Real-time Collaborative Tracking for Underwater Networked Systems

Diba Mirza
Department of Computer
Science & Engineering
University of California,
San Diego
dimirza@eng.ucsd.edu

Curt Schurgers
California Institute for
Telecommunications &
Information Technology
University of California,
San Diego
cschurgers@ucsd.edu

Ryan Kastner
Department of Computer
Science & Engineering
University of California,
San Diego
kastner@cs.ucsd.edu

ABSTRACT

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 collaborative tracking techniques. More specifically the Maximum Likelihood (ML) position estimates of a mobile collective can be computed from measurements of relative positions and motion, albeit centrally and non-real time. While for a number of underwater applications non-real-time position estimates may suffice, in this paper we focus on the design of a collaborative tracking solution that operates in real-time, yet is scalable and energy-efficient. Using the ML solution as a baseline, we identify key factors that fundamentally limit the performance of real-time (centralized and distributed) solutions, quantifying their effects via simulations. In the remaining solution space, we propose a low overhead scheme for real-time and distributed tracking. The specific challenges that we address include determining what information should be shared between vehicles, how this information must be encoded to minimize the communication overhead and when should vehicles communicate with each other to achieve the best performance with the minimum energy consumption. Our proposed technique can strategically trade off localization accuracy and energy consumption, achieving more than 50% reduction in the communication overhead compared to any fixed scheme.

Categories and Subject Descriptors

C.2.3 [Computer-Communication Networks]: Network Operations; C.4 [Performance of Systems]: [Design studies]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WUWNet'12, Nov. 5 - 6, 2012 Los Angeles, California, USA.
Copyright 2012 ACM 978-1-4503-1773-3/12/11 ... \$15.00.

General Terms

Algorithms, Performance

Keywords

Underwater Networks, Real-time Tracking, Acoustic Networks

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 [2]. 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 [9][10]. 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 never be within the communication range of a beacon or it may move out of range over time. On the other hand, a number of localization techniques (e.g. multi-dimensional scaling and iterative multilateration) have been proposed for static terrestrial wireless sensor networks, where only a subset of devices have access to GPS. However, these techniques apply to stationary networks only. 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][7][8]. 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 [6], compared to both individual tracking and collaborative static localization. We proposed a collaborative solution that tracks the 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, however, 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. While non-real time location information is sufficient in some cases, position information is needed in real-time for a wide range of oceanographic applications. These include early warning systems, vehicle navigation and control, adaptive sampling and formation control. Real-time location information is also used in certain routing protocols. As opposed to our past work that only handles non-real-time localization, the goal of this paper is to design a collaborative tracking technique that operates in real time and supports the important applications mentioned above.

Creating a real-time tracking solution for mobile underwater networks is a difficult task due to the unique challenges posed by the underwater environment. The underwater acoustic channel is highly band-limited, and as a result acoustic modems offer very low data rates compared to RF radios. At the same time, they also consume high energy for data transmission. Furthermore, the underwater vehicles themselves are necessarily battery operated and highly energy constrained. The challenge for our real-time tracking solution is to achieve good localization accuracy in the presence of these severe communication and energy constraints.

We start our exploring by modifying our existing ML solution to compute positions in real-time rather than all at once after the mission, but keeping the computations in a central processor (e.g. a buoy at the surface). In this case, vehicles would need to constantly send their measurements (obtained locally) to a single location, where their positions is computed in real-time and communicated back to them. This is clearly not a scalable solution, and not practically feasible given the resource constraints in our system. However, this centralized real-time algorithm will serve as a useful benchmark for further optimization, and to highlight the inherent loss in localization performance that results when we go from a non-real-time to a real-time solution. Going to a mere distribution solution, while eliminating a single point of failure, will not alleviate the resource constraints as we will show later. Instead, we need to create an approach that is localized in addition to being distributed. An algorithm that is localized only relies on information available in the immediate neighborhood of each device, thereby eliminating costly network-wide communication. The need for localized distributed algorithms has been well established in the area of sensor networking, and essentially boils down to decomposing a larger problem into a collection of local sub-problems. We will create 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. Before we can delve into our progression from a centralized real-time tracking approach to a distributed and then localized solution, we will first quickly describe our prior work on non-real-time centralized tracking as its ML solution forms the necessary basis from which we will develop our new contributions.

2. NON-REAL-TIME CENTRALIZED SOLUTION

Our original Maximum-Likelihood (ML) tracking method uses the framework of factor-graphs for position estimation, but can only operate once all information is available at a central location and thus necessarily runs post-facto, i.e. after the mission is completed. In this section our goal is to highlight the key mechanisms that allow information to be effectively fused in 4D (time and space).

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. The so-called ‘temporal’ constraints are estimates of vehicle velocity, acceleration and heading as gathered by the navigational sensors on-board each vehicle. 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 is available in our prior work [6]. The theoretical foundations of this technique have been laid out by Kschischang et. al.[1]. 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 as discussed in the Appendix.

Figure 1: Centralized non-real-time solution (a): Dependency graph (b): Iterative message passing for position estimation

Figure 1(a) shows the dependency graph for a network of four vehicles. We have shown the graph for only four vehicles not to 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 factor-graph described above is constructed in non-real-time, after all the measurements are available at a central location. The unknown positions are computed by running the sum-product (SP) algorithm (in non-real-time) on the factor graph. 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).

The links in the dependency graph shown in Figure 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). When we run the sum-product algorithm on the graph, this information eventually percolates to all the unknown states (vertices) of the factor-graph along the graph edges as depicted in Figure 1(a). To understand how this is achieved we examine the operation of the sum-product (SP) algorithm on a small section of the graph highlighted in Figure 1(a) and shown in detail in Figure 1(b).

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; see also the Appendix) The pdfs computed for each vertex are then passed on as messages to all neighbor vertices. We will refer to the vertex associated with any vehicle i at time t_j as (V_i, t_j) . Now, to visualize how information percolates from the beacon-vertices into the graph let us examine the messages passed over four iterations of the SP for the subsection of the graph shown in Figure 1(b):

Iteration 1: Beacon positions (b_1, t_2) and (b_2, t_3) are passed on to vertices (V_1, t_2) and (V_2, t_3) respectively.

Iteration 2: The pdf of the position of vehicle 1 at time t_2 and that of vehicle 2 at time t_3 are estimated based on the positions of the beacons, (b_1, t_2) and (b_2, t_3) and the measurement of distance to each beacon. (Note that position information that travels along the vertical edges of the graph goes through a non-linear transformation because given the

distance between two vehicles and the position of either one, the other vehicle can be anywhere on a ring.) The new estimates are then passed on to neighbor vertices. Specifically vertex (V_1, t_3) receives messages from both (V_1, t_2) and (V_2, t_3) . Therefore, information on the location of any vehicle at some point in time constrains the location of a neighbor vehicle (with whom an estimate of distance is available).

Iteration 3: The position of vehicle 1 at t_3 is estimated using the messages received in the previous iteration. The new estimates are passed both to past (and future) states (vertex (V_1, t_2) and across vehicles (vertex (V_2, t_3)). Position information that is passed forward or backward along horizontal links increases in uncertainty because of the error in motion measurements and the physical time elapsed between states.

Iteration 4: With this iteration, position information from (b_2, t_3) has reached (V_1, t_2) after going through a series of transformations, and it is combined with the position information obtained from (b_1, t_2) to compute updated estimates of (V_1, t_2) . Similarly position information from both beacons has reached vertex (V_2, t_3) .

The key point in the above technique is that to effectively fuse position information, messages have to be passed a number of times across each of the links of the graph. 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.

3. REAL-TIME TRACKING

In this section, we will present the basic solution of a real-time tracking approach by introducing it in three steps. We propose an initial solution where all measurements are communicated to a central location for computing the vehicle positions. We next consider the problem of performing the computation in a distributed fashion where each vehicle carries out part of the computation and collaborates appropriately with all others in the network to achieve the best possible localization performance. As we will elaborate later in this section, for distributed tracking substantial information has to be shared between vehicles, over multiple hops. To reduce the energy overhead, we consider using only local information and further optimize this solution by strategically deciding how much information to communicate with neighbors and when.

3.1 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 2) 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 $\delta(s)$ as depicted in Figure 2. In addition,

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

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.

We implemented our centralized real-time tracking solution in the NS2 simulator [3] and evaluated its performance against the previously proposed offline solution. The simulation set up is described later in Section 3.3 with parameters given in Table 1. The localization performance is shown in Figure 3, where 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. In principle we can estimate the vehicle positions in 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*. Figure 3 shows the localization performance as a function of the smoothing parameter, τ .

Figure 3: Effect of smoothing parameter, τ on CDF of RMS error

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 3.2 we will discuss other such factors.

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

3.2 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 Figure 1(a) for $N = 4$) as N communicating subgraphs. This decomposed view of the centralized factor-graph is presented in Figure 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 Figure 2. If at a later time, more accurate estimates of these states are communicated to the vehicle, it incorporates this information into its FG.

Figure 4: Decomposed view of centralized dependency graph

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 range¹. We have depicted the problem that arises in the distributed case in Figure 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 Figure 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 Figure 5(b). However, in the distributed scenario this is not possible because A and B are no longer in communication range.

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

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 Figure 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.

¹Note 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.

Table 1: Simulation Parameters

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

3.3 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 below:

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 [3]. We simulated a network with 8 mobile vehicles and 1 beacon. The vehicles were placed in a 2 km x 2 km area. The beacons resided on the surface, while the vehicles were at different depths. We used the mobility model proposed in [6] 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 2), real-time centralized (see section 3.1), real-time distributed (best case) (see section 3.2) 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 Figure 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.

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

Figure 6: CDF of RMS error for proposed information schemes

varied. Figure 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 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.

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. The next question is whether vehicles need to always send their estimates periodically or can we further minimize the amount of information transmitted by them without deteriorating the performance.

4. ENERGY OPTIMIZATIONS

In the previous section we concluded that in most scenarios it is sufficient for vehicles to periodically share only their current position state with their immediate neighbors. Even though this limited information sharing scheme gives us the most gain, it may be possible to further reduce the amount of information transmitted by vehicles. We would like to find the minimum information that must be communicated to achieve the same performance as the above fixed scheme. The amount of localization data transmitted per node depends on the average number of position states transmitted and the bit-wise representation of these states. To reduce the energy overhead we need to minimize both these factors.

Figure 7: Performance for different network parameters

4.1 Dynamic Information Sharing

In the fixed information sharing scheme vehicles periodically transmitted their position state. However, only some of these states actually reduce the uncertainty of neighbor vehicles (as we will show in this section). Since each position state is a pdf, the amount of information needed to communicate the entire state is substantial. So, the question that we address is: *How can vehicles determine if their position information is going to be relevant to their neighbors without actually transmitting it?*

The key idea that we propose is that vehicles exchange some information about their estimates (using only a few bits) in order to predict the impact of actually transmitting their state. Naturally, vehicles with more accurate estimates would have the most impact on reducing the uncertainty of their neighbors' estimates. While the actual error in the estimates is not known, the entropy of the estimated pdfs gives a measure of the uncertainty in the estimates. Further, it can be transmitted using only one byte which is much less compared to the number of bytes needed to transmit the entire pdf, (the nominal number of data bytes used to communicate pdfs is discussed in section 4.2). Therefore, we use the entropy to design a scheme where vehicles dynamically decide when to transmit their position states. The entropy $H(p)$ of any piece-wise constant probability distribution $p(x)$ is computed as

$$H(p) = - \sum w_k \cdot \log(w_k / |V_k|) \quad (1)$$

where, $\{V_1, V_2, \dots, V_k\}$ are disjoint subsets that partition the space of x , w_k is the weight of each V_k s.t. $\sum w_k = 1$.

Using the entropy we propose the following condition for transmitting the estimated pdf of a vehicle's position. *Condition to transmit:* A vehicle transmits its current position state if its entropy is smaller than that of at least one its neighbors by a threshold, $\gamma = 2 \cdot \log(1 + \alpha)$, $\alpha > 0$.

The above condition requires vehicles to know the entropy of their neighbors. To minimize the number of times vehicles

explicitly share their entropy information we propose the following scheme. Each localization frame consists of 2 sub-frames which repeat periodically. In the first subframe a vehicle broadcasts the entropy of its current estimate if (a) it has not heard from any other vehicle (b) the condition to transmit is not satisfied which essentially means that its estimate is not accurate enough compared to those it has overheard so far. Otherwise, a vehicle broadcasts its own current position state (pdf). When a vehicle transmits its estimated position pdf, its neighbors deduce the vehicle's entropy from it using equation (1). So, by the end of the first subframe each vehicle knows the entropy of all its neighbors. If it has not transmitted its position state already it does so in the second subframe if the condition to transmit holds, otherwise it stays silent.

Figure 8: Trade off between tracking accuracy and energy consumption using proposed dynamic information sharing scheme

We evaluated our dynamic information sharing scheme using the simulation setup described in section 3.3. By varying the entropy threshold parameter α , vehicles changed the amount of information they shared with their neighbors thereby trading off energy consumption for localization accuracy. We evaluated the localization performance against the base scheme where each vehicle periodically shares its current estimate. Figure 8(a) shows the CDF of the RMS error in position estimates when the parameter α is varied. Figure 8(b) shows the percentage of data bytes transmitted for localization using the proposed dynamic information scheme relative to the fixed scheme. We observed significant reduction in the amount of data transmitted by vehicles for no loss in performance (compared to the base scheme) for small values of α , ($\alpha = 0$ and $\alpha = 0.1$). The energy consumption can be further traded off for localization accuracy by increasing the value of the parameter, α . In the next section we will show how this trade off also depends on the representation of the pdf of the position estimates.

4.2 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. We approximate the true pdf of states by piece-wise constant distributions that are uniformly weighted over 2D grids [6]. 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 [6]. In the 2D grid representation of the state space, the grid size affects the computation overhead, accuracy and energy consumption. To enable ve-

hicles 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. To this end we propose a two-level representation, depicted in Figure 9, which allows the length of the messages communicated over wireless links to be different from those passed locally on the FG of each vehicle.

Figure 9: Two level pdf representation (a) Representation communicated over wireless links (b): Internal representation

The first level representation, denoted as L_1 is a coarse approximation of the true pdf as depicted in Figure 9(a). It can be efficiently encoded in N_{comm} bytes using only one bit per grid. As the weights computed for the grids change, we dynamically adjust the boundary points over which L_1 is defined, effectively zooming into the region where the vehicle can be. The second level representation, denoted as L_2 is defined over all the non-zero grids of L_1 as depicted in Figure 9(b). A maximum of $N_{internal}$ bytes is used to achieve a much more refined representation. To minimize the communication overhead only L_1 is transmitted to neighbor vehicles. While both N_{comm} and $N_{internal}$ affect the localization performance, only N_{comm} affects the energy consumption. Therefore, we select $N_{internal}$ to be sufficiently high and vary N_{comm} in order to trade off localization accuracy and energy consumption. Figure 10 shows the effect of the message representation parameters on the tracking performance. The simulation set up is the same as before. We used our dynamic information sharing scheme with $\alpha = 0$. We initially chose large values for $N_{internal}$ and N_{comm} such that the localization error could not be further improved by changing either of these parameters. We then varied both N_{comm} and $N_{internal}$ and evaluated the localization performance. Results show that the performance is not very sensitive to the choice of N_{comm} as long as $N_{internal}$ is sufficiently large and N_{comm} is not too small. However, since $N_{internal}$ affects the computation overhead (see [6]) we choose nominal values for both parameters $N_{comm} = 18$ bytes and $N_{internal} = 512$ bytes, and further apply our dynamic information scheme to optimize the energy.

Figure 10: Performance vs. message representation parameters

5. 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 [6]. A number of real-time and distributed localization techniques have been proposed for underwater networks [4] [5] [7] [8]. Among these DNRL [4] and LSL [8] use multi-lateration while USP [7] 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 [8]. 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 [13]. A comparative performance evaluation of some of these schemes has been done in [12].

AUV navigation using a single mobile beacon is proposed in [9][10]. 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 [10]. 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 [11]. 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.

6. CONCLUSION

In this paper we proposed a low overhead collaborative tracking solution for underwater mobile networks that estimates the location of vehicles in real-time and in a distributed fashion. Using our previously proposed (offline) ML solution, 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. By strategically sharing information, our proposed distributed tracking can achieve localization accuracies close to the best case with substantial reduction in the amount of information transmitted compared to any fixed scheme.

7. ACKNOWLEDGEMENTS

This work is part of the project “Networked Sensor Swarm of Underwater Drifters” supported by the NSF under award number 1035828.

8. REFERENCES

- [1] F.R. Kschischang, B.J. Frey, H.A. Loeliger, “Factor graphs and the sum-product algorithm,” IEEE Trans. on Info. Theory, Vol.47, No. 2, 2001.
- [2] J. Jaffe and C. Schurgers, “Sensor networks of freely drifting autonomous underwater explorers,” WUWNet’06, 2006.
- [3] N. Baldo, F. Maguolo, M. Miozzo, M. Rossi and M. Zorzi, “NS2-MIRACLE: a Modular Framework for Multi-Technology and Cross-Layer Support in Network Simulator 2”, NSTOOLS 2007 - 1st International ICST Workshop on Network Simulation Tools, Oct. 2007.
- [4] M. Erol, L.F.M. Vieira and M. Gerla, “Localization with diveN’rise (DNR) beacons for underwater acoustic sensor networks”, WUWNet’07, 2007.
- [5] Z. Zhou, J.H Cui and A. Bagtzoglou, “Scalable localization with mobility prediction for underwater sensor networks”, INFOCOM’08, April 2008.
- [6] D. Mirza and C. Schurgers, “Collaborative tracking in mobile underwater networks,” WUWNet’09, Nov. 2009.
- [7] A.Y. Teymorian, W. Cheng, L. Ma, X. Cheng, X. Lu and Z. Lu, “3D underwater sensor network localization, IEEE Transactions on Mobile Computing”, vol. 8 pp. 1610-1621, 2009.
- [8] Z. Zhou, J. Cui and S. Zhou, “Efficient localization for large-scale underwater sensor networks”, Ad Hoc Networks vol. 8, 2010.
- [9] M.F. Fallon, G. Papadopoulos, J. J. Leonard, and N. M. Patrikalakis, “Cooperative auv navigation using a single maneuvering surface craft”, Intern. J. of Robotics Research, vol. 29, pp. 1461-1474, October 2010.
- [10] G. Rui and M. Chitre, “Cooperative positioning using range-only measurements between two AUVs”, Oceans’10, 2010.
- [11] B. Chen and D. Pompili, “Uncertainty-Based Localization Solution for Under-Ice Autonomous Underwater Vehicles,” WUWNet’11, 2011.
- [12] M. Erol-Kantarci, S. Oktug, L. Vieira and M. Gerla, “Performance evaluation of distributed localization techniques for mobile underwater acoustic sensor networks”, Ad Hoc Networks, vol. 9, pp. 61-72, 2011.
- [13] H.P. Tan, R. Diamant and W.K.G. Seah, “A survey of techniques and challenges in underwater localization”, Journal of Ocean Engineering, vol. 38 pp. 1663-1676, 2011.

APPENDIX

We denote the unknown position and velocity of any vehicle i at time t_k as $P_i(t_k)$ and $V_i(t_k)$ respectively. A vertex in the dependency graph is a simplified notation for a state-variable, $\mathbf{X}_i(t_k) = [P_i(t_k), V_i(t_k)]$ and a set of transformation functions (f_1, f_2 and f_3), as shown in Figure 11. By replacing each vertex in the dependency graph with these elements, the equivalent factor graph is obtained.

Figure 11: Vertex of vehicle i at time t_k

The transformation functions are defined as $f_1 = p(\mathbf{X}_i(t_k) | \mathbf{X}_i(t_{k-1}))$, $f_2 = p(\mathbf{X}_i(t_k) | \mathbf{X}_i(t_{k+1}))$ and $f_3 = p(P_i(t_k) | P_j(t_k))$. The exact definitions of these conditional distributions are given in our prior work [6]. In the above figure, 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 $\mu[\mathbf{X}_i(t_{k-1})]$, $\mu[\mathbf{X}_i(t_{k+1})]$ and $\mu[\mathbf{P}_j(t_k)]$, respectively, to compute independent estimates of the pdf of $\mathbf{X}_i(t_k)$, as per equation (2) of [6]. The outgoing messages $\mu[\mathbf{X}_i(t_k)]$ and $\mu[\mathbf{P}_i(t_k)]$ are computed as per equation (3) of [6].