Resource-aware Decentralization of a UKF-based Cooperative Localization for Networked Mobile Robots

Seyyed Ahmad Razavi and Eli Bozorgzadeh
Computer Science Dept.
University of California, Irvine
Irvine, CA 92695-3435
Email: srazavim,eli@uci.edu

Kanghee Kim
Dept. of Smart Systems Software
Soongsil University
Seoul, Korea
Email: khkim@ssu.ac.kr

Solmaz S. Kia
Mechanical and Aerospace Eng. Dept.
University of California, Irvine
Irvine, CA 92695
Email: solmaz@uci.edu

***Abstract*—Estimation techniques such as Unscented Kalman filter (UKF) are deployed for accurate joint location estimation in cooperative localization of cyber physical systems (CPS), e.g., to locate each robot in a cooperative mobile robotic network in GPS-denied environments. In order to avoid single point of failure in the centralized implementation of such estimation techniques, the decentralization of estimation algorithms has attracted considerable attention in the past two decades. However, the design of decentralized algorithms with reduced communication cost without loss in accuracy for compute-intensive estimation techniques such as UKF has been challenging. In the decentralized UKF, the tasks are partitioned and computed locally at robot nodes. Data communication overhead is overwhelming due to tight data dependency between the robots' computations. In this paper, we present a CPS framework for UKF decentralization in which computation and communication are tightly intertwined and computation replication is deployed in order to reduce the communication overhead among cooperative mobile robots. We demonstrate and evaluate the performance of our proposed work in a wireless network of 15 Raspberry Pi 3 B, with quad-core 1.2GHz 64bit CPU, emulating a network of mobile robots with onboard computation and communication capabilities. Our experimental results show that the End-to-End execution time of decentralized UKF prediction and update steps with replication are faster by up to 12.29 and 3.57 times, respectively, compared to the partially decentralized UKF algorithm of [1].**

I. Introduction

Cooperative cyber physical systems are driven by the tight coordination between computational components, physical sensors and the interaction with each other. Software development for such systems, because of the tight integration, heterogeneity of resources, as well as safety and application timing requirements can be very complex. Networked mobile robot operations are among the challenging cooperative CPS applications. An important component of the networked mobile robot operations is the localization of the robots. The fast and accurate localization is important because a delayed estimation will mislead navigation and other applications and may lead to a mission failure. Cooperative localization is a reliable scheme for localizing a team of communicating robots in GPS-denied environments (Figure 1). This localization technique uses relative measurements among the robots as a feedback signal to jointly estimate the location of robots.

Cooperative Localization (CL) algorithms deploy various estimation strategies such as Extended Kalman filters (EKF) [2], UKF [1], maximum likelihood [3], maximum a posteriori (MAP) [4], and particle filters [5], [6], [7], [8]. Among these techniques, EKF-based cooperative localization algorithms, due to their recursive nature and relative ease of implementation, have been studied extensively. However, the EKF, due to linearization approximation, is known to be inconsistent for highly nonlinear systems or when the filter has high initialization errors. UKF is an alternative recursive estimation filter, which is proven to work more consistently than the EKF for systems with nonlinear state and measurement models (c.f. [9]). For large systems, however, UKF is computationally more expensive than EKF. This paper focuses on UKF-based cooperative localization.

To increase the scalability and avoid a single point of failure, decentralization of localization has been promoted by robotics community. However, due to relative measurement updates, the local estimates of the cooperative robots are highly correlated. This creates a great challenge in the decentralization of cooperative localization algorithms. Decentralized cooperative localization algorithms normally come with a significant processing and communication requirements (see e.g., [10], [11], [12], [4], [13], [14]). Ignoring estimation correlations can lead to filter inconsistency and even divergence [15], [16].

Various decentralization schemes using EKF formulations have been proposed in the literature (see e.g., [10], [13], [14], [17]). In UKF, the estimation equations of the robots in the team are highly correlated. Thus, naive decentralization (or partitioning) of UKF algorithm can result in a large amount of data transfer among the robots. In [1], a partially decentralized UKF is proposed. in which the UKF equations are decoupled in a way that their computation is distributed among all the cooperative robots. However, a shared memory on a server is used to transfer data among robots, leading to

transfer of a large amount of data between the server and the robots. The communication time is usually much higher than the computation time, which makes this method inefficient.

The literature on the decentralized cooperative localization algorithms that is reviewed above is on the algorithmic decoupling of computations without loss in performance and accuracy (or with acceptable performance loss). The proposed methods lack rich knowledge on characteristics of the underlying network and computational resources. Wireless communication overhead, heterogeneity in embedded computation, and sensing features of mobile robots may incur significant delay and long execution time during localization. Therefore, decentralization is challenged by the tight coupling and integration of sensing, computation, and communication among the mobile agents. Without careful trade-off between computation and communication, a decentralized algorithm may suffer from unacceptably long End-to-End delay and may not be deployable for agents with high mobility. This mandates us to develop a CPS framework to overcome the complexity of decentralization of cooperative localization based on complex estimation techniques such as UKF without loss of of accuracy and computationally-identical to the centralized method.

We treat decentralization of UKF as a problem of k-way task partitioning problem. Each partition is a subgraph in UKF task graph that is computed in each robot. Unlike [1], the proposed method does not need a server for computation or sharing data. Applying the partitioning method on the target application, we observed that the majority of delay on critical path comes from communication delay. In order to reduce the data transfer further, we propose a communication graph to minimize the number of data links followed by computation replication based on Min-cut Max-flow Theorem. Replication has been researched extensively in VLSI CAD partitioning and design automation community (e.g., [18], [19]). The replication comes with the cost of an increase in local computation. In our target application, measurements show that the wireless communication delay is significantly higher than computational delay and hence, the computational delay overhead is negligible. The proposed min-cut replication results in significant reduction of UKF data transfer among the mobile nodes. The experimental results show that the End-to-End execution time of decentralized UKF prediction and UKF update steps are faster by up to 12.29 times and 3.57 times, respectively, compared to partially decentralized method [1].

II. Background

A. Cooperative Localization

Localization is one of the basic applications in mobile robots that provides other applications such as navigation with location information. The fast and accurate localization is important because a delayed estimation will mislead navigation and other applications and might lead to mission

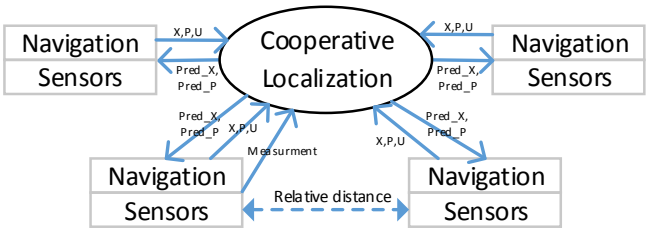


Figure 1: Cooperative localization scheme.

failure. The time interval between execution of localization (T) should be small enough to be able to capture the motion of the robot and provide the higher level applications with accurate and almost real time location. In the GPS-denied environment, because of accumulated proprioceptive sensors error, the estimated location drifts from the actual location after a while.

Cooperative Localization (CL) is a scheme in which robots improve their localization accuracy by jointly processing inter-robot distance measurements obtained by exteroceptive sensors such as Kinect. In CL, a single measurement between two robots can improve the localization accuracy of other robots as well. For CL using Kalman filter and its variants such as EKF and UKF, as it is shown in Figure 1, robots share their control signals (U), state vectors (X), their uncertainty (P), and sensors measurements to obtain more accurate location ($Pred\_X$) and uncertainty of estimated location ($Pred\_P$) for further processing by higher level applications such as navigation. In the context of CL, the state vector (X) includes the global pose (position and orientation of the robots) as well as possibly other states potentially needed to model the dynamics of the robots (for example, steering angle and actuation). In a networked system of N robots, state vector of each robot i is represented by n_i state variables, which adds up to n states of the system (X).

B. UKF

UKF is a recursive filter for estimating the state of a system referred to as X. UKF is composed of prediction and update steps. Prediction step runs periodically, and update step runs whenever there is a measurement in the system. In the prediction step, the state estimation of the system will be predicted based on the control signals, self motion measurements, the previous state of the system, and covariance matrix. Update step runs whenever the system receives a measurement.

*1) **Prediction Step**:* The prediction step for the collective system with n states starts with computing the square root matrix, as a triangular matrix, of matrix P using Cholesky Decomposition (CD) method. After that, a set of $2n + 1$ sample points, called Sigma Points (SP), is generated by eq. 1b. In the equations, (c) denotes the c^{th} column of the matrix. The system model function will use SPs and U to

generate Transformed Sigma Points (eq. 1c). The predicted state is the weighted arithmetic mean of T_SPs (eq.1d). The prediction covariance matrix ($Pred\_P$) will be obtained by eq. 1f using prediction error E (eq.1e). In the equations, $l \in \{0, \cdots, 2n\}$, $c \in \{1, \cdots, n\}$, and w are system defined constant.

$$CD = CholeskyDecomposition(P) \tag{1a}$$
$$SP_{(0)} = x,\ SP_{(c,c+n)} = x \pm (\sqrt{(n+\kappa)})CD_{(c)} \tag{1b}$$
$$T\_SP_{(l)} = SystemModel(SP_{(l)}, U) \tag{1c}$$
$$Pred\_X = \sum_{i=0}^{2n} w_{(i)}\, T\_SP_{(i)}, \tag{1d}$$
$$E_{(l)} = T\_SP_{(l)} - Pred\_X \tag{1e}$$
$$Pred\_P = \sum_{i=0}^{2n} w_{(i)}\, E_{(i)} E_{(i)}^{\top} \tag{1f}$$

*2) **Update Step**:* Update step runs whenever the system receives a measurement ($Meas$). For example, when robot A measures the relative distance from robot B. UKF update function fuses the measurement feedback with the predicted state, computed by prediction step, using Kalman gain. In update step, at first, innovation covariance (S), cross-covariance (P_{xz}), E_z, and innovation (r) will be generated by eq. 2 using the SPs.

$$T\_SP\_meas = MeasurementModel(SP) \tag{2a}$$
$$Pred\_Meas = \sum_{l=0}^{2n} w_{(l)} T\_SP\_meas \tag{2b}$$
$$r = Meas - Pred\_Meas \tag{2c}$$
$$E_{z,(l)} = T\_SP\_meas(l) - Pred\_Meas \tag{2d}$$
$$S = \sum_{i=0}^{2n} w_{(i)}\, E_{z,(i)} E_{z,(i)}^{\top} \tag{2e}$$

After that, the Kalman gain (K) will be computed using S and P_{xz} by eq. 3.

$$P_{xz} = \sum_{l=0}^{2n} w_{(l)}\, E_{(l)} E_{z,(l)}^{\top} \tag{3a}$$
$$K = P_{xz}\, S^{-1} \tag{3b}$$

Finally, the $Update\_X$ and $Update\_P$ will be computed by eq. 4.

$$Update\_X = Pred\_X + Kr \tag{4a}$$
$$Update\_P = Pred\_P - KSK^{\top} \tag{4b}$$

$Pred\_X$ and $Pred\_P$ (the output of prediction step), or $Update\_X$ and $Update\_P$ (the output of update step) will be used as X and P for the next iteration of UKF.

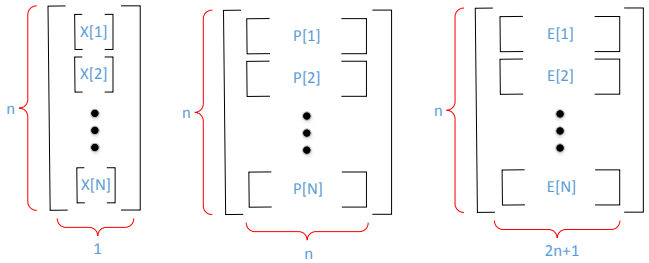


Figure 2: left: Vector X, middle: matrix P, right: matrix E and their sub matrices

C. UKF in Cooperative Localization

In this paper, we denote the components of the aggregated state vector X of the team corresponding to robot i by $X[i]$. In a similar way, the corresponding portion of P and E matrices for robot i is denoted by $P[i]$ and $E[i]$. Note that each of these sub matrices has n_i rows, which is the number of states of robot i. Figure 2 shows the shape and size of the main matrices[1].

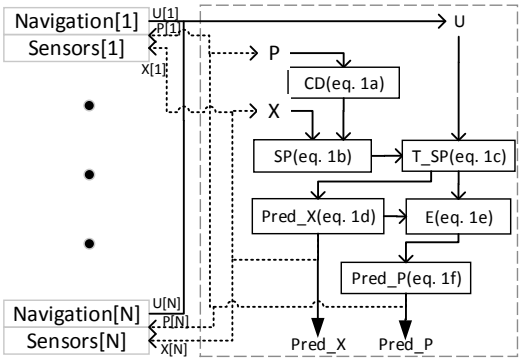
Figure 3 shows the task graphs for UKF prediction and update steps, tagged with the equation numbers, where each robot i sends its control signals ($U[i]$) to the UKF to compute the T_SPs using eq. 1c. At the end, the $Pred\_X$ and $Pred\_P$ might be used as the X and P of the next iteration of UKF. The navigation and other applications of robot i will use $Pred\_X[i]$ and $Pred\_P[i]$. In the update step, the measured relative distance will be used to compute eq. 2. The outputs, which are $Update\_X$ and $Update\_P$, will be used as the X and P of the next iteration of UKF, and also will be sent to robots. The UKF update step runs whenever there is a relative distance measurement in the system.

III. Proposed Framework for UKF Decentralization

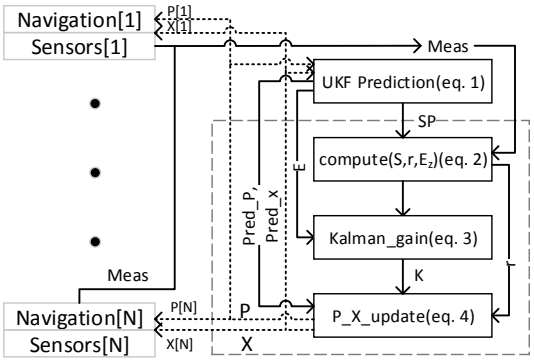
To increase the scalability and avoid a single point of failure, it is necessary to decoupled UKF equations in a way that the UKF computation is distributed and each robot i computes $P[i]$ and $X[i]$, which are its own location parameters. Due to sequential data dependency between the robots to compute CD and $Pred\_P$ (see Figure 4 and [20]), all the robots are actively involved in computing UKF and high amount of data transfer through wireless communication would increase the execution time of UKF. According to [1], all the computations of prediction step of robots are independent, except for the computation of CD and P (eq. 1a and 1f).

To fully decentralize the UKF, at design time, we partition the task graph into a set of subgraphs. Each subgraph is computed locally at the designated robot. In order to provide an effective decentralization, we need to perform optimizations to reduce the data transfer among the robots.

[1] $Pred\_X$ and $Update\_X$ are as the same size of X. Similarly, $|Pred\_P| = |Update\_X| = |P|$, and $|SP| = |T\_SP| = |E|$.



a) UKF Prediction



b) UKF Update

Figure 3: UKF Task Graphs: a) prediction step and, b) update step.

In this paper, our objective is to obtain a fully decentralized UKF with reduced data links and transferred data size, while keeping the computation of UKF decentralized. In addition, we proposed a method that further reduces the data transfer between robots by replicating the computations. Note that the computation in our method is identical to the computation in centralized UKF, so the accuracy of the filter will not be affected.

A. Row-based UKF Partitioning (R-UKF):

As mentioned in the previous section, each robot i needs $X[i]$ and $P[i]$ [2]. The control signals ($U[i]$) is generated locally. The UKF equations can be decoupled in a way that each robot just computes what it needs. For example, robot i just needs to compute $SP[i]$. There is data dependency between computations of some sub matrices, hence each robot has to acquire them from other robots. Based on eq. 1 and [20], in the prediction step, all computations can be completely decoupled, except for CD and $Pred\_P$. $CD[i]$ depends on $CD[1 : i-1]$, and $Pred\_P[i]$ depends on $E[1 : i-1]$, which has to be transferred from other robots.

[2]Recall that we represent the corresponding n_i number of rows in X, i.e. from row n_i*i to n_i*(i+1), with $X[i]$. To show the corresponding rows of X from ith robot to jth robot, we will use $X[i : j]$. The same representation is used for other matrices as well.

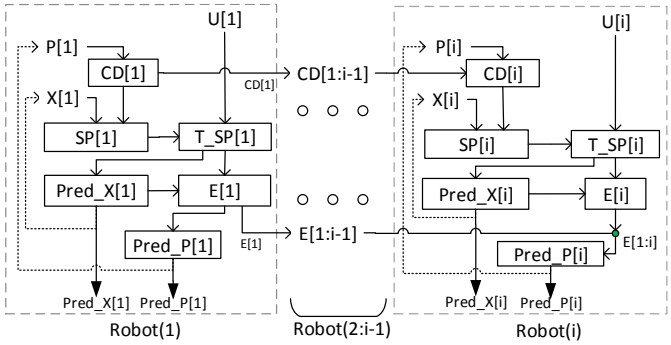


Figure 4: Row-based decentralized (R-UKF) prediction step

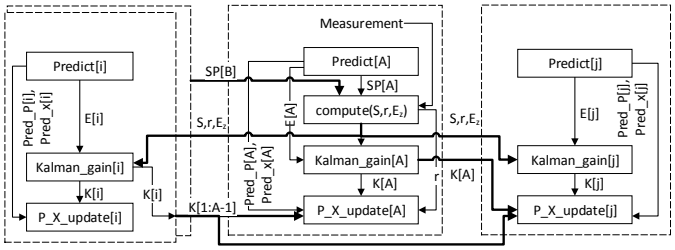


Figure 5: Row-based decentralized UKF (R-UKF) update step

Hence, each robot i after receiving $CD[1 : i-1]$ and $E[1 : i-1]$, computes the $CD[i]$, $SP[i]$, $P[i]$, $T\_SP[i]$, $Pred\_X[i]$, $Pred\_P[i]$, and $E[i]$. Figure 4 shows the proposed UKF prediction task graph. The task graph is divided into sub-graphs associated with each robot i.

In our framework, this is referred to as *Row-based* UKF partitioning. This partitioning will reduce the size of data transfer among the robots. Note that the overall computation and result will be identical to the centralized UKF. Unlike [1], this partitioning does not need a server for computation or sharing data. The UKF prediction task graph shows that estimation of the location of robots is highly correlated and dependent on each other. As a result, the computation is sequential and parallelism is limited.

Figure 5 shows UKF update task graph. Each pair of robots, e.g. robot A and robot B can occasionally measure the relative distance between each other. Robot A that has the relative distance measurement will receive $SP[B]$ from robot B. Then, using $SP[A]$, $SP[B]$, and measurement data, robot A will compute S, r, and E_z. After that, starting from robot 1, each robot j will compute $K[j]$ using $E[j]$, which is generated locally, and using S, r, and E_z. In addition, Robot j receives $K[1 : j-1]$ from robot $j-1$ to compute $Update\_P[j]$ and $Update\_X[j]$. $K[1 : j]$ will be sent to robot $j+1$ for computing the relative UKF update step. Note that R-UKF has exactly the same computation of a centralized UKF for CL.

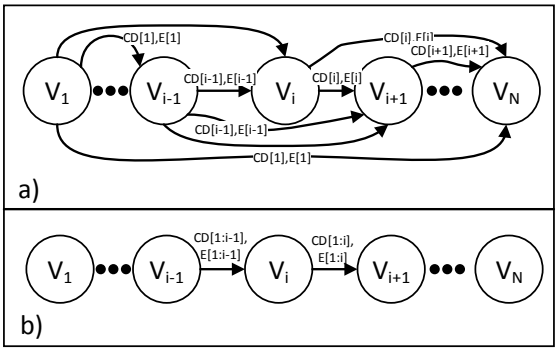


Figure 6: a) Decentralized R-UKF data dependency, b) communication graph

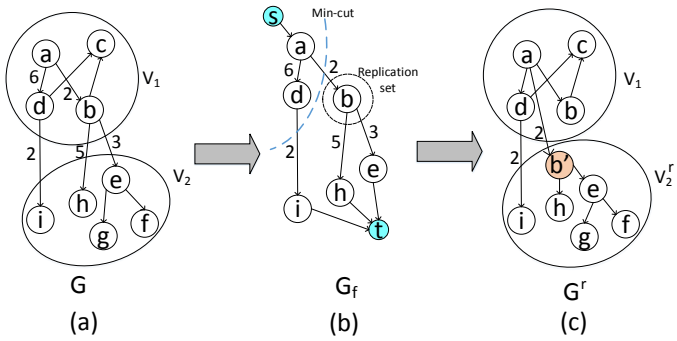


Figure 7: An example of Min-cut Replication a) Original graph, b) Min-cut and replication set, c) Replicated graph

B. Communication Graph Refinement

Based on eq. 1, submatrices such as $CD[i]$ and $E[i]$ that are computed in robot i have to be transferred to all robots with label from $i+1$ to N. We represent the N-way partitioned UKF task graph by $G=(V,L)$ where $V=V_1,V_2,\cdots,V_N$. V_i is a set of UKF computation nodes in V that belongs to robot i, and L represents the data dependency between UKF computation nodes. Figure 6.a shows the data dependency between the nodes. Sending $CD[i]$ and $E[i]$ from V_i *directly* to all other agents with greater label (from $i+1$ to N) requires a complete communication graph, hence $N*(N-1)/2$ communication links will be required which impose a large computation and communication overhead. Note that establishing a TCP/IP connection includes handshaking and several packet transmission, and broadcasting protocols such as UDP cannot be used because they are unreliable. Since each V_i cannot finish the computation of $CD[i]$ before receiving $CD[1:i-1]$, it has to wait till V_{i-1} finish the computation of $CD[i-1]$, and so on. Therefore the computation of $CD[i]$ is sequential, and its critical path starts from V_1 and ends in V_i. Considering the fact that V_i has $CD[1:i-2]$ when it computes $CD[i-1]$, it is possible to send $CD[1:i-1]$ from V_{i-1} to V_i. This way, communication links between V_1 to V_{i-2} and V_i are not needed (see Figure 6.b). Reducing communication links will reduce the communication overhead. Note that in the case of UKF, the computation delay is small compared to communication delay and the time overhead of establishing and maintaining a TCP/IP link. The number of communication links in the proposed communication model is $N-1$ which is the minimum possible number of communication links for a connected network. Note that the data transfer between the nodes can be realized through ad-hoc or infrastructure network. This discussion is out of the scope of this paper.

The proper partitioning and communication graph reduce the communication cost, yet large matrices such as CD and E have to be transferred. Therefore, we propose to deploy a graph-based partitioning refinement technique called min-cut replication to further reduce the size of transferred data.

C. Communication Minimization by Computation Replication on R-UKF (RR-UKF):

The partitioning, stand alone, cannot reduce the data communication significantly. To further reduce the size of data transfer between partitions, we replicate some of the computations. Figure 7(a) shows data links between two nodes V_1 and V_2. The size of data that has to be transferred between V_1 and V_2 is 10. By replicating node b in partition V_2 (Figure 7(c)), the size of data transfer will be reduce to 4 (2+2). To obtain UKF task graph with minimum data transfer, we propose a two phases method. First, Min-cut Max-flow replication is applied to prediction step of UKF, which generates a task graph with replicated nodes. Second, UKF update task graph is pruned given the changes in UKF prediction task graph after replication. Then Min-cut Max-flow replication is applied to the pruned UKF update task graph. The result is a fully decentralized UKF with minimum data transfer, and hence, minimum End-to-End delay.

Replication technique: In the proposed method, we present how to minimize the data transfer using min-cut replication technique on decentralized prediction and update steps of UKF for CL. We are given a N-way partition of $G=(V,L)$ represented by $V=V_1,V_2,\cdots,V_N$. V_i is a set of nodes in V that belongs to partition i, and L represents the data dependency between nodes. The set of cut edges is the set of edges $C\in L$ that connects the partitions. Lets consider two partitions V_1 and V_2 as shown in Figure 7. Lets assume that I is the set of incoming edges to set V_2 from V_1. The objective is to find a subset of nodes in V_1 called *replication set* R_1 such that if it is replicated in V_2, the cutsize between V_1 and V_2 is minimized. After replication of R_1 in V_2, the cut edges between R_1 and V_2 are eliminated from the cut set and the input edges to R_1 are added to cutsize. R_1 is a minimum replication set respect to V_2 if the cutsize after replication is minimum among all possible R_1.

Figure 7 shows an example of G (a), min-cut and replication set (b), and the replicated graph (c). To find the minimum replication set, we construct a network graph $G_f=(V_f,L_f)$, where V_f includes nodes in V_1 that are reachable from I (node a, b, and d) and the nodes in V_2

that are adjacent to I (node i, h, and e). To apply min-cut max-flow theorem, two dummy nodes are added to G_f as source (node s) and sink (node t). The min-cut max-flow problem on this graph will separate the replication set from the rest of the network. The replication set R_1 is the subset of nodes in V_1 starting from min-cut edges to the nodes adjacent to I (node b). After replication of R_1 in V_2, the cut edges between R_1 and V_2 (edge $b \rightarrow h$, and $b \rightarrow e$) are eliminated from the cut set and the input edges to R_1 are added to cutsize (edge $a \rightarrow b'$). The replication in V_2 in respect to set V_1 does not affect the edges between V_1 and other partitions. Hence, we can apply the min-cut replication between every two partitions in a N-way partitioned task graph independently.

To find the minimum weighted cut, min-cut max-flow replication considers weight or capacity for the edges. In N-way partition of UKF, the edges are weighted based on the data size and the wireless communication cost (e.g., number of hops). The pseudo code for min-cut replication is presented in algorithm 1. Next, we present replication algorithm for UKF prediction and UKF update step.

Algorithm 1: Min-cut Max-flow Replication

input : V_i, V_j, C_{ij}
output: V_j^r

initialization:$V_j^r = V_j$;
Construct Graph $G_f(V_i, V_j)$;
$|C| \leftarrow$ min-cut max-flow on G_f;
if $|C| < |C_{ij}|$ **then**
  $V_j^r = V_j^r \cup R_j$
end
Return V_j^r;

We present our algorithm based on aforementioned min-cut replication and show how the cut edge set between the N-way partitions of UKF is drastically reduced both in prediction and UKF update. The pseudo code of RR-UKF is presented in Algorithm 2. The input to the algorithm in sequential row-based N-way partition of UKF, known as R-UKF. First, Min-cut Max-flow replication is applied on each pair of partitions in decentralized UKF prediction (line [3-6]). Then UKF update with respect to measurement between V_a and V_b is considered. The refinement of decentralized UKF update is decomposed of two steps. The first step is referred to as Prune-cut-edges (line [7]). Due to replication in UKF prediction steps, some edges from cutsets between the partitions are moved inside the partitions. If those edges are deployed in the corresponding partition of update graph, the edges are local and hence, those edges from the cut set are removed. Visiting the cut edges in UKF update graph, this function checks if any of them is removed in V_p^r after replication. The second step is to apply Min-cut Max-flow replication between the N-way partitions of UKF update similar to UKF prediction (line [8-11]).

Algorithm 2: RR-UKF-Replication

input : N-way partition of UKF prediction Task Graph $V_p = V_{p1}, V_{p2}, V_{p3}, \cdots, V_{pk}$ and Update(a,b) Task Graph $V_u(a,b) = V_{u1}, V_{u2}, \cdots, V_{uk}$
output: V_p^r and V_u^r

//Initialization;
$V_p^r = V_p$, $V_p^r = V_p$, $V_u^r = V_u$
//========== UKF prediction;
for *each* (V_{pi}, V_{pj}) *where* $C_{ij} \neq \emptyset$ **do**
  $V_{pj}^r \leftarrow$ Min-cut Max-flow replication (V_{pi}, V_{pj})
end
$V_u \leftarrow$ Prune-cut-edges (V_p^r, V_u)
//========== UKF update;
for *each* V_i *and* V_i *on* V_{update} **do**
  $V_{uj}^r \leftarrow$ Min-cut Max-flow replication (V_{ui}, V_{uj})
end
Return (V_p^r and V_u^r)

Figure 8 demonstrates the decentralized UKF prediction with min-cut replication. The minimum weight cut is shown in the figure. The result shows that by replicating $SP[1]$, $T\_SP[1]$, $Pred\_X[1]$, $E[1]$, $Pred\_P[1], \cdots, SP[i-1]$, $T\_SP[i-1]$, $Pred\_X[i-1]$, $E[i-1]$,$Pred\_P[i-1]$, in V_i, the cut size will be minimized. As a result, each partition v_i, after receiving $U[1:i-1]$ from previous partition, will compute $CD[1:i]$. Instead of getting $E[1:i-1]$ and $CD[1:i-1]$ from previous partition, it will calculate them using $U[1:i-1]$ (new cut edge) and the $Pred\_P[1:i]$ and $Pred\_X[1:i]$ from previous step of UKF which are located in the same partition. The size of $U[1:i-1]$ is much smaller compared to $E[1:i-1]$ and $CD[1:i-1]$.

Figure 9 shows the partition between partitions. Partition V_A has the $SP[B]$ and $E[1:A]$ as a result of replication in UKF prediction V_p^r. Hence, S ,r, and also $K[1:A]$ are generated (refer to eq. 2 and 3). Partition V_A will send $K[1:A-1]$ to partitions V_{ui} where $i < A$. Partition A will send only relative distance measurement between robots A and B, to partitions V_{ui} when $i > A$. Since $SP[A]$ and $SP[B]$ already exist in Vp^r, all other parameters are locally computed and hence those edges from cutset are removed.

IV. Experiments

A. Experimental setup

To evaluate the proposed method, we implemented it in a network of single-board computers, called Raspberry Pi 3 B, with quad-core 1.2GHz Broadcom BCM2837 64bit CPU, 1 GB main memory, and a built-in WiFi module of 802.11 b/g/n. For these experiences, the CPU frequency

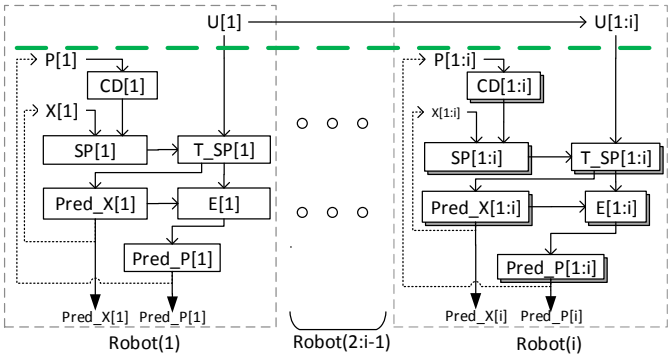


Figure 8: UKF prediction step after min-cut replication (RR-UKF)

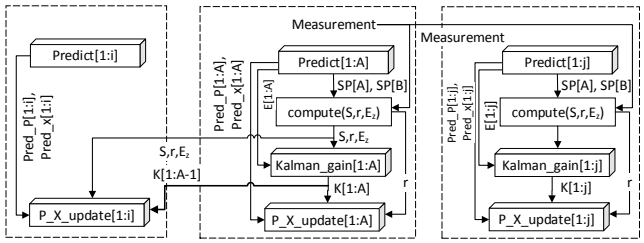


Figure 9: UKF update step after min-cut replication (RR-UKF)

is fixed to 1.2GHz. Each board i runs Linux kernel v4.9 and an application process that implements robot i's task for decentralized UKF in C++ with an open-source linear algebra library EIGEN [21]. To create a realistic network environment, we set up an isolated WiFi network with one access point, Netgear N300 wireless router and N boards. In the following, we report a median value for the End-to-End delay which is obtained from 500 rounds of UKF.

In our experimental studies, we considered robotic team scenarios with robots with 6 local states and with measurement and control signals size of 3. In our implementation, the main performance metric is the End-to-End delay from robot 1's initiation to robot N's completion of each execution cycle of UKF, consisting of the prediction and the update steps, which include both the computation and the communication delays on all the robots (or boards).

B. Evaluation of UKF End-to-End Delay

To demonstrate the effectiveness of our decentralized UKF over the partially decentralized UKF [1], denoted by Partially Decentralized UKF, we compare them in Table I. Because of lack of space, in the tables, just the entries for the odd number of robots has shown. Recall that the Partially Decentralized UKF [1] assumes that there exists a server to store the whole data of UKF algorithm and that for each UKF cycle all robots takes part in the computation of UKF, like RR-UKF. The number of robots, N, is varied from 3 to 15. From the table, it is clear that on average, the End-to-End delay measured for RR-UKF is reduced by a factor of up to 12.29 for prediction step and by a factor of 3.57 for update step, compared to Partially Decentralized UKF [1].

Table I: End-to-end Delay Comparison between Partially Decentralized UKF [1] and RR-UKF

UKF End-to-End delay measured at application level (msec)						
	UKF Prediction			UKF Update		
N	[1]	RR-UKF	ratio	[1]	RR-UKF	ratio
3	20.32	10.18	2.00	17.36	9.84	1.76
5	56.14	11.29	4.97	36.78	20.26	1.82
7	130.51	24.53	5.32	59.61	22.86	2.61
9	282.37	34.65	8.15	85.65	24.81	3.45
11	529.19	54.03	9.79	115.37	32.29	3.57
13	847.29	82.44	10.28	145.27	43.16	3.37
15	1322.86	107.61	12.29	186.21	54.44	3.42

Table II: Data Communication Comparison between R-UKF and RR-UKF

	UKF Prediction			UKF Update		
N	R-UKF	RR-UKF	ratio	R-UKF	RR-UKF	ratio
Total data bytes transmitted by all robots at application level						
3	6120	216	28.33	4174	575	7.26
5	33840	720	47.00	10606	1727	6.14
7	99288	1512	65.67	19918	3455	5.76
9	218592	2592	84.33	32110	5759	5.58
11	407880	3960	103.00	47182	8639	5.46
13	683280	5616	121.67	65134	12095	5.39
15	1060920	7560	140.33	85966	16127	5.33
Total packet frames transmitted by all robots at kernel level						
3	6	3	2.00	5	3	1.67
5	26	5	5.20	10	5	2.00
7	70	7	10.00	16	7	2.29
9	151	9	16.78	26	9	2.89
11	278	11	25.27	36	12	3.00
13	462	13	35.54	51	16	3.19
15	715	15	47.67	66	20	3.30
End-to-End delay measured at application level (in millisecond)						
3	12.25	10.18	1.20	10.69	9.84	1.09
5	27.78	11.29	2.46	23.57	20.26	1.16
7	73.87	24.53	3.01	32.70	22.86	1.43
9	132.46	34.65	3.82	45.76	24.81	1.84
11	243.11	54.03	4.50	61.55	32.29	1.91
13	396.43	82.44	4.81	74.45	43.16	1.73
15	636.12	107.61	5.91	92.02	54.44	1.69

To indicate the effect of reducing the size of transferred data using the replication technique in decentralized UKF, we compare our row-based decentralized UKF implementation without (R-UKF) and with computation replication (RR-UKF). Table II shows the total network traffic and the on board End-to-End delay measured for each UKF iteration in R-UKF and RR-UKF. Total data bytes transmitted by all robots at application level in RR-UKF have been drastically reduced up to 140 times for prediction step and 7.26 times for update step in comparison with R-UKF.

Computation replication eliminates transmission of CD, E and E_Z between robots and replaces with $Measurment$ and U in the transmission which are much smaller. Thus, it leads to significant reduction in the total packet frames transmitted by all robots at kernel level. The reduction ratio is up to 47.67 for prediction step and to 3.3 for update step.

The reason why this reduction ratio is smaller than that of the transmitted data bytes is that every packet frame, whose maximum size is 1500 bytes in our WiFi network, does not convey the same number of data bytes depending on the network I/O behavior of the application. In addition, the replicated computations increases the run time. As a result, the End-to-End delay from robot 1 to robot N required for a whole UKF cycle is reduced from 636 ms (R-UKF) to 107 ms (RR-UKF) for prediction step when N=15. For update step, it is reduced from 92 ms (R-UKF) to 54 ms (RR-UKF). This significant reduction in the End-to-End delay allows each robot to react much faster in response to the resulting localization data of other robots.

V. Conclusions

Cooperative localization is a method for increasing the localization accuracy within a network of cooperative robots. Usually, probabilistic estimation algorithms such as EKF is used for processing the data shared by the group. UKF is a variant of Kalman filter, which is more accurate but has more computation, and decentralization of such filters with tight correlation among its tasks requires a large data transfer between agents. We showed that the size of the transferred data can be reduced by applying a replication technique. Our experimental results showed that the End-to-End execution time of the decentralized UKF prediction and update steps with replication are faster by up to 12.29 and 3.57 times compared to the partially decentralized UKF algorithm of [1]. In future, we plan to extend our work to include the delay and computational complexities of distance measurement sensors as well as to consider probabilistic estimation algorithms other than UKF.

Acknowledgement

Solmaz Kia's work was supported by NIST award 70NANB17H192. The authors would like to thank Jeong Dam Hwang from Kookmin University for her help in this project during her internship at UC Irvine.

References

[1] V. Dinh and S. S. Kia, "A server-client based distributed processing for an unscented kalman filter for cooperative localization," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2015 IEEE International Conference on*, pp. 43–48, IEEE, 2015.

[2] S. I. Roumeliotis, *Robust mobile robot localization: from single-robot uncertainties to multi-robot interdependencies*. PhD thesis, University of Southern California, 2000.

[3] A. Howard, M. J. Matark, and G. S. Sukhatme, "Localization for mobile robot teams using maximum likelihood estimation," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 1, pp. 434–439, IEEE, 2002.

[4] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 1402–1409, IEEE, 2009.

[5] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, no. 3, pp. 325–344, 2000.

[6] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Putting the'i'in'team': An ego-centric approach to cooperative localization," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 1, pp. 868–874, IEEE, 2003.

[7] A. Prorok and A. Martinoli, "A reciprocal sampling algorithm for lightweight distributed multi-robot localization," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 3241–3247, IEEE, 2011.

[8] R. Liu, C. Yuen, T.-N. Do, D. Jiao, X. Liu, and U.-X. Tan, "Cooperative relative positioning of mobile users by fusing IMU inertial and UWB ranging information," in *2017 IEEE International Conference onRobotics and Automation (ICRA)*, pp. 5623–5629, IEEE, 2017.

[9] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *AeroSense: the 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, pp. 182–193, 1997.

[10] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, 2002.

[11] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localisation using interlaced extended kalman filter," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 2816–2821, IEEE, 2006.

[12] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 519–524, IEEE, 2006.

[13] S. S. Kia, S. Rounds, and S. Martinez, "Cooperative localization for mobile agents: a recursive decentralized algorithm based on kalman-filter decoupling," *IEEE Control Systems*, vol. 36, no. 2, pp. 86–101, 2016.

[14] L. Luft, T. Schubert, S. I. Roumeliotis, and W. Burgard, "Recursive decentralized collaborative localization for sparsely communicating robots.," in *Robotics: Science and Systems*, 2016.

[15] P. S. Maybeck, *Stochastic models, estimation, and control*, vol. 3. Academic press, 1982.

[16] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[17] S. S. Kia, J. Hechtbauer, D. Gogokhiya, and S. Martinez, "Server assisted distributed cooperative localization over unreliable communication links," *https://arxiv.org/abs/1608.00609*, 2017.

[18] L. J. Hwang and A. El Gamal, "Min-cut replication in partitioned networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 1, pp. 96–106, 1995.

[19] V. Yazici and C. Aykanat, "Constrained min-cut replication for k-way hypergraph partitioning," *INFORMS Journal on Computing*, vol. 26, no. 2, pp. 303–320, 2013.

[20] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Communication-optimal parallel and sequential cholesky decomposition," *SIAM Journal on Scientific Computing*, vol. 32, no. 6, pp. 3495–3523, 2010.

[21] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3." http://eigen.tuxfamily.org, 2010.