Multi-agent Collaborative Localization in ROS

Jiawei Mo

December 13, 2015

1 Problem Description

Localization is the problem where robot determine its relative position in a given map. While because of sensor uncertainty, robot pose cannot be sensed directly. Multiple measurements over time are integrated to generate better position estimation. However, the integrated result still have more or less uncertainty.

The uncertainty is exponentially large when the robot is kidnapped and dropped to a new position, especially when there are several similar places in the map. For example, in Figure 1, robot is kidnapped from green point to blue point. To robot's belief, all square rooms can be the room it is kidnapped to. To figure out the new position by itself, it has to leave the room and explore the map.

Most existing localization algorithms focus on single robot localization, such as Extended Kalman Filter(EKF) Localization and Adaptive Monte Carlo Localization(AMCL). They have relatively high uncertainty about robot's pose. What's worse, EKF Localization fails in kidnapped problem. AMCL Localization could recover from being kidnapped eventually but takes much time to gain confidence.

This paper introduces a new method called Multi-agent Collaborative Localization(MACL) to optimize robot localization accuracy. The main idea is sharing information between robots. Specifically, when robot_0 detects robot_1, robot_0 estimates robot_1's position and send its belief to robot_1. Robot_1 receives the message from robot_0 and combines it with its own belief. By fusing two position estimates, robot_1 gets better estimate of its position.

MACL is powerful in kidnapped problem. After being kidnapped, robot

could relocate itself immediately using the information shared by robots who have already be there, rather than learning by exploring the new place by itself. Intuitively, in real world, when a person is put into a new building, she has to look around, or even go out of the building, to check which building she is in. However, if someone in this building tells she the answer, she would be able to localize herself immediately with high certainty.

2 Related work

An example of multi-robot localization system was discussed in [6]. The authors introduced an mechanism called collective localization. A single estimate of landmark by one robot has relatively high uncertainty. Their solution is collecting estimates from all robots and using Kalman filter to calculate an accurate estimate. Accurate estimate of landmarks will in turn reduce the uncertainty of each robot. The centralized Kalman filer was decomposed into smaller communicating filter hosted by each single robot. Information exchange happens only when two robots detect each other.
Another example is [2]. The key idea is integrating measurements from multiple robots. Then each robot localizes itself based on integrated measurement

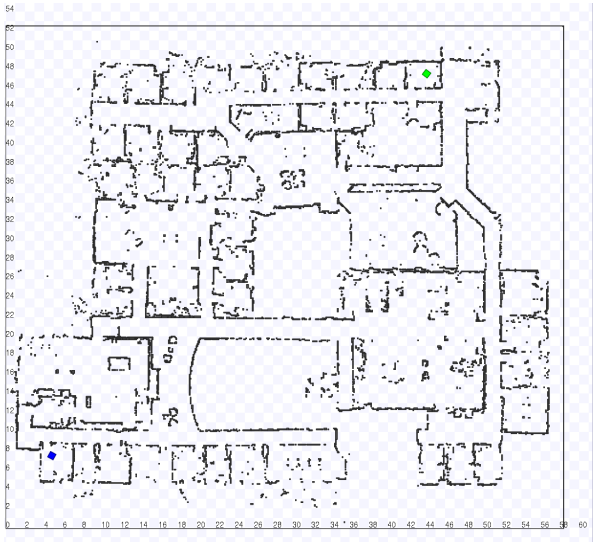

Figure 1: Robot is kidnapped from green point to blue point

rather than a single measurement by itself. They uses Markov localization to estimate robot's pose. Their approach maintains factorial representation to estimate the pose of whole team, which is the product of each robot's marginal distribution.
Both of these two examples focus on fusing measurement over every team members in order to improve localization accuracy. But fusing measurement from every robot must be very computational expensive. On the other hand, great measurement does not necessarily mean great localization. Terrible prediction will still make result unreliable.
While MACL focus on decentralized collaborative localization. Instead of fusing information from the whole team, MACL fuses information between two robots at a time. Several groups could fuse information simultaneously and independently. This is efficient especially when the team size is large.

3 Description of Multi-agent Collaborative Localization

The current version of MACL is based on the assumption that the environment is 2 dimensional. This is reasonable because most mobile robots(other than UAV) operate on flat ground.
When robot_1 is within the view of robot_0, robot_0 recognizes robot_1 and measures its distance to robot_1 using its sensor such us LiDAR. After being transferred to map coordination, the relative distance is $[\widetilde{d_x}, \widetilde{d_y}]$. Due to the inaccuracy of sensor, it has an covariance of

$$\Sigma_d = \begin{bmatrix} \sigma_d(x,x) & \sigma_d(x,y) \\ \sigma_d(y,x) & \sigma_d(y,y) \end{bmatrix}$$

At this time, robot_0's belief of its own position is of mean $[\widetilde{x_0}, \widetilde{y_0}]$ and and covariance

$$\Sigma_0 = \begin{bmatrix} \sigma_0(x,x) & \sigma_0(x,y) \\ \sigma_0(y,x) & \sigma_0(y,y) \end{bmatrix}$$

One thing to notice is that since the orientation of robot is hard to read from LiDAR, I ignore orientation and just focus on (x, y) position.
To combine these two distributions, I use the prediction step of Kalman filter. Robot_1's position estimate generated by robot_0 is of mean $\mu_{01} =$

$[\widetilde{x_0} + \widetilde{d_x}, \widetilde{y_1} + \widetilde{d_y}]$ and covariance $\Sigma_{01} = \Sigma_0 + \Sigma_d$.
At this point, Constructing robot_1's position estimation by robot_0 is finished. Then robot_0 sends this belief to robot_1.
When robot_1 receives the message from robot_0 about its position, it needs to fuse received belief with its own belief. When robot_1 receives the message, its own belief is represented as mean $\mu_1 = [\widetilde{x_1}, \widetilde{y_1}]$ and covariance

$$\Sigma_1 = \begin{bmatrix} \sigma(x_1, x_1) & \sigma(x_1, y_1) \\ \sigma(y_1, x_1) & \sigma(y_1, y_1) \end{bmatrix}$$

The way I use to fuse these two distribution is the product of two multivariable Gaussian distributions:

$$bel_{01}(x) \sim N(\mu_{01}, \Sigma_{01}) = \frac{1}{(2\pi)^{d/2}|\Sigma_{01}|^{1/2}} exp[-\frac{1}{2}(x-\mu_{01})^T \Sigma_{01}^{-1}(x-\mu_{01})]$$

$$bel_1(x) \sim N(\mu_1, \Sigma_1) = \frac{1}{(2\pi)^{d/2}|\Sigma_1|^{1/2}} exp[-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1)]$$

$$\begin{aligned}
bel_{fused}(x) &\sim N(\mu_{01}, \Sigma_{01}) N(\mu_1, \Sigma_1) \\
&= \eta exp[-\frac{1}{2}(x-\mu_{01})^T \Sigma_{01}^{-1}(x-\mu_{01})] + -\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1)] \\
&= \eta exp[-\frac{1}{2}((x^T \Sigma_{01}^{-1} x + x^T \Sigma_{01}^{-1} x) - 2(x^T \Sigma_{01}^{-1}\mu_{01} + x^T \Sigma_1^{-1}\mu_1) + (\mu_{01}^T \Sigma_{01}^{-1}\mu_{01} + \mu_1^T \Sigma_1^{-1}\mu_1))] \\
&= \eta exp[-\frac{1}{2}(x^T(\Sigma_{01}^{-1} + \Sigma_{01}^{-1})x - 2(x^T(\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1)) \\
&+ (\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1)^T(\Sigma_{01}^{-1} + \Sigma_{01}^{-1})^{-T}(\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1)] \\
&= \eta exp[-\frac{1}{2}(x^T(\Sigma_{01}^{-1} + \Sigma_{01}^{-1})x - 2(x^T(\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1)) \\
&+ (\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1)^T(\Sigma_{01}^{-1} + \Sigma_{01}^{-1})^{-T}(\Sigma_{01}^{-1} + \Sigma_{01}^{-1})(\Sigma_{01}^{-1} + \Sigma_{01}^{-1})^{-1}(\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1)] \\
&= \eta exp[-\frac{1}{2}(x-\mu_{fused})^T \Sigma_{fused}^{-1}(x-\mu_{fused})]
\end{aligned}$$

where

$$\mu_{fused} = \Sigma_{fused}(\Sigma_{01}^{-1}\mu_{01} + \Sigma_1^{-1}\mu_1) \quad (1)$$

$$\Sigma_{fused}^{-1} = \Sigma_{01}^{-1} + \Sigma_1^{-1} \quad (2)$$

Robot_1 gets its updated position estimate of $bel_{fused} \sim N(\mu_{fused}, \Sigma_{fused})$.
The reason I use product of two multi-variable Gaussian distribution to

present the fused belief is that the received belief can be viewed as $\overline{bel}(x)$, which is robot_1's position prediction by robot_0; and robot_1's belief can be seen as $p(z|x)$, where z is the position estimate of EKF or AMCL or other filter when robot_1 is located at x. Thus we use Bayes filter to get an optimized position estimate:

$$bel(x) = \eta p(z|x)\overline{bel}(x)$$

Therefore, the fuse step of MACL is a special case of the update step of Kalman filter. So we can view the whole process as a Kalman filter.
MACL is particularly efficient for kidnapped problem, because received information has certainty guarantee no matter where the robot is kidnapped to. Mathematically, robot_1' belief bel_1 has a super large covariance Σ_1 after being kidnapped. Since robot_0 has much higher certainty, Σ_{01} is very small, Σ_{01}^{-1} is absolutely larger than Σ_1^{-1}. By equation (1), μ_{01} has much higher weight, and Σ_{fused} is much smaller than Σ_1. Therefore, bel_{fused} has relatively high confidence.

4 Implementation in ROS

My implementation is based on Stage simulator [3] of Robot Operating System(ROS) [5]. The reason is: first, I do not need to find real robots; second, I can test the algorithm repeatedly in exact same environment and settings; third, a robot can communicate with another robot easily; last, I have access to many parameters of robots.
The first step is to bring up the environment. I use Stage, a 2-D simulator. I deploy willow-pr2-multi, a world file in navigation [1] package, as my robot world. The willow-pr2-multi world is shown in Figure 1.
Because all robot operate in the same world and their shared information should based on same coordination, same map is given to each robot, which is willow-full.pgm corresponding to willow-pr2-multi.world.
Next step is to set up the robots. I spawn two robots in the simulator. I give each robot its own namespace so that each robot's node and topic operate in corresponding namespace. Each robot has its own move base to move itself independently. I also set different tf_profix for each robot to distinguish transformation between coordinate frames.
To generate robot's belief of it own position, I deploy amcl [1] node in each robot. This node is the implementation of AMCL. It uses laser data to match the map, and generates the most likely position(mean) with uncertainty in

a covariance matrix. I use its node to get robot's position estimate.
To recognize another robot using laser data is hard and tedious. I use fake_localization [1] node to exact the true position of each robot. Then I use robot_1's position minus robot_0's position to get the accurate relative distance. I manually set laser data's covariance matrix. After all, how to recognize robot is not very relevant to MACL. To prevent amcl interfere with fake_localization, I set amcl's tf_broadcast parameter to false so that it would not broadcast its result to robot's odometer, and I also remap amcl's output because its has same name with fake_localization.
To keep laser scan updating, I deploy a random_walk node to keep the robot moving.
At this point, the layout of nodes should be: \map, (\robot_0\movebase, \robot_0\random_walk, \robot_0\amcl, \robot_0\fake_localization), (\robot_1\movebase, \robot_1\random_walk, \robot_1\amcl, \robot_1\fake_localization).
Each robot observes others and sends information to that robot. In the mean time, it should be prepared to receive information from others and fuse received information with its own.
Observer node listens to the observed robot's accurate position and its own accurate position from fake_localization node. It uses these two position to calculate the distance. Observer node gets its own approximate position from amcl node. It uses the approach discussed above to get a position estimate of the observed robot. After that, it sends the message to pose_comm topic. Importantly, I set message stamp to be the moment of capturing observed robot's position. I set the frame_id to be the id of observed robot to make sure the message is received by the desired robot.
Fuse node listens to pose_comm topic. Once there is a message coming, if

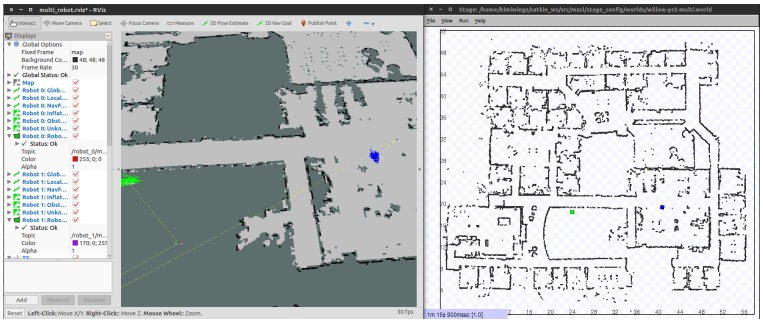

Figure 2: Rviz visualization tool and Stage simulator

the frame_id is equal to robot id, which means the message is sent to this robot, it accepts the message. Then it gets its position estimation from amcl node. It reads message time stamp of these two messages. If the difference is large, it aborts these two messages. When two messages are accepted, it uses the approach discussed above to fuse these two messages. Since the older the received message is, the less reliable it is, I enlarge the covariance matrix based on time lag. After that, the fused message is sent to initialpose topic to update AMCL process. I use Eigen library [4] to handle matrix operations. When all set, the relationship of nodes and topics should be like Figure 3.

5 Results and Analysis

I tested MACL in three scenarios. First is when two robots runs smoothly nearby, one robot receives teammate's information and combine it with its own, trying to improve accuracy. Second scenario is when one robot shifts

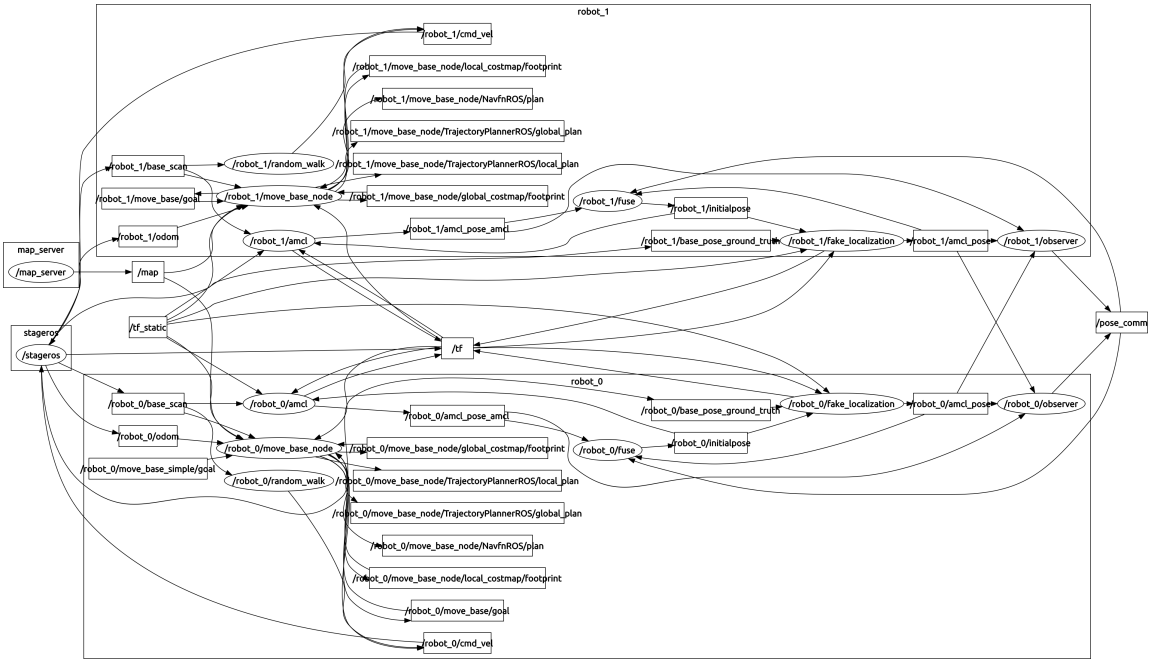

Figure 3: ROS nodes and topics relationship

due to slippery ground. It needs teammate to help it recover quickly. Third scenario is when a robot is kidnapped and unable to figure out its new place. It needs teammate to rescue it.
Figure 4 is the test result of first scenario. The robot started from the marked start point. It first went straightly to top-right. After moving about 2.2 meters, it made a sharp left turn and moved 2.5 meters to top-left. Then it turned smoothly to left. reaching bottom-left direction. At the end, it moved 3.2 meter straightly.
In Figure 4, as well as Figure 5 and Figure 6, the green line is the real path that the robot traveled, red line is the path generated from Adaptive Monte Carlo Localization, and blue line is the path combined AMCL result with the information shared by teammate.
At the very beginning, three path are very close. This is because I give each robot's initial pose to AMCL. As the robot moved, three lines begin to separate with each other. Since robot cannot follow command perfectly, it started to deviate from desired path. An important notice is that the fused path(blue) is more close to true path(green) than the AMCL one(red). This indicates that the received massage from teammate did optimize robot's position.
The second phase after turning is strange. The fused path(blue) is worse than AMCL result(red). But their difference is not big, these two lines are very close. Its teammate has its own uncertainty so that the received information is with uncertainty. It is very possible that uncertainty in received message leads to this strange pheromone. More research is needed to eliminate this pheromone.
The last phase turned out to be very ideal. Combined belief is closer to the truth. However, AMCL path(red) and true path(green) are straight but not the combined one(blue). This is because teammate is not stable during last phase, they may turn or drift. So the incoming message is changing. While no matte how it changes, the combined result is desired.

Figure 5 is the path when robot drifted. Robot turned left after starting and three lines separates at that point. Again, the strange pheromone shows up that combined result is worse than AMCL result at some time.
The robot shifted at the marked shift point. The AMCL result(red) is disturbed dramatically. It is far from true path after shift. It took the robot about 2 meters to recover its position.
However, the combined path keeps close to truth after shift. Because of

shift, AMCL result has high uncertainty, its covariance is large. While the incoming massage is still trustful. Therefore, robot put less weight on its own belief and trust much on received belief. The effect is that the combined path(blue) is very different from AMCL result(red).
When robot recovered from shift, it regained confidence to its own belief. Thus, three lines become similar and close again.

Figure 6 is the path when robot is kidnapped and put into a new position. Robot started from start point. It was kidnapped after turning around. It was dropped to the dropped point near (34, 27). Being kidnapped, robot cannot trace how it is moved to the new place. Its dropped point could be many positions on the map, shown in Figure 7.
At this point, robot had no idea about its position and did not trust its own belief almost at all. If no one tells it about its position, the only thing it could do is to guess and prove. The moment after being dropped, it guessed its position was near (34, 36). It moved a little bit and found it was im-

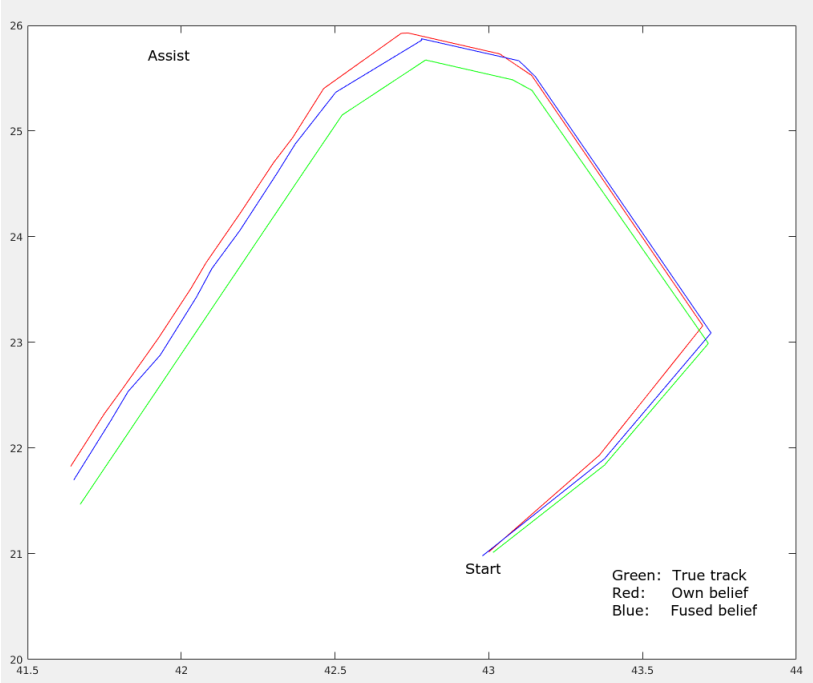

Figure 4: Smooth Run

possible to get current laser scan data near (34, 36). So it switched to (5, 10), but the guess was proven to be wrong again. It kept exploring until feel confident for its position estimate. Unfortunately, it failed in Figure 6. I am still working on AMCL to make robot able to recover from being kidnapped without external help.
On the other hand, since its teammate saw the kidnapped robot and was confident about self position, teammate computed a belief of dropped position and sent it to the kidnapped robot. Kidnapped one had no choice but trust its teammate. Fortunately, the received information was quite accurate so that combined position(blue) is very close to truth.

Overall, MACL did help robot improve localization accuracy. It exponentially fasten robot recovering from shifting and being kidnapped. In most cases, the combined belief is very close to truth.

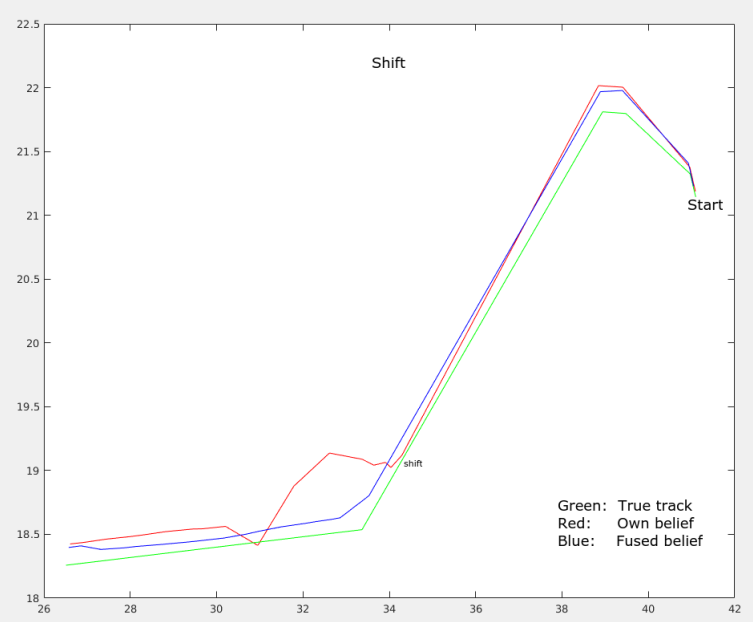

Figure 5: Recoverable Shift

6 Conclusions and Future Work

In this paper, Multi-agent Collaborative Localization was presented to improve robot localization accuracy. Robot recognizes its teammate, calculates

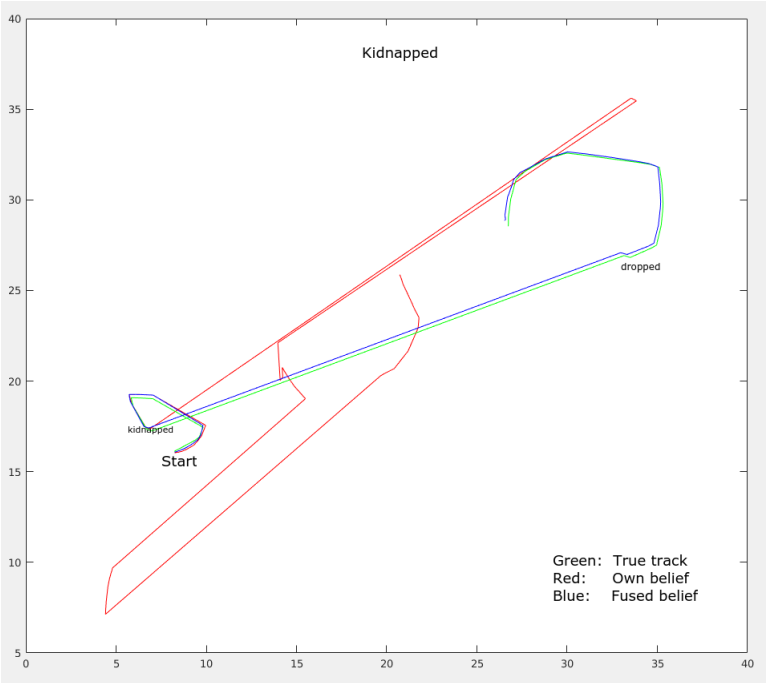

Figure 6: Being Kidnapped

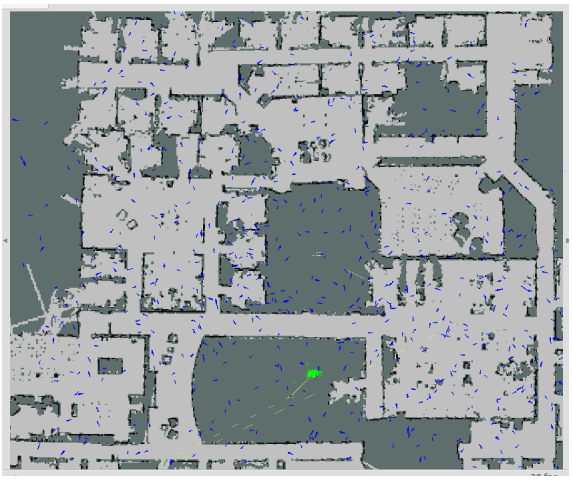

Figure 7: Particles randomly distributed

teammate's position from its view and sends it to that teammate. After receiving message from teammate, robot combines the received belief with its own, calculates a better position estimate.
The way to construct teammate's position estimate is adding its own position estimate with their relative distance. The result is that the new mean/covariance is simply sum of self position mean/covariance and distance mean/covariance.
The product of received belief and self belief is used to represent fused position estimation. The result is shown in equation (1) and (2).
Experimental results validated the assumption that MACL improves localization accuracy, especially when robot shifts or be kidnapped.
Future work includes extending MACL to 3D environment, and implementing teammate recognition from laser scan.
Extending to 3D environment is not hard. The only work is to extend mean and covariance from (x, y, yaw) to (x, y, z, roll, pitch, yaw). However, normal LiDAR will not work properly since it can only see objects in one plane. The perspective sensor must be three denominational such as stereo camera.
Implementing teammate recognition from laser scan is hard, especially when the robots in the team are heterogeneous. Alternative way could be equipping robots with pose broadcasting/receiving device.

References

[1] Navigation package in ros. http://wiki.ros.org/navigation. Accessed: 2015-12-17.

[2] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. Collaborative multi-robot localization. In *Mustererkennung 1999*, pages 15–26. Springer, 1999.

[3] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th international conference on advanced robotics*, volume 1, pages 317–323, 2003.

[4] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[5] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[6] Stergios I Roumeliotis and George A Bekey. Distributed multi-robot localization. In *Distributed autonomous robotic systems 4*, pages 179–188. Springer, 2000.