a node can receive good information from other nodes when they are collaborating. We will see examples later where sharing bad information can actually hurt the localization performance of a specific node.

The results support what we intuitively expect: the error in position estimates is least when a node is in the direct communication range of a beacon. Additionally, we observe that both in the case of collaborative and non-collaborative localization, there is significant disparity in the error when nodes are within the range of the beacon compared to when they are not. This indicates that a key factor in determining the position accuracy of a node is whether or not it is within the communication range of a

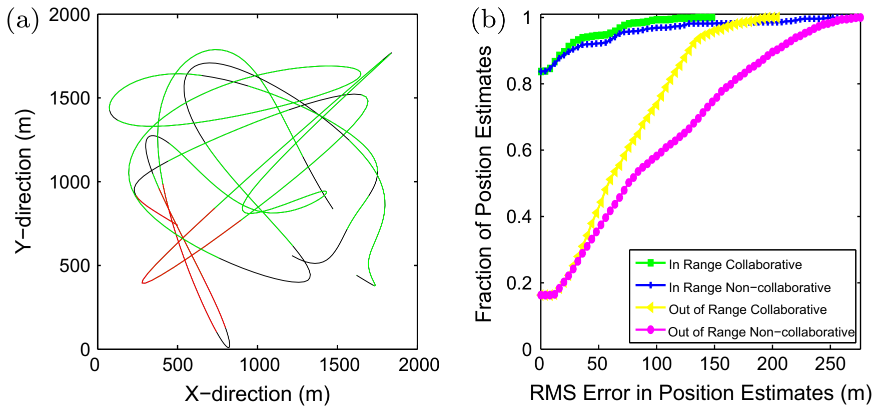

Fig. 12. (a) Trajectory of one of the nodes under evaluation with regions indicating direct communication with beacon (red), regions indicating communication with only other unknown nodes (green) and regions of no communication (black). (b) CDF of RMS error comparing the difference between the error when nodes are in range of the beacon versus when they are out of range of the beacon. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

beacon and the benefit of direct communication with a beacon outweighs the gains from inter-node collaboration.

Based on the above findings we would like to develop the importance of a node periodically receiving localization information from a beacon. Addressing this question relates to strategizing the placement of beacons in a network, as the number of available beacons is increased.

6.2. First hand versus second hand information

In this section we consider the following question: *If the number of beacons available for tracking is increased, would it be better to distribute them uniformly or non-uniformly over the area of deployment?*

We consider the above question because when the spatial distribution of beacons is not uniform we expect the error of a subset of the nodes to be minimized. In principle these nodes can then localize other nodes in the network that have less frequent (or no access) to beacons. On the other hand when the beacons are uniformly distributed in the network we expect all nodes to achieve approximately the same level of accuracy. Since most nodes do not have an information advantage over others, we would also expect little gains from collaboration in such a scenario.

To investigate the above conjectures we considered two simulation scenarios shown in Fig. 13(a) consisting of six stationary beacons and two unknowns that are always within the communication range of each other. In one of the scenarios all six beacons were arranged linearly in a *non-staggered* fashion such that the bottom node (whose trajectory is shown in blue) is always within the range of a beacon and the top node never hears a beacon. In the second scenario the beacons were placed in a *staggered* fashion, such that the bottom (blue) track intersects the communication range of three of the beacons and the top (green) track intersects the communication range of the other three beacons. We considered the localization error of the top (green) track to compare the benefit of receiving information from a beacon and a neighboring node (the staggered beacon arrangement) to the case where we only receive information from a neighboring node who receives information from twice as many beacons (the non-staggered beacon arrangement). The cdf of the RMS error in the position estimate of the top track is shown in Fig. 13(b). The results show that the staggered beacon arrangement provides much better localization accuracy for the top node compared to the non-staggered arrangement. We also observe that there is little gain from collaboration in the staggered case. This is because in this case the errors in the position estimates of both nodes are comparable. Consequently, there is little gain from sharing these estimates. However, in the non-staggered arrangement where the top node never hears a beacon there is noticeable gain from communicating with the bottom node.

Additionally, we considered the case shown in Fig. 14(a) where the beacon range is extended so that the bottom node intermittently receives updates from two beacons at a time to further reduce its error. In this case there is an additional improvement in the accuracy of the top track when the tracking is done collaboratively as shown in Fig. 14(b). This indicates that we can increase the location accuracy of both nodes by increasing the number of beacons used to localize one of them. However, this gain is clearly inferior to allowing the top node to receive location information from a beacon directly.

6.3. Redundant information

In this section we would like to understand how much collaboration is really useful and when. We evaluate whether a node can improve its estimate when communicating with nodes traveling in similar trajectories. To do this we consider the networks of Fig. 15, where one node is never in communication range of a beacon while the rest of the nodes are in range of the beacon at some point during the simulation. In the first case, Fig. 15(a), there

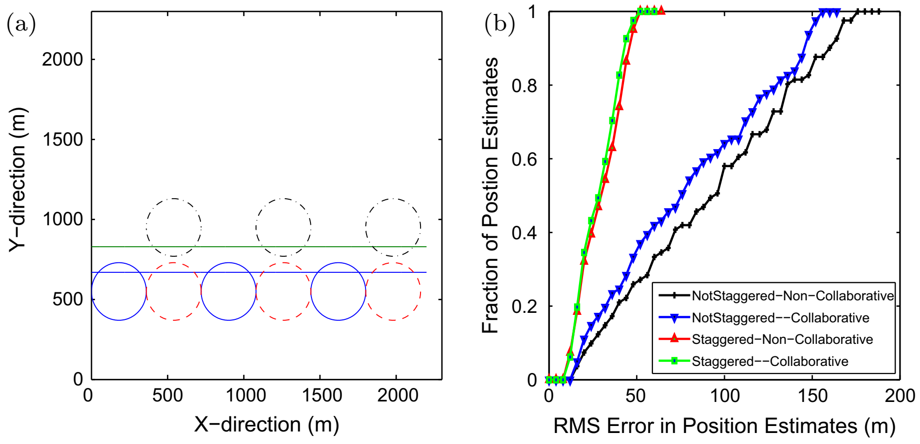

Fig. 13. Experiments describing hearing a beacon versus receiving location information from another unknown (a) *Non-Staggered:* (red dotted line for beacon range) the top node cannot communicate with the beacons *Staggered:* (black .- line for beacon ranges) the top node periodically communicates with beacons (b) CDF of error in position estimates for collaborative and non collaborative localization schemes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

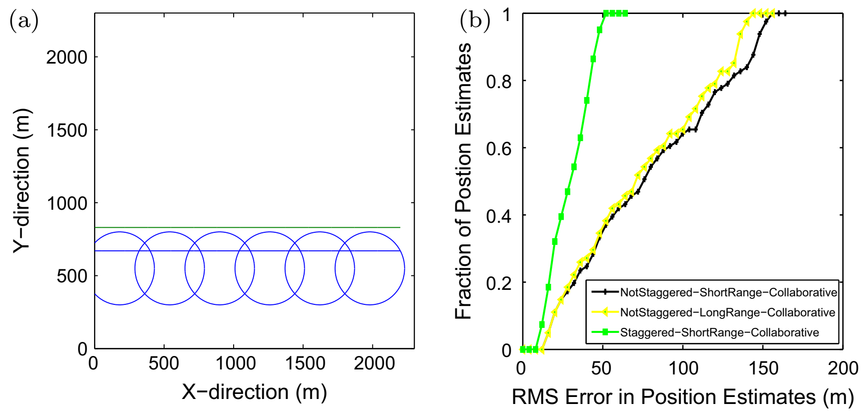

Fig. 14. (a) *Non-Staggered* beacon arrangement with increased beacon transmission range (blue .- line for beacon ranges) and (b) results plotted for the top (green) trajectory. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

are 2 nodes following similar trajectories. The bottom node is in range of the beacon for a portion of the simulation while the top node can only receive localization information by communicating with the bottom node. The other case presented in Fig. 15(b) is different from Fig. 15(a) in that there are now five nodes that receive a signal from the beacon and can communicate with the node that never hears a beacon.

The results of these simulations are shown in Fig. 15(c), plotted only for the node that can never communicate with the beacon. These results show that the collaborative case with only a two-node network provides the best localization accuracy. In fact, the collaborative result performs worse than the non-collaborative case in the six-node network. This brings up an interesting consideration about the proposed collaborative localization framework. When nodes have poor estimates of their positions the collaborative localization will actually decrease the localization accuracy of each node. In other words, sharing

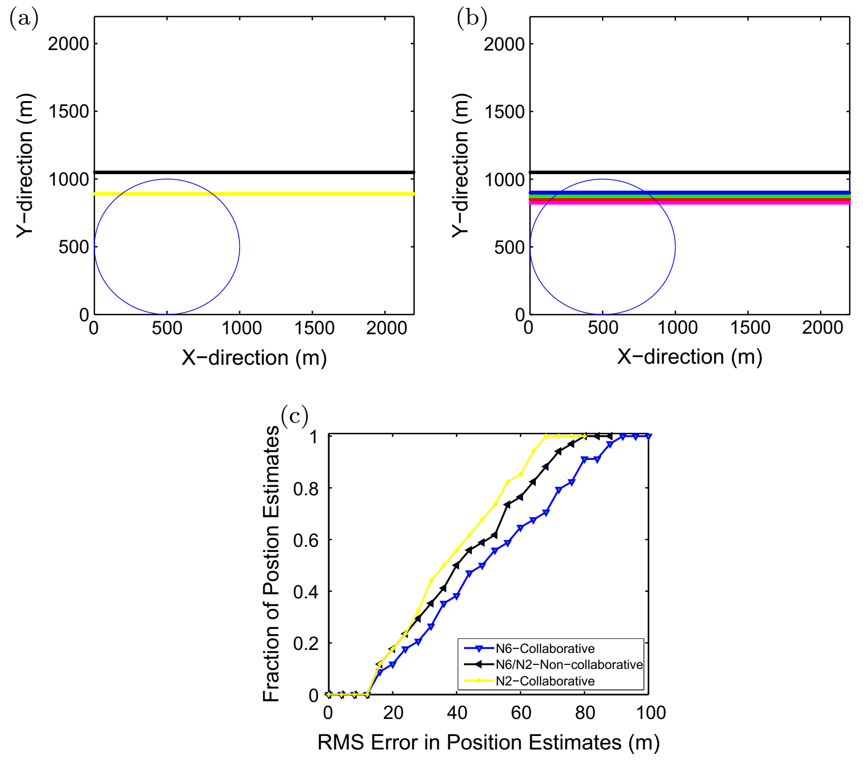

Fig. 15. Trajectories quantifying the results of sharing redundant information (beacon range shown in blue ellipse): (a) 2 node network, (b) 6 node network and (c) results-CDF of RMS error in position estimates: note that the performance of the non-collaborative tracking is the same for all cases. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

bad information produces worse results than sharing no (or less) information. This effect is due to practical limitations on the implementation of the algorithm rather than an inherent property of the algorithm.

The fact that the six node network performed worse than the two node network can be explained by revisiting the discussion of Section 5.2. In the above considered simulation scenarios, the five bottom nodes' localization accuracy decreases when they leave the proximity of the beacon, yet they are still receiving ranging information from each other. Each time new range information is received by a node from a neighboring node, an additional state is inserted into the factor graph. As such the additional states introduced in the graph are not informative since all nodes have similar errors and these errors are growing monotonically. This causes us to go from our desired case of Fig. 10 to the case we were trying to get away from in Fig. 9. The introduction of these uninformative intermediate states causes our solution to suffer the rounding errors discussed in Section 5.2. As explained earlier in that section, these rounding errors are a result of representing the estimated pdfs using only a finite and nominal number of grid points. As such this effect would not come into play if the pdfs could be represented with infinite accuracy. The above simulation scenario was specifically chosen to show that when all nodes in the network have poor estimates of their positions, sharing redundant information does not provide any benefit, but rather can hurt the localization accuracy of the nodes due to practical limitations in the implementation of the algorithm.

Our simulation results show that direct communication with beacons, network connectivity and inaccuracies in the position estimates limit the scenarios where the proposed collaborative algorithm outperforms a non-collaborative approach. To summarize our conclusions:

1. The localization accuracy of any node is primarily determined by the extent of time it is in direct communication with beacons. If multiple beacons are available it is more beneficial to distribute them in such a way that all vehicles have an equal opportunity to periodically hear at least one beacon
2. If a node is in the range of beacons, there is not much gain from using collaboration. However, if it is out of range of beacons significant gains can be obtained by leveraging the position estimates of other nodes.
3. Collaboration is beneficial when nodes do not have direct communication with beacons, however, due to practical limitations of the factor-graph algorithm, nodes with large and growing position errors should refrain from sharing their estimates with others.

7. Conclusion

In this paper we reviewed a low overhead collaborative tracking solution for underwater mobile networks that estimates the location of vehicles in real-time and in a distributed fashion. We identified key factors that fundamentally limit the performance of real-time (centralized and distributed) solutions, quantifying their effects via simulations. We also showed that in certain scenarios, vehicles need to share information over multiple hops for best case performance. However, in our proposed scheme they only use information that is obtained locally. We further tested the collaborative localization framework against different motion models and concluded that in a network where nodes follow more predictable trajectories, the localization performance can be most significantly enhanced by strategically placing beacons so that all vehicles have the opportunity to periodically hear a beacon. The collaborative scheme is less suited to such networks, where there is little disparity in the localization accuracy of vehicles due to sufficiently frequent communication with beacons. However, networks where the vehicles move in an unpredictable fashion are less amenable to a strategic placement of beacons. For such networks significant gains are possible by using our proposed collaborative solution. Even still, the collaborative approach does have its limitations. In poorly designed networks, where nodes experience extended outages in beacon communication as a group, the collaborative localization framework can do more harm than good. By avoiding inter-node communication during these times, the collaborative localization can effectively expand the network, allowing good position estimates to nodes who temporarily or even permanently are not in range of a beacon.

Acknowledgements

This work is part of the project "Networked Sensor Swarm of Underwater Drifters" supported by the NSF under Award Number 1035828, "INSPIRE Track I: Distributed Sensing Collective to Capture 3D Soundscapes" and by the National Science Foundation under IGERT Award Number DGE-0966375, "Training, Research and Education in Engineering for Cultural Heritage Diagnostics." Additional support was provided by the Qualcomm Institute at UC San Diego, the Friends of CISA3 and the World Cultural Heritage Society. Opinions, findings, and conclusions from this study are those of the authors and do not necessarily reflect the opinions of the research sponsors.

Appendix A

This paper is an extension of our previous work on collaborative tracking [8], published in the International Conference on Underwater Networks and Systems (WUWNet'12). In this paper we have made two key additions compared to the conference version:

1. An in depth discussion of the implementation of the Factor-Graph Algorithm, presented in Section 5.2
2. Deployment considerations that can significantly impact the localization accuracy of the network, presented in Section 6. In this section we have identified and analyzed scenarios where our collaborative approach shows significant gains, as well as those that highlight the limitations of our approach

The materials presented in both Sections 5.2 and 6 have not appeared in any previous publication.

References

[1] J. Jaffe, C. Schurgers, Sensor networks of freely drifting autonomous underwater explorers, in: Proceedings of the 1st ACM International Workshop on Underwater Networks, WUWNet '06, ACM, New York, NY, USA, 2006, pp. 93–96. http://doi.acm.org/10.1145/1161039.1161058.

[2] M.F. Fallon, G. Papadopoulos, J.J. Leonard, N.M. Patrikalakis, Cooperative auv navigation using a single maneuvering surface craft, Int. J. Robot. Res. (2010). 0278364910380760.

[3] G. Rui, M. Chitre, Cooperative positioning using range-only measurements between two auvs, in: OCEANS 2010 IEEE – Sydney, 2010, pp. 1–6. http://dx.doi.org/10.1109/OCEANSSYD.2010.5603615.

[4] M. Erol, L.F.M. Vieira, M. Gerla, Localization with dive'n'rise (dnr) beacons for underwater acoustic sensor networks, in: Proceedings of the Second Workshop on Underwater Networks, WuWNet '07, ACM, New York, NY, USA, 2007, pp. 97–100. http://dx.doi.org/10.1145/1287812.1287833.

[5] A. Teymorian, W. Cheng, L. Ma, X. Cheng, X. Lu, Z. Lu, 3D underwater sensor network localization, IEEE Trans. Mobile Comput. 8 (12) (2009) 1610–1621, http://dx.doi.org/10.1109/TMC.2009.80.

[6] Z. Zhou, J.-H. Cui, S. Zhou, Efficient localization for large-scale underwater sensor networks, Ad Hoc Netw. 8 (3) (2010) 267–279. http://dx.doi.org/10.1016/j.adhoc.2009.08.005. <http://www.sciencedirect.com/science/article/pii/S157087050900081X>.

[7] D. Mirza, C. Schurgers, Collaborative tracking in mobile underwater networks, in: Proceedings of the Fourth ACM International Workshop on UnderWater Networks, WUWNet '09, ACM, New York, NY, USA, 2009, pp. 5:1–5:8. http://doi.acm.org/10.1145/1654130.1654135.

[8] D. Mirza, C. Schurgers, R. Kastner, Real-time collaborative tracking for underwater networked systems, in: Proceedings of the Seventh ACM International Conference on Underwater Networks and Systems, WUWNet '12, ACM, New York, NY, USA, 2012, pp. 3:1–3:8. http://dx.doi.org/10.1145/2398936.2398940.

[9] Z. Zhou, Z. Peng, J.-H. Cui, Z. Shi, A. Bagtzoglou, Scalable localization with mobility prediction for underwater sensor networks, IEEE Trans. Mobile Comput. 10 (3) (2011) 335–348, http://dx.doi.org/10.1109/TMC.2010.158.

[10] H.-P. Tan, R. Diamant, W.K. Seah, M. Waldmeyer, A survey of techniques and challenges in underwater localization, Ocean Eng. 38 (14) (2011) 1663–1676.

[11] M. Erol-Kantarci, S. Oktug, L. Vieira, M. Gerla, Performance evaluation of distributed localization techniques for mobile underwater acoustic sensor networks, Ad Hoc Netw. 9 (1) (2011) 61–72. http://dx.doi.org/10.1016/j.adhoc.2010.05.002. <http://www.sciencedirect.com/science/article/pii/S1570870510000612>.

[12] B. Chen, D. Pompili, Uncertainty-based localization solution for under-ice autonomous underwater vehicles, in: Proceedings of the Sixth ACM International Workshop on Underwater Networks, ACM, 2011, p. 13.

[13] F. Kschischang, B. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inform. Theory 47 (2) (2001) 498–519, http://dx.doi.org/10.1109/18.910572.

[14] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi, M. Zorzi, Ns2-miracle: a modular framework for multi-technology and cross-layer support in network simulator 2, in: Proceedings of the 2Nd International Conference on Performance Evaluation Methodologies and Tools, ValueTools '07, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2007, pp. 16:1–16:8. <http://dl.acm.org/citation.cfm?id=1345263.1345284>.

Diba Mirza is curreclty a Lecturer in the Department of Computer Science and Engineering at the University of California, San Diego. She received a M.S. and Ph.D. in Electrical Engineering from UCSD in 2010 and a B.E. in Electrical Engineering from Birla Institute of Technology and Science, Pilani, India in 2002. She was a postdoc at the Department of Computer Science and Engineering at UCSD from 2010 to 2013. Her research interests include underwater acoustic navigation and networked systems.

Perry Naughton is currently a Ph.D. student in the Electrical Engineering department at the University of California, San Diego. He is interested in remote imaging tools for cultural heritage diagnostics, especially in underwater environments. He received his B.S. in Electrical Engineering from the University of California, San Diego in 2012.

Curt Schurgers received his Ph.D. from UCLA in Integrated Circuits and Systems, and his Engineering degree from the Katholieke Universiteit Leuven (Belgium). He was a Lecturer at UCLA in VLSI System Design, and held research assistantships at UCLA Networked & Embedded Sysytems Lab and the Interuniversity Microelectronics Center in Belgium. He joined UCSD in 2003 as a Professor in ECE, and currently works at the UCSD Qualcomm Institute as a Research Engineer, Project Leader, Area Manager and Development Engineer.

Ryan Kastner is currently a Professor in the Department of Computer Science and Engineering at the University of California, San Diego. He received a Ph.D. in Computer Science at UCLA, a masters degree (MS) in Engineering and bachelor degrees (BS) in both Electrical Engineering and Computer Engineering, all from Northwestern University. He is the co-director of the Wireless Embedded Systems Master of Advanced Studies Program. He also co-directs the Engineers for Exploration Program. His current research interests reside in the realm of embedded system design, in particular, the use of recon gurable computing devices for digital signal processing.