A Novel Way to Implement Self-localization in a Multi-robot Experimental Platform

Sheng Zhao and Manish Kumar, Member, ASME

Abstract—Self-localization is an important part of multi-robot experiments. The traditional way to obtain the global pose (position and orientation) of a robot is to attach a unique pattern to the robot. A centralized sensor, such as a camera, is typically used to identify the pattern and estimate its pose directly. However, the pattern-based localization scheme lends itself a poor scalability property because, in presence of a large number of robots, the recognition of all patterns may result into a computational bottleneck. Moreover, the implementation of this technique in a decentralized framework is questionable. Additionally, the practicality of designing a large number of unique patterns and attaching them on often miniature robots present some challenges. To overcome the use of a pattern, a novel method has been proposed in this paper for the robots to obtain their global poses without using any pattern. In this method, we run Extended Kalman Filter (EKF) on each robot to fuse the global position data with the control input data to estimate the orientation. In a multi-robot setting, this becomes challenging because the positional data is untagged and robots do not have a priori knowledge about which data pertains to their own position. To overcome this issue, we propose a method in which each robot can identify its own track from the other’s tracks by comparing the average errors over time between prediction and measurement in the EKFs that are run on each candidate trajectory. Therefore, instead of trying to identify robots in a centralized manner for localization, we distribute the task to each robot and let them estimate their pose on their own. Extensive simulations and experiments have been conducted and the results are provided to show the effectiveness of this method.

I. INTRODUCTION

MULTI-ROBOT research has gained a broad attention over the recent years [1, 2]. Comparing to the single robot system, multiple robots working in a team can be more robust to individual failures and more efficient in time to accomplish a job. The multi-robot system has wide potential applications including large area surveillance, search and rescue operation, perimeter protection, and intruder detection [3].

A lot of theoretical advancement has been made in multi-robot cooperative control area, and a preferred way to verify the theories has been the use of simulation. However, experimental validation of theories is highly desirable because the real world environment presents several challenges including sensor limitations, noise in sensory information, communication error, and time delays which are difficult to simulate in a computer. There have been some efforts in developing multi-robot experimental testbed. For example, the University of Pennsylvania has built their own miniature robots called Scarab [4] which are equipped with onboard cameras and laser sensors. Three IR emitters on the top of the robot emit lights at certain frequencies. Overhead cameras can capture these patterns to identify each robot and estimate the pose directly. The other examples of multi-robot experimental testbeds include the one developed at MIT. They also have designed miniature robots called SwarmBot [5] which are equipped with specially designed IR sensor rings that can measure the relative positions of the neighboring robots. A single IR emitter on the top of each robot is used to identify the robots and provide their ground truth global position. The heading of robots in the global coordinate system is estimated by combining local estimates of bearing to neighbor and the global estimates of neighbor’s positions [5]. It can be noted that both of the above testbeds require unique patterns for identification of the robots. Other than IR emitters, visual patterns such as circular codes in SwisTrack [6] or ARTag in Augmented Reality system [7] have been widely used to identify robots and estimate the position and the heading. Actually, due to the fact that the pattern recognition algorithms are already mature in literature, using patterns has become a popular method to identify robot and provide the global pose in an indoor experiment setup.

The benefit of using a pattern is that it provides a unique ID and can be used to estimate the heading of a robot quite easily. However, using patterns has its own drawbacks. Firstly, the pattern based localization system is poorly scalable to the number of robots. When the number of robots is large (hundreds or thousands), creating a unique pattern for each of them is very difficult from a practical point of view. Furthermore, compared with the simple blob extraction algorithm for image processing, the identification of patterns usually require heavy computational load and the time taken to recognize all patterns will increase dramatically with the increase in the number of robots. Secondly, the pattern based localization system won’t work in the situations where it is infeasible to attach any pattern on the robot. Usually, either the IR emitters or other visual patterns take extra space to install. Some robots like miniature UAV don’t have the room to install patterns like that. Furthermore, pattern based localization schemes usually require a central sensor, and is difficult to implement in a decentralized fashion. Attaching a pattern to a robot amounts to its identification, a concept which against the true spirits of many multi-robot control and swarming algorithms.

Manuscript received September 22, 2009.
S. Zhao is with the University of Cincinnati, Cincinnati, OH 45221 USA (513-763-9924) zhaose@mail.uc.edu
M. Kumar is with University of Cincinnati, Cincinnati, OH 45221 USA kumarmu@ucmail.uc.edu
This paper investigates if we can carry out the self-localization without using a pattern. We propose a method which distributes the self-localization task to each robot and let them estimate their position and heading on their own. This provides a decentralized and scalable way to solve the self-localization problem in a multi-robot scenario.

In this paper, we utilize an overhead camera to provide positional information of all the robots without tagging them. There is no orientation information provided by the camera. This information is fed to each robot which uses our proposed algorithm to carry out data association and tracking. The challenges in accomplishing the self-localization in multi-robot setting arise from two aspects. First is the estimation of robot’s heading. Since we do not use any pattern, the camera does not provide heading information. Then the problem boils down to the estimation of heading based upon only positional observation. The second challenge is the data association. Without a unique ID to the data, it is difficult to associate data to the robot, i.e., to determine which position data from a set of data points belongs to a particular robot.

The following sections of the paper will introduce the proposed method. First, the next section formulates the EKF used to estimate the heading and track the robot in a single robot scenario. Next, the third section introduces the data association and self-localization technique in a multi-robot scenario. Finally, we demonstrate the effectiveness of our algorithm via experimental data which is followed by the conclusion and a discussion of the future work.

II. Tracking in a Single Robot Scenario

In this section, we formulate an Extended Kalman Filter (EKF) to track a robot’s position and heading. EKF is broadly used for nonlinear system in estimation and tracking problems [8]. In our problem, we define the state of the robot to include its two-dimensional position \( x \) and \( y \), and heading \( \theta \). Therefore the state \( X \) at each sampling time step \( k \) is defined as:

\[
X_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}
\]

Now we can write down the discrete system equations:

Dynamic equation:

\[
X_k = f(X_{k-1}, U_k, W) = FX_{k-1} + B_{k-1}U_k + W
\]

Observation equation:

\[
Z_k = HX_k + V
\]

Where

\[
F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
B_{k-1} = \begin{bmatrix} T_s \times \cos(\theta_{k-1}) \\ T_s \times \sin(\theta_{k-1}) \\ 0 \end{bmatrix}
\]

\[
H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\]

\[
U_k = \begin{bmatrix} v_k \\ \omega_k \end{bmatrix}
\]

The \( Z_k \in \mathbb{R}^2 \) is the observation of the robot’s position, i.e., \( x_k \) and \( y_k \). \( U_k \in \mathbb{R}^2 \) is the motion command or control input of the robot that is composed of \( v_k \), the translation velocity, and \( \omega_k \), the turning rate. \( T_s \) is the sampling time. \( W \in \mathbb{R}^3 \) and \( V \in \mathbb{R}^2 \) are the system noise and observation noise respectively. We assume they obey the Gaussian distribution. The covariance matrices of the system and observation noise are represented by as \( Q \in \mathbb{R}^{3 \times 3} \) and \( R \in \mathbb{R}^{2 \times 2} \) respectively. As we can see, due to the nonlinear property of matrix \( B_{k-1} \), the system is nonlinear. Hence an EKF is formulated to track the state of the system. The update equations for the EKF are listed below:

The prediction equations are:

\[
\dot{X}_{k|k-1} = FX_{k-1|k-1} + B_{k-1}U_k \\
\dot{P}_{k|k-1} = A_kP_{k-1|k-1}A_k^T + Q
\]

The measurement update equations are:

\[
K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \\
\dot{X}_{k|k} = \dot{X}_{k|k-1} + K_k(Z_k - HX_{k|k-1}) \\
\dot{P}_{k|k} = (I - K_kH)P_{k|k-1}
\]

where \( Z_k - H\dot{X}_{k|k-1} \) is called measurement residual, and \( A_k \) is the Jacobian matrix of partial derivatives of \( f \) with respect to \( X \) [9]:

\[
A_{[i,j]} = \frac{\partial f_{[i]}(X_{k-1}, U_k, 0)}{\partial X_{[j]}}
\]

Applying this to (2), we get the \( A_k \) as:

\[
A_k = \begin{bmatrix} 1 & 0 & -v_k \times T_s \times \sin(\theta_{k-1}) \\ 0 & 1 & v_k \times T_s \times \cos(\theta_{k-1}) \\ 0 & 0 & 1 \end{bmatrix}
\]

By using this EKF we can estimate the heading \( \theta_k \) and position of the robot at the same time.

III. Tracking and Self-Localization in a Multi-Robot Scenario

Tracking and self-localization in a multi-robot scenario present new challenges. When there are multiple robots in the platform, each robot will receive all the other robots’ positions at the same time. In order to use the EKF developed in the last section to track a robot’s position and heading, the robot needs to associate to its own observation data from
the set of data received at each time step. This would be easy if the state of the system is fully observable. In other words, if we can observe the heading $\theta$, we can associate the data by simply choosing the measurement with the smallest measurement residual at each time step if we assume the command $U_k$ is different for each robot. However, our system is partially observable (see (4)). Since we don’t have the precise knowledge of the current heading which plays a critical role in prediction and data association, we cannot perform the association task on a stepwise basis. Fig. 1 shows that the incorrect estimation of the initial heading of a robot can possibly lead to incorrect identification of its own track. Hence, the data association should be carried out in a different way in our case.

Data association is a typical problem in multi-target tracking area which involves finding the correspondence between the data received at current time step with the previous estimates. In this paper we assume that, for every robot’s track, there is only one potential data among the received data at each time step that can be associated. So the data association problem in this paper can be divided into two sub-problems:

- One is how to generate all the potential tracks from the data.
- Then the next one is how to identify a robot’s own track among these candidate tracks.

To solve the first sub-problem, we use the Nearest-Neighbor algorithm (NN) [10] to help in the association process. This algorithm uses Euclidean distance as the metric of correlation between two data points $x_1$ and $x_2$:

$$\text{Correlation}(x_1, x_2) = ||x_1 - x_2||$$  \hspace{1cm} (16)

Firstly, we initiate a potential track on each measurement data obtained at the first time step. Then at each time step a new measurement data arrive, we apply NN algorithm on the current measurement $Z_k$ and the previous measurement $Z_{k-1}$ on each potential track then associate the current measurement data to the track with the smallest $\text{Correlation}(Z_k, Z_{k-1})$. In this way, the measurement can be associated at the beginning to generate all potential tracks. However, till now the tracks are still not associated to robots.

To overcome the track identification problem, we need some unique information for each robot and the only unique information each robot has is the motion command or control input $U_k$. Relying on this unique information, we can successfully distinguish a robot’s track from other tracks by running one EKF on each candidate track. In the proposed technique, each robot maintains one EKF for each track and applies its own control input data on each EKF. Then, the predicted position is compared with the current measurement associated to this track. If the track matches the motion command of the robot, the average measurement residual over time should be the smallest one among all other EKFs. This happens because of the fact that if the data association is carried out properly, the estimation of heading will converge and eventually the average measurement residual will also converge to the smallest value. Hence, we define:

$$e^l_k = ||Z^l_k - H \hat{X}^l_{k|k-1}||$$  \hspace{1cm} (17)

Where $e^l_k$ is the prediction and measurement error (or norm of measurement residual) at time step $k$. $Z^l_k$ is the associated measurement and $H \hat{X}^l_{k|k-1}$ is the predicted position. The superscript $l$ indicates different candidate tracks. The average prediction and measurement error on candidate track $l$ over time is defined as:

$$E^l = \frac{1}{T} \sum_{k=1}^{T} e^l_k$$  \hspace{1cm} (18)

$E^l$ is in fact the metric reflecting the error in the estimation of heading and so chosen to decide the correct track of the robot. After $T$ steps, we choose the track, among all potential tracks, with the smallest $E^l$ as the correct track of the robot. The choice of $T$ is based on trial and error.

To summarize, the tracking and self-localization in multiple robot scenario includes two phases. In the first phase, we use NN algorithm to generate all possible tracks and run one EKF on each track with the commanded input data. After $T$ steps, we compare the average error $E^l$ on each track and choose the track that has the minimum $E^l$ as the correct track of the robot. The second phase starts after the successful initiation of track. In this phase, we apply the NN algorithm in a stepwise fashion to associate the new measurements to this initialized EKF at each time step. Since the correct track has been initiated and the estimation of heading has converged, the prediction is much accurate now. Hence the NN algorithm, in second phase, is used to associate data based on $\text{Correlation}(Z^l_k, H \hat{X}^l_{k|k-1})$. The illustration of the whole algorithm is shown in Fig. 2.

IV. Experimental Setup

In this paper, we use Khepera III robots for the experimental validation. Overhead camera (Cognex Insight 5600) is used to capture images and transfer them through Ethernet to the desktop for further processing. On the desktop, we
Fig. 2: The scheme of the self-localization algorithm in one robot.

use SwisTrack open source image processing software to extract the positions of robots. This can be done using simple blob extraction algorithm. In order to obtain the ground truth of the global heading (for comparison purposes) for validating our method, we attach the circular patterns [6] on the top of robot. The processing speed on the desktop is 40ms per frame. After the SwisTrack obtains all robots' positions and headings, it broadcasts position data to each robot via wireless network. Each robot logs the measurement it receives and the local odometry data (representing the commanded control input) to the desktop at each time step. Then we apply EKF on these data offline using Matlab. The sampling time $T_s$ is approximately 0.5s. In the following experiments, the control program running in each robot is random motion program with obstacle avoidance. The maximum translational velocity is 0.04m/s and the maximum turning rate is 0.3rad/s. Player/Stage [11], an open source software, is used as the control and communication the robot. The schematic of the whole overall system control and communication scheme is shown in Fig. 3.

A. Experiment 1 (Single robot)

In the first experiment, only one robot is used to test the performance of the proposed EKF's tracking method. The trajectory of robot is shown in Fig. 4. As we can see from Fig. 5, the estimation error of the heading converges and remains mostly between ±5 degrees. Also, the estimation converges to the correct heading in 10 time steps. We can also notice that there are few steps have larger error that represents outliers in data because of errors in image processing.

B. Experiment 2 (Multiple robots)

The second experiment is conducted using four robots. In this experiment, at each time step, each robot receives all the robots’ positions and headings, it broadcasts position data to each robot via wireless network. Each robot logs the measurement it receives and the local odometry data (representing the commanded control input) to the desktop at each time step. Then we apply EKF on these data offline using Matlab. The sampling time $T_s$ is approximately 0.5s. In the following experiments, the control program running in each robot is random motion program with obstacle avoidance. The maximum translational velocity is 0.04m/s and the maximum turning rate is 0.3rad/s. Player/Stage [11], an open source software, is used as the control and communication the robot. The schematic of the whole overall system control and communication scheme is shown in Fig. 3.

A. Experiment 1 (Single robot)

In the first experiment, only one robot is used to test the performance of the proposed EKF’s tracking method. The trajectory of robot is shown in Fig. 4. As we can see from Fig. 5, the estimation error of the heading converges and remains mostly between ±5 degrees. Also, the estimation converges to the correct heading in 10 time steps. We can also notice that there are few steps have larger error that represents outliers in data because of errors in image processing.

B. Experiment 2 (Multiple robots)

The second experiment is conducted using four robots. In this experiment, at each time step, each robot receives all the robots’ positions. The problem is to determine which among the complete data is a robot’s own position, and to estimate the track that includes its heading. During the first phase of our algorithm, each robot needs to maintain four EKFs (one for each trajectory candidate). The trajectories of four robots are shown in Fig. 6.

If we plot the average error given by (18) for each EKF track in each robot (Fig. 8), we can see that as time goes by, one of the tracks’ average errors converges to the smallest value as compared with other tracks. Therefore the robot can obtain the correct track among several tracks by comparing the average errors for each EKF track. In this experiment, after 50 steps, all robots finish the first phase of the algorithm. At the end of this phase, the robot will terminate all other EKF tracks and keep the correct one. In the second phase, each robot uses the nearest neighbor algorithm to associate data to this EKF track.

An important aspect for the successful phase 1 is that the...
command control input for each robot should be different to some extent from the others. For example, if multiple robots run in a straight line or a circle or exhibit some similar motions, the first phase can possibly fail or take a longer time to finish. Experiment 3 is conducted to illustrate this aspect.

C. Experiment 3 (Worst case)

The robots’ trajectories are shown in Fig. 7. As we can see, robot 3 and 1 are moving in a similar trajectory (start moving in a straight line and then turn left). Hence, it can be seen from Fig. 9c that it takes approximately 150 time steps for robot 3’s trajectory error to be separable from others. So is the case with robot 1 (Fig. 9a). However, for robots 2 and 4, which follow very distinct trajectories, the trajectory errors converge quickly (Fig. 9b and d). This is intuitive because the control inputs, which play a crucial role in trajectory identification, are similar for robots 1 and 3.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel method to localize robots in a multi-robot control experiment. The proposed method is based on Extended Kalman Filter (EKF) in which each robot estimates its global position and heading by fusing global position measurements and control inputs. It was shown that the heading of the robot can be estimated pretty accurately without even its direct measurement. The error of the estimated heading is mostly within ±5 degrees of error. In the multi-robot scenario, each robot associates data at each time step to create all possible tracks based upon all measurements and then identify its own track among these tracks by comparing the average error between the predicted position and the observation for each EKF track. This method helps initiate the track, and once initiated, NN algorithm is used to associate the further data to maintain the track. The major advantage of the proposed technique is that it can be applied in a decentralized manner and is scalable. It was seen that the ability of the proposed method to initiate the correct track depends upon distinct control inputs of the robots. In other words, the motion of the robots should be different enough for them to identify their tracks in a short time. Hence, a specially designed random motion at the beginning can be used to aid in the track initiation process. Future work includes applying the proposed method in a decentralized experimental framework in which the robots use the onboard sensors to carry out the self-localization and tracking.

REFERENCES

Fig. 8: The average prediction-measurement errors $E_l$ for experiment 2

Fig. 9: The average prediction-measurement errors $E_l$ for experiment 3