Multi-Robot Cooperative Localization Through Trust Based Sensor

ROB 530 Final Project: Team 15

Cameron Kabacinski
camkab@umich.edu

Huashu Li
lihs@umich.edu

Gregor Limstrom
limstrom@umich.edu

***Abstract*—Multi-robot localization is an important challenge to solve as urban autonomous systems grow in popularity and implementation. In this paper, we explore the use of an Extended Kalman Filter, a Particle Filter, and an Unscented Kalman Filter in tandem with a time-since-last-measurement based trust system for a network of robots communicating through a single central node, attempting to localize using the UTIAS Multi-Robot Localization and Mapping data set. Project source can be found at [4] and a video presentation at [5].**

I. Introduction

In order to successfully accomplish their missions, autonomous mobile robots are required to have precise and continuous knowledge of their position and orientation. In 2D space, this is usually accomplished by fusing state sensors, such as wheel encoders and inertial measurement unit data, with reference data from the outside world such as landmarks. This allows the robot to correct for the inherent drift in state sensors with known positions.

The majority of current robot localization methods developed for mobile applications are designed for a single robot. These single robot localization approaches are not optimized for multi-robot systems, and adapting them for a network of robots is an active research area.

Relative observations are an especially relevant issue that must be solved. To define, a relative observation is a range and bearing reading of one robot by another. Traditionally, when localizing within a known map, a robot can pinpoint its location by comparing the landmark reading to a known position. However, with relative readings, a robot must compare its own position to the estimated position of another robot, which has varying degrees of uncertainty depending on the state of the robot observed.

In reference [1], an Extended Kalman Filter (EKF) is used to fuse robot state and measurement data for estimation. The filter is decentralized, so the network state is decomposed into individual robot filters that communicate the relative observations between each other. In the paper, the method was validated using a three robot network.

In this paper, we explore the implementation and effects of a relative observation trust metric on the effectiveness of three different filters: an EKF, a Particle Filter (PF), and an Unscented Kalman Filter (UKF). We believe that this is an important metric to consider, especially in environments that may be landmark sparse, which will lead to the robots relying heavily on relative observations to maintain localization. Section II will cover a preliminary overview of the filter algorithms that we will be using. Section III will cover out methodology and system network implementations. Sections IV will cover the results, and discussion, conclusions, and future work will be covered in section V.

II. Preliminaries

In this section, we will cover the important concepts of our implementation, including the algorithm and update steps of each filter, as well as their advantages and disadvantages.

A. UTIAS Data Set

To investigate our problem of multi robot cooperative localization with real data, we found the University of Toronto Institute for Aerospace Studies (UTIAS) Multi Robot Collaborative Localization and Mapping data set [3].

This data set is a collection of nine 2d indoor data sets, where each data set contains 15 landmarks and 5 robots with odometry, camera, and ground truth measurements. The nine data sets are called "MRCLAM#" where "#" is the number of the data set. The robots and landmarks are uniquely marked with bar codes on vertical cylindrical tubes for identification. The odometry data is given as time stamped forward and angular velocities for each two-wheel drive robot, and the camera measurements are given in the form range, bearing, and bar code measurements. The data for each robot is synchronized and the odometry data occurs every 0.02 seconds while the camera measurement data occurs irregularly at no fixed time interval as it depends on each robots perspective.

B. Motion and Measurement Models

The EKF and PF use the motion model described in II-B1 and the measurement models described in II-B3. The UKF uses a different motion model described in II-B2 and the measurement models remain the same with EKF and PF.

1) Velocity Motion Model 1: We use the velocity motion model from Chapter 5 of Sebastian Thrun's Probabilistic Robotics book [2]. The discrete dynamical equations for a robot given an input of linear and angular velocities (v, ω) are described in equations (1), (2), (3). A third input γ is

considered to resolve a degeneracy issue described in [2] but is always commanded to be zero.

$$x_{k+1} = x_k - \frac{\hat{v}}{\hat{\omega}} sin(\theta_k) + \frac{\hat{v}}{\hat{\omega}} sin(\theta_k + \hat{\omega}\Delta t) \tag{1}$$

$$y_{k+1} = y_k + \frac{\hat{v}}{\hat{\omega}} cos(\theta_k) - \frac{\hat{v}}{\hat{\omega}} cos(\theta_k + \hat{\omega}\Delta t) \tag{2}$$

$$\theta_{k+1} = \theta_k + \hat{\omega}\Delta t + \hat{\gamma}\Delta t \tag{3}$$

The motion noise terms are indicated in (4),(5),(6) where $\epsilon_v, \epsilon_\omega, \epsilon_\gamma$ are uncorrelated zero mean Gaussian noises and $\alpha_1, ..., \alpha_6$ are robot-specific error parameters.

$$\hat{v} = v + \epsilon_v, \quad \epsilon_v \sim \mathcal{N}(0, \alpha_1 v^2 + \alpha_2 \omega^2) \tag{4}$$

$$\hat{\omega} = \omega + \epsilon_\omega, \quad \epsilon_\omega \sim \mathcal{N}(0, \alpha_3 v^2 + \alpha_4 \omega^2) \tag{5}$$

$$\hat{\gamma} = \epsilon_\gamma, \quad \epsilon_\gamma \sim \mathcal{N}(0, \alpha_5 v^2 + \alpha_6 \omega^2) \tag{6}$$

2) Velocity Motion Model 2: It is a simple motion update model using the odometry information and previous pose. The model assumes the robot rotates in place and drives in a straight line. The model will be accurate enough with a small Δt.

$$x_{k+1} = x_k + \hat{v}\Delta t \text{cos}(\theta_k + \frac{\hat{\omega}\Delta t}{2}) \tag{7}$$

$$y_{k+1} = y_k + \hat{v}\Delta t \text{sin}(\theta_k + \frac{\hat{\omega}\Delta t}{2}) \tag{8}$$

$$\theta_{k+1} = \theta_k + \hat{\omega}\Delta t \tag{9}$$

The motion noise terms of this model are the same terms (4) and (5) in II-B1.

3) Stacked Measurement Model: The measurement model is shown in equation (10), where (m_x, m_y) is the position of observed landmark or robot, and the top and bottom row is the bearing and range, respectively.

$$z_k = \begin{bmatrix} \text{atan2}(m_y - y_k, m_x - x_k) - \theta_k \\ \sqrt{(m_y - y_k)^2 + (m_x - x_k)^2} \end{bmatrix} + r_k \tag{10}$$

$$r_k \sim \mathcal{N}(0, R_k) \tag{11}$$

To properly estimate the pose of the robot, at least 2 measurements are required in filter correction steps to fully observe a robots pose in 2d and to avoid degeneracy issues. The stacked measurements are presented in (12).

$$z = \begin{bmatrix} z_1 \\ z_2 \\ ... \\ z_n \end{bmatrix} \tag{12}$$

4) Jacobians: The EKF requires Jacobians of the motion and measurement model. The Jacobians are computed as specified in equation (13).

$$F_k = \frac{\partial f}{\partial x}\bigg|_{x=\mu_{k-1}}, W_k = \frac{\partial f}{\partial w}\bigg|_{x=u_{k-1}}, H_k = \frac{\partial h}{\partial x}\bigg|_{x=\mu_k^-} \tag{13}$$

C. Filters

We describe in detail the EKF, PF, and UKF we used in this section.

1) Extended Kalman Filter: The EKF is an extension of the work on Kalman filters that allows estimation for non linear dynamic systems. While the regular Kalman filter is an optimal linear estimator for linear dynamic system models, it is not effective for many real world applications that involve non linear dynamics. The EKF can handle these non linear systems by using linearization via Taylor expansion around an operating point for the system, and using this linearization instead of the linear model in the Kalman filter framework

The Kalman filter framework creates a state and measurement prediction based on the system process and measurement models, and then performs a correction update to these predictions based on provided measurements. The filter assumes the initial estimate and uncertainty are known and the process and measurement model are excited by zero-mean white Gaussian noise. The Extended Kalman Filter is described in algorithm 1, from Thrun [2], using motion and measurement models and Jacobians as described in section II-B.

Algorithm 1 Extended Kalman Filter

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k

1: $\mu_k^- \leftarrow f(u_k, \mu_{k-1})$
2: $\Sigma_k^- \leftarrow F_k \Sigma_{k-1} F_k^T + W_k Q_k W_k^T$
3: $\nu_k \leftarrow z_k - h(\mu_k^-)$
4: $S_k \leftarrow H_k \Sigma_k^- H_k^T + R_k$
5: $K_k \leftarrow \Sigma_k^- H_k^T S_k^{-1}$
6: $\mu_k \leftarrow \mu_k^- + K_k \nu_k$
7: $\Sigma_k \leftarrow (I - K_k H_k)\Sigma_k^-$
8: **return** μ_k, Σ_k

2) Particle Filter: The term particle filter describes a set of sequential Monte Carlo methods used to solve filtering problems such as localization. These methods work by simulating a large number of potential states, represented as particles, and propagating them through a stochastic process, then re-weighting them using observations to score their accuracy to the observed measurements, as described in algorithm 2.

Particle filters are often used in real world implementations due to their ease of implementation and robustness. The filtering process can be independent of the motion model of the robot, and can be solved with just odometry readings and measurement fusion. No derivatives need to be computed in the process, and particle filters can estimate non-linear systems without much extra effort needed. Particle filters can also resample around high weight particles to recover localization if particle starvation occurs, which is very useful in the our use case with high inconsistency in our measurement frequency.

We chose to implement a particle filter due to the simplicity and robustness it provides. We anticipated the filter to be resilient against bad measurements and odometry drift due to the high number of possibilities it is able to simulate and

Algorithm 2 Particle Filter

procedure INITIALIZATION(G_0, e)
  $t = 0$
  for i =1 to N **do**
    $sample X_0^{(i)} \sim \mathcal{N}(G_0, e)$
  $t = 1$
procedure PREDICTION(X_t, z)
  for i =1 to N **do**
    $X_{t+1}^{(i)} = MotionModel(X_t^{(i)}, v, \omega)$
    if $len(Z) > 1$ **then**
      $W^{(i)} = MeasurementModel(X_t^{(i)}, Z)$
  Normalize Importance Weights
  Compute Number of effective particles (Neff)
  if $Neff < N/5$ **then** $Resampleweights$
procedure STATE UPDATE(X_{t+1})
  $Estimate_{t+1} = WeightedMean(X_{t+1})$
procedure RESAMPLING($X_{t+1}, W^{(i)}$)
  Resample with replacement N particles from X_{t+1} according to importance weights

Algorithm 3 Unscented Kalman Filter

Require: belief mean μ_{k-1}, belief covariance Σ_{k-1}, action u_k, measurement z_k
$X_{k-1} \leftarrow$ compute the set of 2n+1 sigma points using μ_{k-1} and Σ_{k-1}
$\omega^- \leftarrow$ compute the set of 2n+1 weights
$\mu_k^- \leftarrow \sum_{i=0}^{2n} \omega^- f(u_k, \mu_{k-1,i})$
$\Sigma_k^- \leftarrow \sum_{i=0}^{2n} \omega^-(f(u_k, \mu_{k-1,i}) - \mu_k^-)(f(u_k, \mu_{k-1,i}) - \mu_k^-)^T + Q_k$
$X_k^- \leftarrow$ compute the set of 2n+1 sigma points using μ_k^- and Σ_k^-
$\omega \leftarrow$ compute the set of 2n+1 weights
$z_k^- \sum_{i=0}^{2n} \omega_i h(x_{k,i}^-)$
$\nu_k \leftarrow z_k - h(\mu_k^-)$
$S_k \leftarrow \sum_{i=0}^{2n} \omega_i (h(x_{k,i}^-) - z_k^-)(h(x_{k,i}^-) - z_k^-)^T + R_k$
$\Sigma_k^{xz} \leftarrow \sum_{i=0}^{2n} \omega_k^{[i]} (x_{k,i}^- - \mu_k^-)(h(x_{k,i}^-) - z_k^-)^T$
$K_k \leftarrow \Sigma_k^{xz} S_k^{-1}$
$\mu_k \leftarrow \mu_k^- + K_k \nu_k$
$\Sigma_k \leftarrow \Sigma_k^- - K_k S_k K_k^T$
return μ_k, Σ_k

compare. Our particle filter is implemented as described from [2].

3) Unscented Kalman Filter: The Unscented Kalman Filter uses the statistical linearization technique to deal with the nonlinear functions. It is often used to linearize a nonlinear function of a random variable through a linear regression between n points drawn from the prior distribution of the random variable. Like EKF, UKF consists of the same two steps: prediction and correction. In the correction step, the selection of sigma points is added.

The UKF helps us to approximate a probability distribution with arbitrary nonlinear function or transformation more easily. The key segment of UKF is choosing the sigma points. The sigma points are chosen strictly according to their prior distribution. Each sigma point is then propagated through the nonlinearity. The unscented transformation is used to calculate the statistics of a random variable with nonlinear transformation.

The UKF is also implemented as described in [2].

III. METHODOLOGY

In this section, we will cover the simulated network infrastructure of our multi-robot localization, the techniques we implemented to handle challenges associated with measurements, and the design and integration of the trust factor into our system.

A. System Network Infrastructure and Experiment

To simulate our system, we chose to process all the information from the mobile robots through a centralized node. This node is represented via our Matlab script, which facilitates communication of estimates and measurements between all robots on the network. We assume no communication delay and no packet loss in order to simplify our simulation environment.

Each robot's localization is handled by an individual filter, which, at each time step, updates the prediction of the robot's pose, and then, if available, uses observation measurements to correct the prediction and estimate the robot's pose.

When a robot observes another robot, the robot will need the observed robot's estimated pose for use in the measurement model. The robot will request the robot's current estimated pose from the centralized table, and use the range and bearing measurement with this estimated pose to create a measurement that can be used in the correction step of the filter.

1) Infrequent and Single Measurements: Measurements are considered infrequent as we always have odometry data at every time step, but we may not have any new measurements. This could be due the robot not seeing any other robots or landmarks, or just taking a while to process measurements from the camera and make them available. Single measurements are not enough to reliably correct predictions, as we need two measurements to fully observe a robots pose in 2d, described in section II-B3.

To solve these challenges, the filters would simply predict their pose estimate at every time step using odometry data, and otherwise ignore single measurements.

2) Invalid Measurements: In order to deal with non existent bar codes, we simply checked if the bar code matched a valid entry in our table, otherwise we ignored the measurement.

Detecting a misread was much harder. The solution we settled on was to compute the expected position of an observation using the current pose and the range/bearing measurement, and compare that expected position to the given position of the observation based on the bar code id. If the distance between the two positions was greater than a threshold, we considered

that there was a misread.

This method only works if we know the given position of the observed bar code id, so it does not work well for observed robots as they move (and we only have access to their estimated pose and that may have error larger than the threshold check), but does work well for landmark misreads. If a misread occurred with a landmark observation, we would try to correct the misread by checking all the other landmarks and seeing if the range/bearing measurement would agree with the other landmarks position, and using that landmark instead if it was below a certain distance threshold away from the range/bearing measurement.

B. Trust Factor

The trust factor is the key innovation of our project. Through this, we attempt to quantify the accuracy of a robot's own pose estimate from its filter. This trust factor for a single robot will affect other robots as it will determine if other robots are allowed to use the single robot's estimated pose if the single robot thinks its estimate is trustworthy or not. The main benefit of a single robot having an idea of its pose accuracy is it will allow other robots to intelligently choose whether or not to factor into their update loop a measurement of the single robot that may correspond to an untrustworthy pose estimate, which could cause the other robots to incorrectly update their own estimate due to the single robot's bad pose estimate.

There are two main approaches to involving a trust factor. The first approach is to treat trust as a binary decision and the second approach is to consider trust as a continuous gradient. In either case, each time the robot has a successful update correction, it would update its trust factor to be more trustworthy. Otherwise, the trust factor would decrease as a function of time.

In the binary approach, each robot will maintain a binary state that its own estimate of its position is either trustworthy or not trustworthy. When the robot successfully corrects its prediction, the trust factor will reset and become trustworthy. Otherwise, the trust factor will become untrustworthy after a set amount of time. When asked by other robots for its estimate, the asked robot will see if its trust factor is trustworthy or not and correspondingly provide its estimate or let the asking robot know the estimate is not trustworthy. If the estimate is not trustworthy, then the other robot will not be able to update off the first robot's estimated position, as it is deemed invalid.

In the continuous gradient approach, each robot would maintain a value corresponding to its trustworthiness. Then a function would have to be defined for how the value changes to increase trust on successful updates, and for how the value changes to decrease trust over time, such as a decaying function.

In our implementation, we choose to use the binary approach for trust. We choose this approach as it was simpler and less prone to mistakes in implementation and complexity across the different filters we implemented.

We choose a time cutoff of 20 seconds, meaning if the robot went for 20 seconds without any successful measurement updates, it would believe its pose to not be trustworthy. Otherwise it would update its trust factor to be trustworthy any time it successfully updated its correction.

We choose this time value as it takes about 40 seconds for the robot moving without updates (solely relying on motion model predictions) for its pose to drift significantly away from the GT value, so half would keep the robot safely localized.

IV. Experimental Results

In this section we will evaluate the performance of the EKF, PF, and UKF on the UTIAS Data set "MRCLAM1" for time steps 1 to 30,000 (which corresponds to a time of 0 seconds to 600 seconds) in three different scenarios.

The first scenario is to evaluate the filters when they only use observations of landmarks in the correction step. This means they are unable to observe other robots, and can only use the landmark positions in the measurement model.

The second scenario is to evaluate the filters when they are using the trust factor as described in section III-B. The robot will treat landmarks the same as in the first scenario, but now when the robot observes other robots, it will request that robot's estimated pose from the network to use as the position in the measurement model. If the requested robot does not belief its estimate is trustworthy, then it will not provide the pose and the asking robot will ignore that measurement of the untrustworthy robot.

The final scenario is to evaluate the filters when they use the ground truth pose of other robots for their estimated position in the measurement model.

Metrics for evaluating these scenarios are described in section IV-A.

A. Evaluating Metrics

To evaluate each filter, we need to devise a metric for comparison. The metric we decided to use is a distance metric that compares the robot's estimated pose from the filter to the ground truth position of the robot, which is computed at each time step.

We chose this metric as it is easy to compute and gives us a good comparison for how well the filter did compared to the ground truth, which is the ideal result if the filters worked perfectly.

We will compute a root mean square error for the distance to get a single cumulative result of the distance error, according to equation (14), and compute the standard deviation of the distances from each time step, so we can understand the spread of distance error, which will tell us how far away the robot was from the ground truth during the experiment in general.

$$\text{distanceRMSE} = \sqrt{\frac{\sum_{k=1}^{\text{numTimeSteps}} \text{dist}_k^2}{\text{numTimeSteps}}} \tag{14}$$

B. Landmarks Only

The results for this approach is presented in figure 1 and table I.

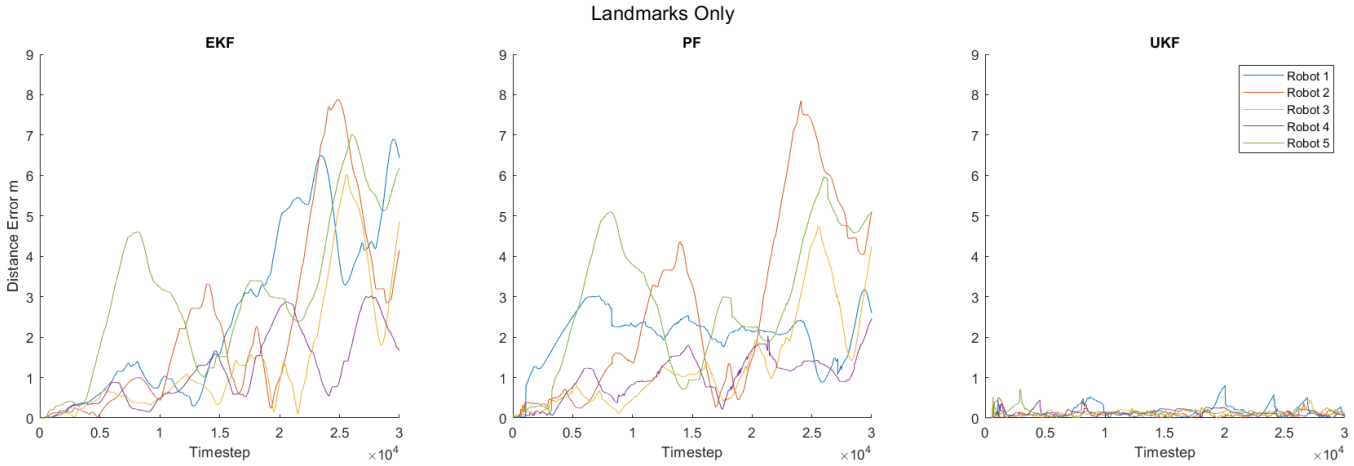

Fig. 1: Distance error for EKF, PF, and UKF using the landmark only approach.

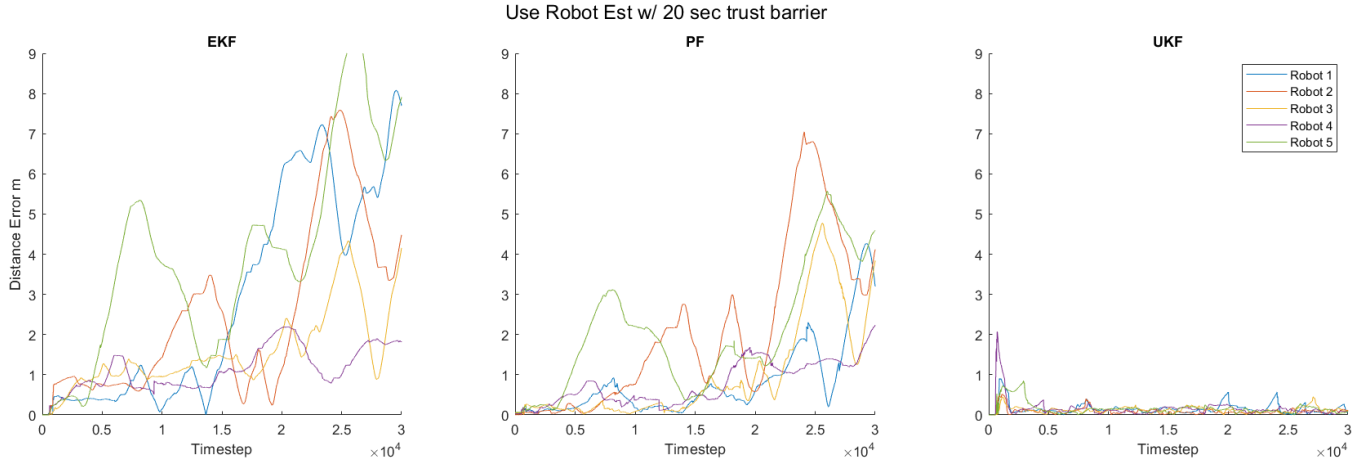

Fig. 2: Distance error for EKF, PF, and UKF using the trust factor for using other robots' estimated pose in measurement updates.

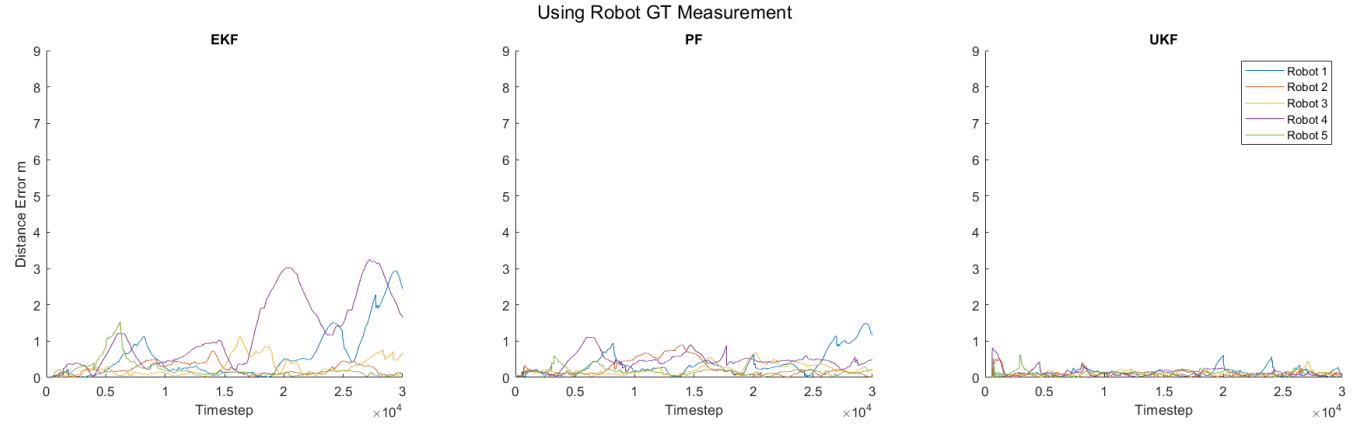

Fig. 3: Distance error for EKF, PF, and UKF using the ground truth pose of other robots in measurement updates.

C. Trust Factor

The results for the trust factor approach are presented in figures 2 and table II.

D. Robot Ground Truth

The results for the ground truth approach are presented in figures 3 and table III.

TABLE I: Distance RMSE and Standard Deviation for Landmarks only approach

	R1	R2	R3	R4	R5
EKF RMSE	3.31	3.19	2.11	1.47	3.56
EKF Var.	2.10	2.21	1.55	0.88	1.84
PF RMSE	2.14	3.53	1.76	1.14	3.29
PF Var.	0.63	2.27	1.17	0.53	1.65
UKF RMSE	0.21	0.13	0.13	0.15	0.16
UKF Var.	0.14	0.08	0.06	0.07	0.08

TABLE II: Distance RMSE and Standard Deviation for Using Trust Factor

	R1	R2	R3	R4	R5
EKF RMSE	3.90	3.25	1.83	1.25	4.67
EKF Var.	2.62	2.10	0.96	0.54	2.52
PF RMSE	1.25	2.86	1.61	0.94	2.57
PF Var.	0.95	1.99	1.26	0.53	1.51
UKF RMSE	0.20	0.12	0.14	0.27	0.22
UKF Var.	0.14	0.08	0.07	0.21	0.15

TABLE III: Distance RMSE and Standard Deviation for ground truth approach

	R1	R2	R3	R4	R5
EKF RMSE	0.92	0.27	0.38	1.56	0.34
EKF Var.	0.70	0.15	0.25	0.97	0.26
PF RMSE	0.49	0.34	0.26	0.50	0.19
PF Var.	0.33	0.23	0.14	0.21	0.09
UKF RMSE	0.16	0.12	0.12	0.17	0.14
UKF Var.	0.10	0.08	0.06	0.11	0.07

V. Discussion, Conclusion and Future Work

In all filters, we clearly observed a marked improvement in the quality of pose estimation through the implementation of the trust factor. The number of measurement updates decreased, but the accuracy of the filter approaches the ground truth error more than the landmark only mode. Looking at the result figures, we see that the distance error often increases for the latter half of the experiment. This is because we don't receive many camera observations during this time, so we do not really get a chance to correct our poses.

The UKF has a simple structure. We used a simple but reliable motion model when applying UKF. The distance error stays at a low level under various conditions. The correction of the landmark measurement and trust factors improve the performance of UKF indeed.

More work could be put in to tuning the trust factor to be dependent on more factors than just time since last measurement update. The current implementation fails to account for a stalled robot. For instance, if a robot is commanded to wait at a certain location for a period of time, it could be considered no longer trustworthy, even though it's accuracy hasn't diminished in any way. To improve the trust measurement, we can implement a distance based metric, or some fusion of the time and distance factors.

A gradient trust factor can be implemented as well. We chose to not pursue this approach due to the complications of implementing partial updates across multiple filters, but if we had focused on just one filter and derived how to implement lower weighted trust based updates. For example, with a PF we can simply perform a weighted odds calculation across landmarks when there is a trust disparity between them. However, with the two Kalman variants, this becomes much more tricky.

An additional future step is to begin removing landmarks in order to force the robots to rely on relative measurements for their updates. If we explore this route more thoroughly, we can see the trust factor take more of an impact as the landmarks will be more sparse. However, this will require more in depth tuning of the noise parameters for the filters to ensure that everything will propagate properly without losing localization between measurement updates.

The algorithms we implemented can also be applied to different datasets in order to validate their functionality in different circumstances. For example, MRCLAM9 has several walls set up around the arena, so sightlines are limited and observations tend to be more scarce. By analyzing the performance of our algorithms under different conditions, we can see where they are weak and improve them in those areas.

Lastly, additional work can be done exploring the propagation of trust through robots. In the current implementation with the binary trust metric, once a robot receives a valid relative measurement it is fully trustworthy again, but with the gradient implementation a partial trust restoration could be implemented in order to specifically tune robot trust propagation in context of their situation.

References

[1] S.I. Roumeliotis and G.A. Bekey, 2002, Distributed Multirobot Localization, IEEE Transaction On Robotics And Automation Vol 18, No.5, October 2002

[2] Thrun, S. and Burgard, W. and Fox, D., "Probabilistic Robotics".Intelligent Robotics and Autonomous Agents series, MIT Press, 2005.

[3] Leung K Y K, Halpern Y, Barfoot T D, and Liu H H T. "The UTIAS Multi-Robot Cooperative Localization and Mapping Dataset". International Journal of Robotics Research, 30(8):969–974, July 2011.

[4] Project Source: https://github.com/graygr/rob-530-project

[5] Project Presentation: https://youtu.be/r0BZHNcIqdw