Localization and Rover Home Approach

Mason U'Ren

***Abstract*— Something abstract.**

I. Introduction

Compexity of multi-agent systems can be broken down into two pieces: 1) clear and current inter-agent communication, and 2) percision agent localization about a global point of reference. To successfully accomplish a group task, each agent needs to have relavant, accurate infromation about it's neighbors and be able to correctly determine its own location. Characterized by low-cost, high-quantity demands, the challenge surrounding swarm robotics has thus been how to maintain high performance with economical sensors that are prone to error. These observations prompted us to develop, then evaluate alorgithms that seek to eliminate noise within data feedback. To this end, we propose a unique approach for dynamically calculating agent location within a multi-agent network.

Concise communication across a field of agents, boils down to the structure of the proposed model and the alrgorithm that decides when and what to communicate.

The first is discussed by Ferber et al. in [1], where the authors seak to create a interlinked rover communication network based around a two tiered obstraction model. They propose a concrete level, composed of the physical organization of the rovers core concepts (agents, groups, and role), and a second tier or abstract/methodological level, which defines all possible roles, valid interactions, and structure of groups and organizations. At the concrete level, agents are a member of a group, group contains a role, and role is handled by agents. While on the abstract level, group is contained by an organization which is instantiated from an organization structure, organization structure holds a group structure which can then instantiate to a group or define an interaction, interaction is defined between roles, and an agent is instantiated from an agent class. Thus by having a two tiered model built on simplicity and abstraction, we can confidently create any style of organization based around agents, groups, and roles.

The later, examines the threshold that decides when it is most appropriate to $send$ or $not - send$ information. The authors [2] establish an inter-rover communication thresh- old, which decides when to communicate information, when constrained by partial rover maps of their overall world. Communication actions are simply $communicate$ or $dont\ communicate$, where if one agent chooses to communicate, all agents broadcast their local state to eachother at some cost. This synchronizes the world view generated by the rovers, providing complete information about the current world state. They develop a communication protocol, $Model\ Lookahead$, that only requires the rovers to hold the latest message of their projected world, instead of saving the entire message history, which ultimately saves times computationaly. Introducing and controlling messages in this way proved favorable as overcommunication of redundant information was decreased. In two different experimental setups, this approach produced smooth and monotonous degradation of values as communication costs increased.

Beyond comminication, navigating within any environment requires the estimation of an agents current position in reference to a global anchor. Analysed by Aragues et al. [3], the writers address the problem of estimating rovers poses who do not share any common reference frames, when working with multi-agent systems. Their goal, in the presence of added noise, is to compute each agents pose, either relative to a specified anchor node or exclusively using neighbor- to-neighbor or local interations. The setup is to arrange independent rover interaction (poses) into a directed graph and have an anchor node placed at pose (0,0,0). Localization then is achieved in three phases: 1) an estimate of all the node orientations $\hat{\theta} \in \mathbb{R}^{n-1}$ in realation to the anchor, 2) then tangential position measurements of the nodes are expressed in terms of previuos iterations, 3) finally estimate pose positions, **p**, and orientations, **x**, are calculated using $\mathbf{p}^* = ((\mathbf{x}^*)^T, (\theta)^T)^T$. Results proved favorable as both implementations converged to values within one standard deviation of the actual poses. An interesting aside found that the distributive approach converged under a wider range of classes of communications graphs.

In summary, the main contribution to this paper is to provide algorithms that correct the inherint drift within GPS units and establish a stable communication network between multiple agents. To achieve this, localization algorithms must collectively create and agree on a global anchor node for the agents to reference, while communicating only pertinant environmental information to the group members, reducing redundant messages.

The remainder of this paper is organized as follows. Section II provides a mathematical description of the sensor models and the localization problem, which is followed in Section III by an description of the methods used. Next analysis and experimental results are detailed in Section IV, followed in Section V by the final conclusions and directions for future work.

II. Problem Statement

The advised communication and localization algorithms are provoked by the desire to use swarms of small inexpensive rover agents to explore unknown environments and successfully complete tasks as group. Each rover is assumed to be a small, autonomous, skid-steer, robotic vehicle driven by four independently powered wheels. Each rover is equipped with three sonar sensors, a webcam, an inertial measurement unit (IMU), wheel odometry sensors, and a global positioning system (GPS) receiver. Rotating the tires at different speeds creates skid-steer which gives the rover navigational control when maneuvering within a 2-dimensional horizontal plane. Hence, if the speed of the right tires are greater than the left, the rover will steer left and vice-versa. Although the following explanation details the use of a skid-steer robot, the proposed algorithms can be easily applied to general wheeled mobile agent with noisy positioning sensors.

The state of a skid-steer vehicle x can be represented by the triplet $(x_d, y_d, \theta) \in SE(2)$, where $(x_d, y_d) \in \mathbb{R}^2$ describe the position of the vehicle and $\theta \in \mathbb{S}^1$ represents the orientation. Vehicle control input $u \in \mathbb{R}^2$ is modeled $(v, w) \in \mathbb{R}^2$, where v is the forward velocity, and w is the rate of change in vehicle orientation. Motion evolves according the following kinematic equation:

$$\dot{x} = \begin{bmatrix} \dot{x_d} \\ \dot{y_d} \\ \dot{\theta} \end{bmatrix} = f(x, u) = \begin{bmatrix} v\ cos(\theta) \\ v\ sin(\theta) \\ w \end{bmatrix} \quad (1)$$

When traversing to specific location we rely on a low-cost GPS unit to provide feedback for our algorithm. The sensor generates a perceived position within a, unit specific, error margin $\mathcal{E}_r$, by measuring the difference in time stamped satellite location messages and the time the message was received. The receiver uses the messages it receives to determine the transit time of each message and computes the distance to each satellite using the velocity of light [4]. Due to potential positioning error, the calculation distance should be done with at least four satellites [4]. We can model this with:

$$X_A = (X_s, \delta t, \mathcal{D}_r) \quad (2)$$

where X_A is the calculated perceived pose, X_s is the position of the satellite at the time of transimition, δt is the differnce in transmittion and reception times, and $\mathcal{D}_r$ is radial drift.

III. Experiment

NOTE: algorithms have been omitted, but will be in conference copy.

To test the validity of our proposed algorithms we needed a way to mimic the error inherit within cost-effective GPS units. Using python, we modeled the drift form our actual position to our perceived location X_{GPS}, seen below:

$$X_{GPS} = \begin{cases} (x_c, R_{max}, d_{max}) & \text{On first iteration} \\ (x_p, R_{max}, d_{max}) & \text{otherwise} \end{cases} \quad (3)$$

where x_c is the origin pose, x_p is the previously generated error pose, R_{max} creates the min/max bounds that dictate how far the rover can drift, and d_{max} is the range the rover is allowed to drift in any direction at each iteration.

Using eq. 3, we can create relative agent position for both the static and dynamic cases. When dealing with a stationary agents, we simply set x_p to x_c and then proceed to the next iteration, but if the rover is moving we cannot do a straight substitution. Instead we need to factor in the goal position of the rover x_g in combination with the timestep between sensor readings t to calculate a change in distance $\triangle x$, thus $\triangle x = x_g \cdot t$. We then add the difference to x_c and then substitute the final value into x_p, as seen below:

$$x_p = \begin{cases} x_c & \text{static} \\ x_c + \triangle x & \text{dynamic} \end{cases} \quad (4)$$

Key to our algorithms is the ability to correct the induced error, adjusting the percieved pose by an offset. Assuming that w is characterized by:

$$w = \begin{cases} 1 & \text{if } d_a < 0.1 \\ e^{-d_a} & \text{otherwise} \end{cases}$$

where d_a is the agent's distance to the anchor node, u is model with:

$$\begin{cases} 1 & \text{if } d_g < 0.05 \\ \frac{1}{(1+d_g)} & \text{otherwise} \end{cases}$$

such that d_g is the agent's distance to the goal pose, and v follows:

$$\begin{cases} 0 & \text{if } \frac{\delta x}{\delta t} < 0.01 \\ \frac{\delta x}{\delta t} \cdot 0.1 & \text{otherwise} \end{cases}$$

then we can say,

$$x_c = w \cdot (x_c - x_p) + u \cdot (x_g - x_c) + v \cdot cos/sin(\theta) \quad (5)$$

such that x_c, x_p, x_g are current, previous, and goal positions respectively, and cos/sin is used when considering x or y respectively. This allows us to account for both the static and dynamic cases of agent localization where, $w \cdot (x_c - x_p)$ corrects GPS drift, $u \cdot (x_g - x_c)$ corrects odometry error, and $v \cdot cos/sin(\theta)$ looks to see if the rover is moving and in which direction.

IV. Results

A frequent issue with localization when using cost effecitve GPS sensors is a radial drift component that causes inaccurate location readings. Managing this issue boiled down to separating the problem into two cases: 1) anchor initialization, then 2) dynamic localization.

Anchor initialization uses trianglulation and vigorous averaging to mark a global reference point. The goal is to, through a designated number of iterations, I_{max}, create a anchor node that exists at an agreed upon new origin of each agents virtual map. Initial experiments placed rovers at locations (0,1), (1,1), (1, 0), (-1,0), (-1,-1), and (1,-1) and then based on simulated GPS drift of each rover

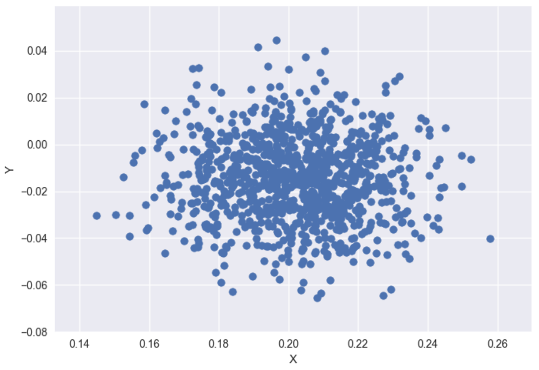

Fig. 1: *Possible anchor node positions, between rovers (1000 iterations).*

Classification Report		
	X	Y
Count	1000	1000
Mean	0.201831	-0.014522
Std	0.017800	0.017833
Min	0.144812	-0.065531
25%	0.190542	-0.026514
50%	0.202951	-0.014718
75%	0.213567	-0.002392
Max	0.257954	0.044440

TABLE I: *Classification report of the our Anchor Initialization algorithm over 1000 iterations.*

approximated the central anchor node. Figure 1 displays promising results of this experiment over 1000 iterations.

If each point in the figure is an agreed upon global reference point, then the graph suggests that our initialization algorithm is percise, but not entirely accurate. This is verified by Table I, where we can see that the mean of the distribution lies at roughly at (0.2, -0.01), which is approximately 20cm off of our goal reference point. This is close to our goal origin of (0,0), but while manuevering unique environments, confidence in current agent position is imperitive to accomplishing group tasks, thus this offset may present issues with complex algorithms. On the contrary, the low standard deviation of (0.0178, 0.01783), for x and y respectively, suggests that colaboratively each agent is finding the same common global point of reference between them, with minimal and excusivable error.

Although, when working with non-static systems, small errors begin to expontentially grow and destroy the integrity of multi-agent control algorithms. Using eq. 5 we can correct these errors by accounting for drift, wheel skid, and current path. Figure 2 shows the localization of a single rover. We can see the ideal path in black, originating at the origin (0,0) and traversing to point (1,3). The green dots are the simulated GPS error generated by eq. 3 as they attempt to follow the goal path, while the blue dots are the corrected path created by eq. 5.

Initial results look promising, since no trianglulation is needed between rovers to improve relative postion accuracy.

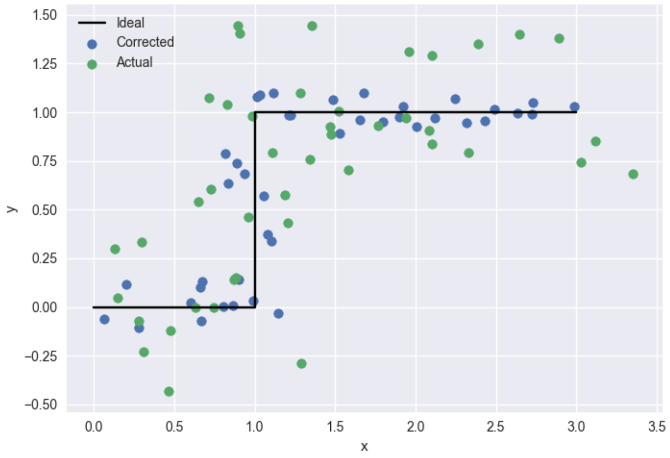

Fig. 2: *Dynamics localization as rover travels from (0,0) to (3,1).*

This is bolstered by a almost zero value of mean square error, seen below, which corresponds to the expected value of the squared error loss, that is the difference between the estimator and what is estimated.

- Mean Square Error = 0.014038
- R^2 Regression Score = 0.925466

In combination with the R^2 regression score of roughly 1, we can say that this dynamic localization equation appears to be working effectively and efficiently.

V. Conclusion

NOTE: this is not my conference paper conclusion. This is an area where I'm updating you on the progress and concerns of the project.

Initial tests of our localization equations and algorithms proved promising. Expected values within either the static or dynamic case showed the capability to mark a global reference point and correct local positions while traversing an environment, although there is cause for concern for each case.

Our Anchor Initialization algorithm can mark a global reference point between all the rovers, but this point does not always align with the true origin, which can cause issues for more complex algorithms that monitor thier distance from the origin. Also without taking directed GPS data from the rovers, there is the possiblity that our GPS genator model is not a true fit and we are building equations that correct the wrong aspects of the generated error. Thus the equations used would not work effectively in the real world. Using and recording true GPS error from the rovers would help fine tune our model, but reusing the same data set to prove an algorithms correctness is prone to overfitting.

Regarding dynamic initialization, there is cause for concern as well for overfitting. The fact that the rover did not need to be trianglulated after the anchor node was created is suspect. Eq. 5 may be taking too many liberties with information that may be unkown in the real world.

In later generations of this implementation we plan to actively triangulate the agents in hopes to confirm that our algorithms are valid. Also the $driveway$ component of this implementation, although already written out, greatly depends on the accuracy of localization. Since localization is still being fixed and validated, the $driveway$ portion was left out, but will hopefully be in later papers.

References

[1] J. Ferber and O. Gutknecht, "A meta-model for the analysis and design of organizations in multi-agent systems," in *Multi Agent Systems, 1998. Proceedings. International Conference on*. IEEE, 1998, pp. 128–135.

[2] R. Becker, A. Carlin, V. Lesser, and S. Zilberstein, "Analyzing myopic approaches for multi-agent communication," *Computational Intelligence*, vol. 25, no. 1, pp. 31–50, 2009.

[3] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues, "Multi-agent localization from noisy relative pose measurements," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 364–369.

[4] N. Rahemi, M. Mosavi, A. Abedi, and S. Mirzakuchaki, "Accurate solution of navigation equations in gps receivers for very high velocities using pseudorange measurements," *Advances in aerospace engineering*, vol. 2014, 2014.