Distributed, Collaborative Human-Robotic Networks for Outdoor Experiments in Search, Identify and Track

Daniel Lee, Mark McClelland, Joseph Schneider, Tsung-Lin Yang, Dan Gallagher, John Wang, Danelle Shah, Nisar Ahmed, Pete Moran, Brandon Jones, Tung-Sing Leung, Aaron Nathan, Hadas Kress-Gazit, Mark Campbell[a]

[a]Cornell University, Sibley School of Mechanical and Aerospace Engineering, Ithaca, NY 14853

ABSTRACT

This paper presents an overview of a human-robotic system under development at Cornell which is capable of mapping an unknown environment, as well as discovering, tracking, and neutralizing several static and dynamic objects of interest. In addition, the robots can coordinate their individual tasks with one another without overly burdening a human operator. The testbed utilizes the Segway RMP platform, with lidar, vision, IMU and GPS sensors. The software draws from autonomous systems research, specifically in the areas of pose estimation, target detection and tracking, motion and behavioral planning, and human robot interaction. This paper also details experimental scenarios of mapping, tracking, and neutralization presented by way of pictures, data, and movies.

Keywords: cooperative robots, mapping, tracking, human-robotic interaction

1. INTRODUCTION

Cooperative teams of robots and humans are envisioned for use in a wide variety of future missions, including both defense applications and search and rescue operations. For example, in a burning building, robots could be sent in to explore and map the building, potentially with *a priori* information. Firefighters could then be sent in only to areas where humans were trapped and required help that the robots could not provide.

Many groups are currently addressing research for cooperative robotic, and human robotic teams. Example areas include multi-modal sensor fusion;[1] cooperative mapping,[2,3] tracking[4–6] and planning;[7,8] human-robotic interaction;[9–11] indoor and outdoor operations;[12] and ground and air operations.[13]

Distributed robotics is at a critical phase, and is now the subject of an international robotics competition entitled the Multi Autonomous Ground-robotic International Challenge (MAGIC 2010).[14] The MAGIC competition, which is an evolution from successful prior competitions such as the DARPA Grand Challenges,[15] is jointly sponsored by the Australian and US Departments of Defense with the goal of developing next-generation autonomous ground vehicle systems that can be deployed effectively in military operations and civilian emergency situations. The challenge requires robots and humans to work together to accurately and completely explore and map the challenge area, while correctly locating, classifying and recognizing all simulated threats.

The goal of this paper is to present an overview of a cooperative human-robotic system developed at Cornell University for these types of applications. The robotic hardware is based on the Segway RMP-50 platform, with lidar, vision, inertial and GPS sensors. The human can interact with the robotics in a variety of ways ranging from fully autonomous operations to high level commands such as verifying targets to drawing/gesture based commands on a tablet type of interface.

This paper is outlined as follows. First, the system architecture is presented which outlines each major component of the system. This is followed by summaries of the hardware, the indoor and outdoor localization approach, cooperative mapping and tracking algorithms, multi-level planning approaches, and human-machine interface. Finally, several example scenarios and results are detailed including mapping, tracking and indoor/outdoor operations.

For further information, send correspondence to author Mark Campbell. E-mail: mc288@cornell.edu

Unmanned/Unattended Sensors and Sensor Networks VII,
edited by Edward M. Carapezza, Proc. of SPIE Vol. 7833, 78330E
 · doi: 10.1117/12.868572

2. SYSTEM LEVEL OVERVIEW

Team Cornell's human-robotic testbed draws from experience gained in autonomous systems research and applications, specifically in the areas of pose estimation,[6] target detection and tracking,[4] motion/behavioral planning,[7,8] and human robot interaction.[11] Figure 1 gives a pictorial overview of the architecture.

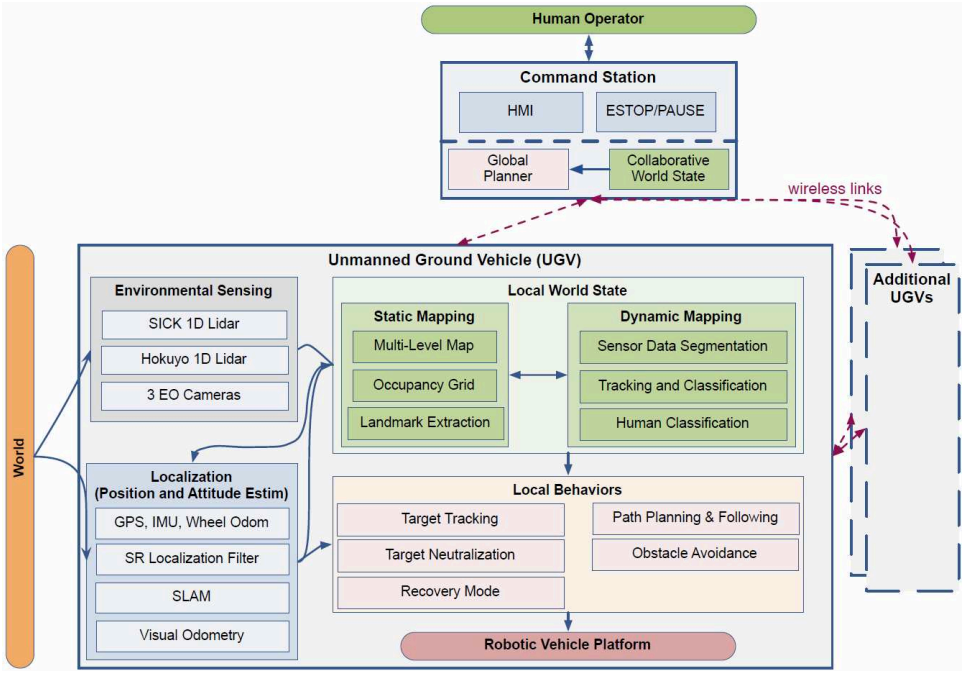


Figure 1. System architecture for Cornell's distributed human-robotic testbed.

Key elements of the architecture as as follows:

Robotic Vehicle Platform and Sensors: The testbed is based upon the Segway RMP-50 platform because of its proven durability and robustness. Environmental sensing includes SICK and Hokuyo laser rangefinders, and three EO cameras. Localization sensing is provided by an IMU, wheel odometry and a dual GPS system. A distributed Real-time Data Distribution Network[6] was developed, allowing for accurate time synchronization of data for key algorithms such as sensor fusion.

Localization: The pose (localization and attitude) estimator uses a custom Bayesian filter, supplemented with lidar based SLAM methods in order to ensure robust localization in complex terrain (e.g. during wheel slippage) and during indoor operations.

Local and Collaborative World State: An estimate of the world state, defined as both the static environment and dynamic targets, is maintained locally as well as globally (collaborative). Static terrain maps are developed using probabilistic grid fusion methods, extended to collaborative, decentralized system. An important characteristic of the static mapping is to maintain local and global representations of the maps for different functions such as path and mission planning. For dynamic mapping, a Bayesian dynamic object tracking estimator is used, including sensor fusion and collaboration across multiple robots, as well as classification. The problem of autonomous target identification is addressed using computer vision detection methods fused with lidar in a formal Bayesian tracking and classification estimator. The expert human operator validates the targets and removes false positives.

Local Behaviors and Global Planner: Existing local path planners are used over grid based static maps for obstacle avoidance and goal reaching. Custom, optimization based methods are developed for mission behaviors such as search and target tracking. Key extensions include optimization over a probabilistic

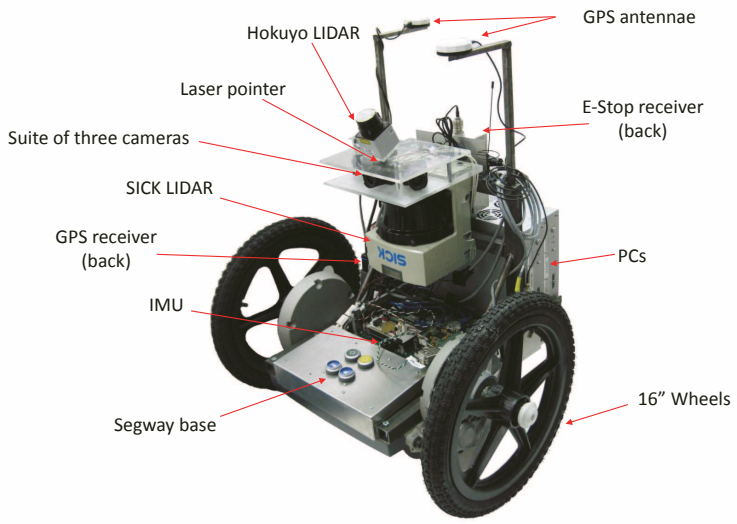


Figure 2. Hardware platform

model, and robustness in the presence of uncertainties (GPS/communication outages, or loss of a robot). Collaborative behaviors are managed at the global level, with the connections to the Human-Machine Interface (HMI).

Human-Machine Interaction: Building on previous work in human tasking of robots, a flexible interface has been designed which is capable of both Playbook and tablet style interaction. The Playbook enables strategic tasking of the team by operators including fusion of human sensory inputs across a network. The tablet PC interface allows a more natural interface to commands using drawing methods.

The subsequent sections detail each of these components, followed by experimental results of the integrated system.

3. HARDWARE

3.1 Mechanical Design

The Segway RMP-50 platform is used for its proven durability and robustness. Some modification to the original Segway RMP-50 was done to meet Multi Autonomous Ground-robotic International Competition (MAGIC) 2010 requirements. These modifications included customized steel space-frame, 16in wheels, the removal of the front caster, and removal of its charger. In addition, the robot is designed to mount various sensors, electrical components, two mini-ITX computers, and two LiPo batteries. The electronics packaging, including enclosures and connectors is designed for ease of service.

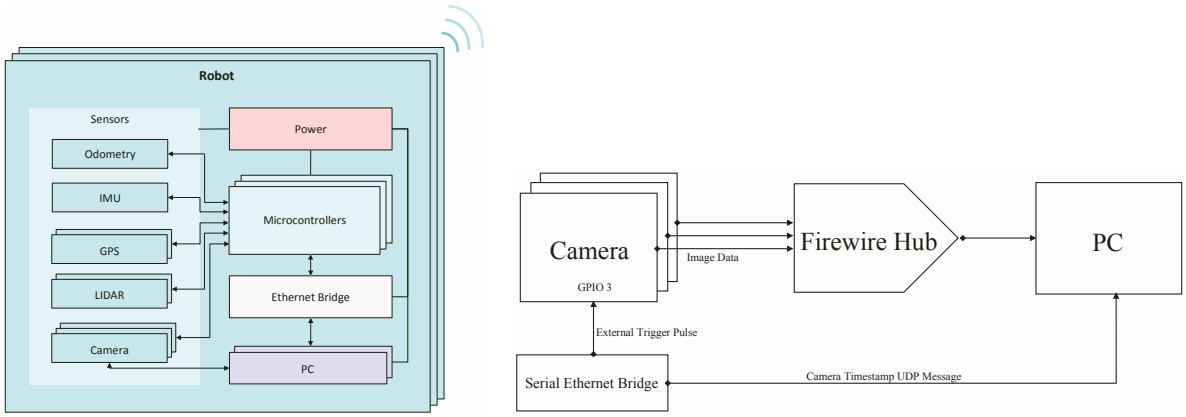
Each robot is equipped with one SICK LMS-291 and one Hokuyo URG-04LX laser range finders, three PGR Firefly MV Firewire Cameras, one MicroStrain Inertia-Link Inertial Measurement Unit (IMU), and two Septentrio AsteRX1 GPS Receivers (Figure 2).

3.2 Real-Time Data Network

In order to provide a more accurate timestamp for all of the sensors on the robot, Cornell Skynet's real-time data network technology is leveraged.[6] Each sensor measurement packet obtains its timestamp from a microcontroller. The collection of these microcontrollers is called sensor interface. In order to provide accurate timing resolution for acquired sensor data, the micro-controller units (MCUs) must be synchronized to a single timestamp. This is done by designating one MCU as a timeserver which maintains the central time base and updates other MCUs every second. All MCUs maintain their own tick counter, where each tick represents 0.1 ms. The timeserver

broadcasts both its timestamp in integer second and a pulse every second to the other MCUs on a robot. Listening MCUs synchronize their second count to this pulse, and reset their tick counts.

All sensor interfaces communicate with onboard computers through standard Ethernet UDP messages. The block diagram of the electrical system is shown as Figure 3(a). The camera is also connected to the computer via IEEE 1394 Firewire interface for image acquisition. The block diagram of camera interface is shown as Figure 3(b). Cameras are triggered externally, and the corresponding timestamp is recorded.



(a) System network (b) Camera network

Figure 3. Network scheme and camera interface design

3.3 Computers

Each robot is designed to mount two Mini-ITX form factor custom computers. Each computer is equipped with a Core 2 Duo Mobile CPU, 2GB RAM, a wireless network PCI card, and a IEEE 1394 Firewire port. It is also equipped with a solid state hard-drive to prevent hard-drive failure from vibration as the robot maneuvers over rough terrain.

3.4 Power

The power solution on the robots is based upon the Cornell Autonomous Underwater Vehicle team's solution. Its features include battery monitoring, over-current protection, under-voltage shutdown, and over-discharge protection for LiPo 5Ah batteries. A 5Ah LiPo battery is able to run for one hour of operation from a single charge. The power system also includes power merging function which allows the vehicle and elements to draw power from the battery with the most power remaining among the two connected.

3.5 Emergency Stop (E-Stop)

An E-Stop system is fitted to each of the robot in order to allow safe operation. The E-Stop system consists of three vehicle states, RUN, FREEZE, and DISABLE. Each state is triggered remotely via a dedicated E-Stop Terminal. The E-Stop system is tested with range of 300m outdoor without link-lost. Three vehicle states are required for robots to be 'failsafe'; that is, E-Stop should be triggered by the absence of a signal. The E-Stop transmitter broadcasts failsafe messages at 50Hz. A robot goes into corresponding vehicle state when a failsafe message is received on its E-Stop receiver. Upon link-lost, the robot goes into FREEZE state within 0.5 second and DISABLE after 5 second. There is also an E-Stop button on each robot to allow DISABLE state to be triggered on each robot. Table 1 shows robot behavior in three vehicle states.

4. ROBOT LOCALIZATION

4.1 Purpose and Goals

The purpose of the Localization component, henceforth termed Pose, is to form the best estimate of the robot position, attitude, and internal states possible given all of the data collected by the robot's sensor systems,

	RUN	FREEZE	DISABLE
Ethernet Bridge	ON	OFF	OFF
Communications	Normal	None between sensors and computers	None between sensors and computers
Microcontrollers	All ON	All ON but Segway interface	All OFF except E-stop receiver
Computers	ON	ON	OFF
Brake	Deactivated	Activated	Activated
Recovery	N/A	Yes	NO

Table 1. E-Stop mode

subject to computational limits. Maintaining a smooth trajectory estimate is required for robot tasks such as obstacle avoidance and local target tracking. Without an accurate history of its own position and orientation, the robot would be unable to properly merge data sets taken at different points in time, without which operation would be extremely limited if not impossible. Additionally, joint operation of multiple vehicles requires the maintenance of a single global coordinate system across multiple moving platforms. A single reference frame is essential in the construction of consistent maps from data gathered by multiple systems, as well as long-term tracking of potentially dynamic targets.

4.2 Unified Bayesian Framework

Pose estimation is accomplished by means of a single unified Bayesian framework based on the square root sigma points filter previously developed by [16, Brunke2003]. Precise measurement of the time at which each set of input data is taken allows this algorithm to run asynchronously, with pose updates occurring as soon as new data arrives. All incoming data packets are placed on a single event queue and sorted based on recorded timestamp. Packets are then processed after a slight delay, set to allow the queue to properly sort packets which may have arrived slightly out of sequence due to network delays. Data packets are treated as either process inputs, which propagate the robot state $\boldsymbol{x}_r$ forward through time, and stored in a temporary buffer, or as measurements used to update the state estimate and are immediately processed. On receipt of a new measurement packet a set of N sigma points $\boldsymbol{X}_i$ is generated based on the mean $\widehat{\boldsymbol{x}}_r$ and square-root covariance S_{xx} from the previous iteration. Each $\boldsymbol{X}_i$ is then run through the dynamics model with the buffered process inputs to bring it up to the current point in time. A predicted measurement $\boldsymbol{\zeta}_i$ is generated from each $\boldsymbol{X}_i$, with

$$\boldsymbol{\zeta}_i = h\left(\boldsymbol{X}_i\right), i \in (1, \ldots, N), \tag{1}$$

where h is a measurement function, generally non-linear, whose form depends on the type of measurement packet. If the measurement is $\boldsymbol{z}_m$, the new state estimate mean and covariance are generated as follows;

$$\overline{\boldsymbol{x}}_r = \frac{1}{N}\sum_j \boldsymbol{X}_j, \quad \overline{\boldsymbol{z}} = \frac{1}{N}\sum_j \boldsymbol{\zeta}_j, \quad X_{e_0} = \boldsymbol{X}_0 - \overline{\boldsymbol{x}}_r, \quad Y_{e_0} = \boldsymbol{\zeta}_0 - \overline{\boldsymbol{z}}$$

$$X_e = [(\boldsymbol{X}_1 - \overline{\boldsymbol{x}}_r), (\boldsymbol{X}_2 - \overline{\boldsymbol{x}}_r), \ldots, (\boldsymbol{X}_N - \overline{\boldsymbol{x}}_r)], \quad Y_e = [(\boldsymbol{Y}_1 - \overline{\boldsymbol{z}}), (\boldsymbol{Y}_2 - \overline{\boldsymbol{z}}), \ldots, (\boldsymbol{Y}_N - \overline{\boldsymbol{z}})]$$

$$K = \left(X_e Y_e^T + X_{e_0} Y_{e_0}^T \frac{W}{W_0^c}\right)\left(Y_e Y_e^T + Y_{e_0} Y_{e_0}^T \frac{W}{W_0^c}\right)^{-1} \tag{2}$$

$$\widehat{\boldsymbol{x}}_r = \overline{\boldsymbol{x}}_r - K(\boldsymbol{z}_m - \overline{\boldsymbol{z}}) \tag{3}$$

$$S_{xx} = cholupdate\left\{\sqrt{W}\cdot orth\{X_e - KY_e\}, \sqrt{W_0^c}\cdot(X_{e_0} - KY_{e_0}), sgn\{W_0^c\}\right\}, \tag{4}$$

where W and W_0^c are sigma point scaling parameters and the QR method is used for orthogonalization. After each measurement-update $\widehat{\boldsymbol{x}}_r$ and $P_a = S_{xx}S_{xx}^T$ are stored in a short-term buffer. Other system components can then calculate the robot pose at any recent time by interpolating between stored data points.

4.3 GPS-aided localization

In order to globally localize themselves, robots rely on two 4Hz Septentrio GPS receivers with antennas mounted fore and aft of the robot center. These receivers are connected via wireless link to a ground station providing

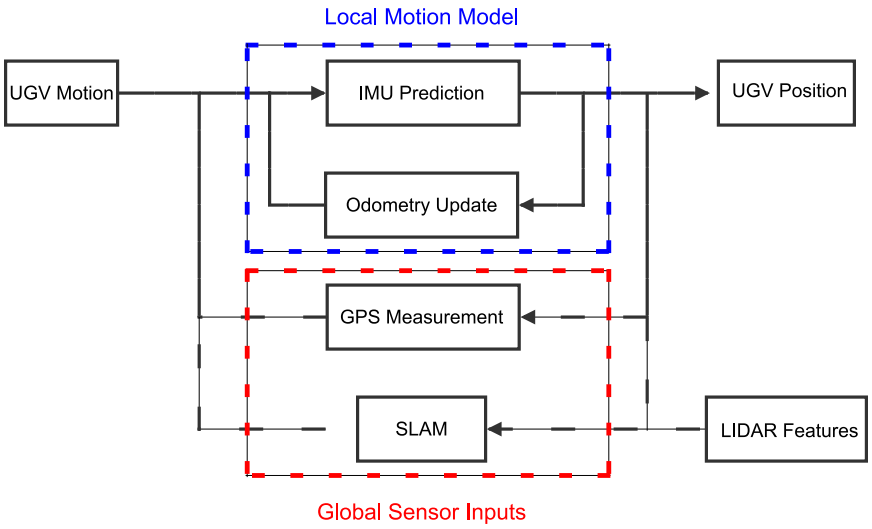


Figure 4. Location and Attitude Estimation Framework

differential corrections. While GPS errors are notoriously difficult to quantify, they can be approximated as white noise signals with intensities experimentally determined as $\sigma \approx 1m$. During outdoor operations the IMU signal is treated as a process input driving the state forward through time, while wheel odometry provides an approximate measure of forward velocity v_x and yaw-rate ω_z. In addition, soft constraints on lateral and vertical motion are imposed by adding a "pseudo-measurement" of $v_y = v_z = 0$ to each odometry measurement, with the error statistics on these measurements used as a tuning parameter.

The slowly varying biases on the IMU components are modeled as independent first order Markov processes[17] of the form

$$\boldsymbol{\beta}_i(t + \Delta t) = (1 - f\Delta t)\boldsymbol{\beta}_i(t), \tag{5}$$

where f was determined to be 0.01s^{-1}. The combination of odometry and GPS data is sufficient for the estimation of these bias terms to rapidly converge during an initialization phase of operation.

The attitude of the robot is encoded as the time-varying quaternion $\boldsymbol{q}$, a representation selected for its combination of compactness, elimination of singularities, and computational efficiency. Following [18, Crassidis2003], if

$$\Omega(t) = \begin{bmatrix} \cos(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2}) & \sigma_z \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & -\sigma_y \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & \sigma_x \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| \\ -\sigma_z \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & \cos(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2}) & \sigma_x \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta T}{2})/|\boldsymbol{\sigma}(t)| & \sigma_y \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| \\ \sigma_y \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & -\sigma_x \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & \cos(\frac{|\boldsymbol{\sigma}(t)|\Delta T}{2}) & \sigma_z \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| \\ -\sigma_x \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & -\sigma_y \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & -\sigma_z \sin(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2})/|\boldsymbol{\sigma}(t)| & \cos(\frac{|\boldsymbol{\sigma}(t)|\Delta t}{2}) \end{bmatrix}$$

the attitude can be time-updated as

$$\boldsymbol{q}(t + \Delta t) = \Omega(t)\boldsymbol{q}(t) \tag{6}$$

where $\boldsymbol{\sigma}(t) = (\sigma_x\ \sigma_y\ \sigma_z) = \boldsymbol{\omega}(t) + \boldsymbol{\beta}_r(t)$ is the bias compensated IMU rotation rate. The robot position and velocities then propagate as follows:

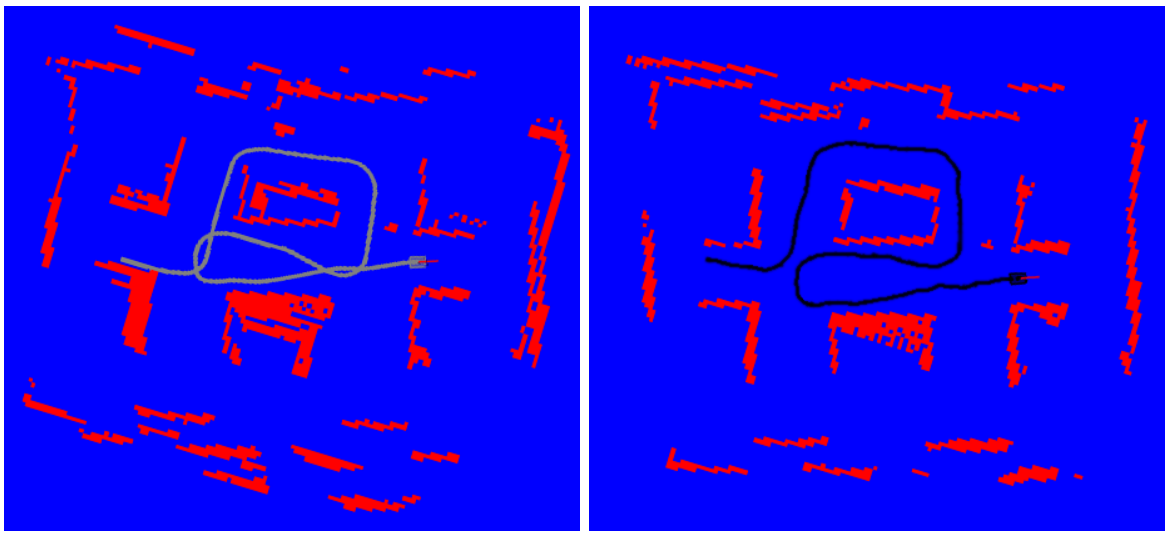
$$\boldsymbol{p}(t + \Delta t) = \boldsymbol{p}(t) + C_b^n \boldsymbol{v}(t)\Delta t \tag{7}$$

$$\boldsymbol{v}(t + \Delta t) = \boldsymbol{v}(t) + \boldsymbol{\eta}(t)\Delta t + (\boldsymbol{\sigma}(t) \times \boldsymbol{\eta}(t))\Delta t^2, \tag{8}$$

where $\boldsymbol{p}(t) = (x\ y\ z)$, $\boldsymbol{v}(t) = (v_x\ v_y\ v_z)$, C_b^n is the direction cosine matrix representation of $\boldsymbol{q}$, and $\boldsymbol{\eta}(t) = \boldsymbol{\alpha}(t) + \boldsymbol{\beta}_a(t) + C_b^n \boldsymbol{g}$ is the bias and gravity compensated IMU acceleration.

4.4 Simultaneous Localization and Mapping (SLAM)

Indoor pose estimation is performed using the same Bayesian framework as for the outdoor solution. However, while the IMU and odometry data input into the algorithm are the same, the robot state is reduced to save on computation time and to make use of a *priori* information regarding the structure of indoor environment. Specifically, it is assumed that the robot is at a constant altitude and zero pitch and roll unless the robot is traversing a ramp with a known size. Consequently, only the x, y, and yaw states are fully estimated. Roll and pitch are determined directly from integration of the IMU gyroscopes, and are reset to zero every 20 seconds to limit drift, unless the accumulated rotation indicates that the robot is on a ramp. The reduced robot state is then augmented[19] with a landmark map consisting of the (x, y) positions of detected environmental features, in order to form a combined state $\boldsymbol{x}_a$. Estimation of $\boldsymbol{x}_a$ allows the robot to continue to operate in a global reference frame in the absence of GPS data, provided that an initial global position and orientation are known. An example of an estimated robot trajectory and landmark map is shown in Figure 5.

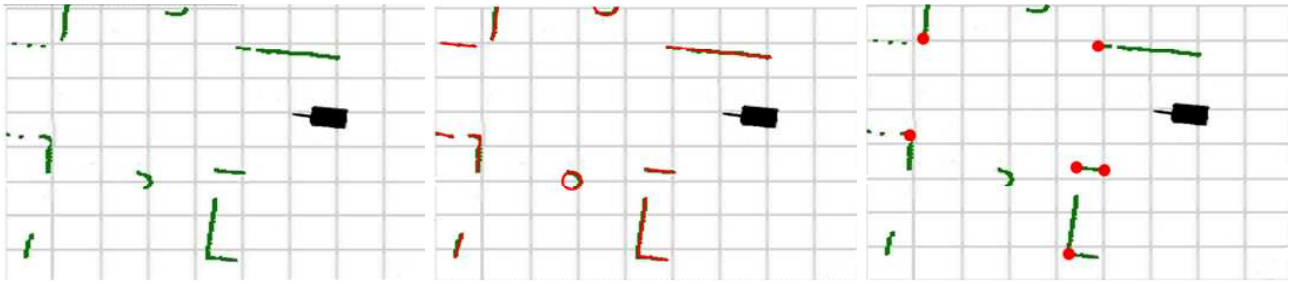


(a) Odometry Only (b) SLAM

Figure 5. Example of indoor occupancy grid comparing odometry-only integration with SLAM solution. Black and grey lines indicate estimated rover trajectory with odometry and SLAM respectively, while red cells indicate detected walls. Note the map rotation introduced by odometry errors in (a).

Map features are extracted from LIDAR data from the SICK laser range finder following the sequence developed in [20, Xavier2005] and shown in Figure 6. LIDAR points are first segmented based on discontinuities in range. At each stage of this process segments which contain less than five points are removed from consideration. Doing so allows the feature extractor to locate stable features such as corners and doors in otherwise cluttered environments, and has the added benefit of rejecting measurements of other robots, humans, or other small dynamic objects. In order to reduce spurious measurements, features are required to be within 5cm of a feature from the previous LIDAR scan before they can be considered a positive return. This minimum distance was determined based on the size of objects in the experimental environment, as well as the potential error accumulated between scans. Data association between extracted features and landmarks in the map is done via the maximum likelihood method.[19]

When a new landmark is detected, $\boldsymbol{x}_a$ is augmented with the measured 2-D position of the landmark and S_{xx} is augmented with a new 2x2 diagonal square-root covariance matrix whose elements are set to an arbitrary large number. If the size of the map has grown beyond a set cutoff value, determined by available computational resources, the landmark with the fewest measurements whose age is greater than 10 seconds is removed from the map, and $\boldsymbol{x}_a$ and S_{xx} are truncated as required. This has the effect of removing spurious measurements that have been added as landmarks and preserves landmarks close to the robot's current position. While this map management strategy tends to maintain a good landmark density in the robot's immediate vicinity, it



(a) Raw LIDAR returns (b) Line and circle determination (c) Corner location

Figure 6. Feature extraction sequence.

unfortunately does not accommodate large scale loop-closure. As a consequence, the global coordinate system encoded in the estimated robot position and heading tend to slowly drift away from that of other vehicles in large indoor spaces. Measurement updates use the same form as in Equation 2, with $\boldsymbol{z}_m = (r_1\ \theta_1\ r_2\ \theta_2 ...)^T$ as the measurement vector, where r_i and θ_i are the range and bearing to feature associated with the i^{th} landmark.

5. MAPPING

Robots need to have some understanding of the environment not only for local-level planning and obstacle avoidance, but also for global planning for multiple robots. For a static environment representation, a global height map is constructed based on the data collected from multiple robots using Gaussian mixture model. Objects of interest (OOIs) and dynamic objects are detected using vision detection algorithm, localized using LIDAR measurement, and tracked using square-root sigma point information filter. A list of these dynamic objects is stored in order to allow the removal of corresponding laser returns. The following subsections discuss mapping algorithm, vision detection system, and target tracking.

5.1 Mixture-Model Based Terrain Estimation

5.1.1 Algorithm

Each robot is equipped with two laser range finders (LRFs), a forward-facing SICK LRF, and Hokuyo URG LRF mounted at 25 degree downward pitch angle. Based on robot pose and sensor noise estimation, a fully probabilistic approach for height estimation is accomplished using mixture-model based terrain estimation. The algorithm was introduced by Miller and Campbell[2] to translate laser scanner measurement into an elevation distribution over multiple cells on a planar grid. A large occupancy grid is maintained in order to store the history of the global map.

Based on current robot position and its relative sensor positions, each lidar scan from two LRFs is represented in the East, North, Up (ENU) global frame via a measurement function, $f(\hat{p}, \hat{r})$, where $\hat{p}$ represents robot pose and $\hat{r}$ represents the noisy sensor measurement. The covariance of this measurement function, P_{ENU} is obtained using a Taylor expansion and the Jacobian of the transformation.

$$\hat{r_i}^{ENU} = f(\hat{p}, \hat{r_i}) = \begin{bmatrix} \hat{e} \\ \hat{n} \\ \hat{u} \end{bmatrix} \tag{9}$$

$$P_{ENU} = E\big[(r^{ENU} - \hat{r}^{ENU})(r^{ENU} - \hat{r}^{ENU})^T | I\big] \tag{10}$$

$$= \begin{bmatrix} P^{EN} & P^{EN,u} \\ P^{u,EN} & P^{u} \end{bmatrix} \tag{11}$$

The EN coordinate of a laser return is truly associated with a single grid cell. However, due to sensor noise and uncertainty in pose, it is necessary to compute the probability that the lidar return is associated to each grid cell in the map. Assuming the ENU probability distribution, $p_i(E, N, U)$, is known, the probability of each cell to associate to the i-th laser return can be computed by integrating $p_i(E, N, U)$ over the area covered by the grid cell.

The vertical coordinate is marginalized out to yield the in-plane distribution, $p_i(E, N)$, and this distribution is approximated as a bivariate Gaussian.

$$p_i(E,N) = \int_{-\infty}^{\infty} p_i(E,N,U)dU = \int_{-\infty}^{\infty} \mathcal{N}(\hat{p},\hat{r})dU \approx \mathcal{N}(\hat{p},\hat{r}) \tag{12}$$

Then, the association probability of each cell is computed by integrating over the area of the j-th cell

$$Pr(\hat{r}_i \in j) = Pr(E_j, N_j) = \int_{E_{j-}}^{E_{j+}} \int_{N_{j-}}^{N_{j+}} p_i(E,N)dNdE \tag{13}$$

$$= (\Delta E)(\Delta N)\ p_i(E_j, N_j) \tag{14}$$

While the distributions are captured in continuous space, the grid based approach utilizes discretized spaces as shown in Equation (14). While the distribution in theory covers the entire grid, computing the integral for every single cell is too expensive for real-time implementation. Therefore, the range of cells to update is limited to be within 2σ boundary based on P_{EN} term in Equation (11).

The original measurement estimates $\hat{r}$ are uncertain in three axes: East, North, and Up. In order to improve computational cost, univariate distribution of elevations at a particular cell is desired for this adopted grid-based approach. The posterior univariate elevation estimate $\hat{\mathcal{U}}_i = E\left[\hat{u}_i | E, N\right]$ and its variance $\sigma^2_{\hat{U}_i}$ are determined.

$$\hat{\mathcal{U}}_i = \hat{u}_i + P^{u,EN}(P^{EN})^{-1}\left[\begin{pmatrix} E_{j_c} \\ N_{j_c} \end{pmatrix} - \begin{pmatrix} \hat{e}_i \\ \hat{n}_i \end{pmatrix}\right] \tag{15}$$

$$\sigma^2_{\hat{U}_i} = P^u_i - P^{u,EN}_i \left(P^{EN}_i\right)^{-1} P^{EN,u}_i \tag{16}$$

where E_{j_c} and N_{j_c} represent the center coordinate of j-th cell in East and North frame. For each timestamp, the height distribution is computed for each affected cell, for each laser measurement. Since the elevation is represented using a Gaussian distribution, the height distribution for j-th cell $\mathcal{P}(\mathcal{U}_j | \hat{p}_{1,\ldots,M}, \hat{r}_{1,\ldots,M})$ conditioned on the full sets of laser and pose measurements up to time M is

$$\mathcal{P}(\mathcal{U}_j | \hat{p}_{1,\ldots,M}, \hat{r}_{1,\ldots,M}) = \frac{1}{c_j} \sum_{i=1}^{M} Pr(\hat{r}_i \in j)\mathcal{N}(\hat{\mathcal{U}}_i, \sigma^2_{\hat{U}_i}) \tag{17}$$

where $c_j = \sum_{i=1}^{M} Pr(\hat{r}_i \in j)$. However, this is computationally infeasible as the height distribution keeps growing over time. Thus, considering only the first and second moments of this Gaussian mixture, the terrain height is represented by its mean and covariance, or

$$\hat{\mathcal{U}}_{GM_j} = \frac{1}{c_j} \sum_{i=1}^{M} Pr(\hat{r}_i \in j)\hat{\mathcal{U}}_i \approx E\left[\mathcal{U}_j | \hat{p}_{1,\ldots,M}, \hat{r}_{1,\ldots,M}\right] \tag{18}$$

$$\sigma^2_{GM_j} = \frac{1}{c_j} \sum_{i=1}^{M} Pr(\hat{r}_i \in j)(\hat{\mathcal{U}}^2_i + \sigma^2_{\hat{U}_i}) \ - \ \hat{\mathcal{U}}^2_{GM_j} \tag{19}$$

5.1.2 Implementation

In order to generate a high resolution map, small grid cell sizes are required. However, updating a larger number of cells for every laser measurement is computationally expensive, especially online. In order to reduce the computational burden and construct a global map from multiple robots, a central processing unit is designed to receive packets containing laser returns, sensor pose, and robot pose data from all robots in order to update the global map. The central planner converts this global height map, $\mathcal{O}(x, y)$, into a drivability map by thresholding with a constant, $\sigma_{\text{threshold}}$. A 2D thresholded binary global map is generated by the same mean.

$$\mathcal{O}_{\text{threshold}}(x,y) = \begin{cases} 1 & \text{if } \mathcal{O}(x,y) \geq \sigma_{\text{threshold}} \\ 0 & \text{if } \mathcal{O}(x,y) < \sigma_{\text{threshold}} \end{cases} \tag{20}$$

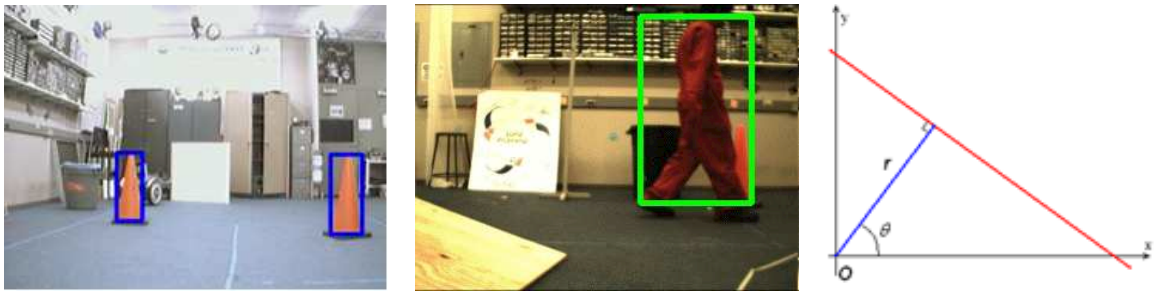
The SICK laser mounted facing forward contributes to detection and mapping of obstacles far from the robot while the Hokuyo URG laser is the main contributor for terrain estimation. As the robot moves, the URG laser acts in a "push broom" fashion, filling in the terrain estimate as if brushes past.

The LRFs occasionally have returns from other cooperative robots and dynamic objects. These measurements leave their trajectories on the built map unless a robot goes to the updated area and re-updates with Hokuyo LRF's measurement. To prevent this from happening, 1) other robots positions are known by the central processor and 2) objects of interest are tracked and localized to ignore laser return from them, so that the global map is not updated by any of the laser return from other robots and OOIs.

Features such as exploration and obstacle avoidance are improved with additional information from other robots. For example, a robot plans more efficiently if it knows where other robots have already explored. Therefore, each robot requests additional map information from central processing unit. The central processing unit will compare the requesting robot's local map with the constructed global map and extract the difference. This information is passed back down to the local map of the robot for its own planning.

5.2 Vision Detection System

Vision detection is required in order to distinguish difference among moving objects as well as to detect OOIs. An algorithm is developed to detect and locate all OOIs which are pre-defined in color and shape within the field of view of the robot.



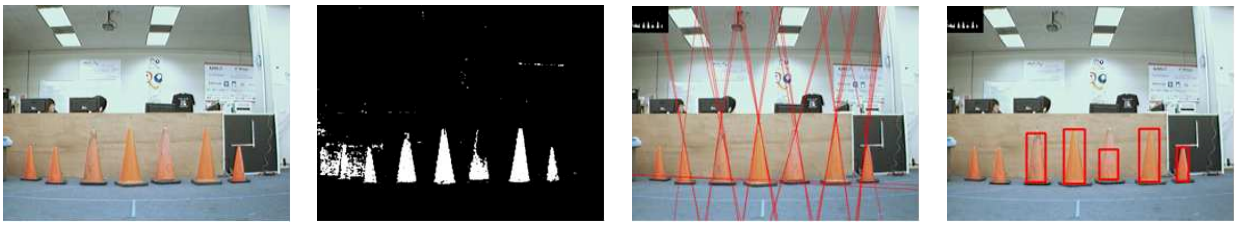
(a) Static OOI Detection (b) Mobile OOI Detection (c) Angle θ calculation

Figure 7. Vision detection of static and mobile OOI. Bounding box information are sent to tracking module to localize and track OOIs

In order to achieve robust OOI detection, a fully trained machine learning algorithm is used to extract features regardless of OOI's scale and point of view. The vision algorithm is trained with a number of still pictures of targets at different points of view. The static OOI detection algorithm is written in following way: First, filter original image using a range of color space which then generates a binary picture as in Figure 8(b). Then, small isolated regions are removed. If the algorithm uses classification, it fits straight lines to each region and compute angle θ (Figure 8(c), 7(c)). If θ is within the specified threshold values, then it is detected as a static OOI. A mobile OOI is defined to be a person in red jumpsuit. While the static OOI can be detected only using color and shape classification, this jumpsuit requires a trained set of data to be detected (Figure 9) since the suit has a different shape at different points of view. The detection system also implements auto white-balance and auto gain features for robust target detection in various environment, since colors depend heavily on lighting condition.

5.3 Probabilistic Target Tracking

Objects of interest are categorized into two types: static and mobile. Based on detections from the vision system, the tracking module is initialized and recursively updated. Targets are localized accurately with respect to the robot local reference frame using SICK LRF. The tracking algorithm uses both vision and laser return data for localization, and a square root sigma point information filter to update target states.



(a) Original image (b) Binary image (c) Line fitting (d) Detection output

Figure 8. Static OOI detection process

Figure 9. Robot learned from a number of still images of mobile OOI

5.3.1 Algorithm

The algorithm for tracking was introduced by Campbell and Whitacre[4] for cooperative tracking using vision measurements on UAVs. A sigma point filter is used because it is computationally less demanding, but still generates better results than the extended Kalman filter for nonlinear systems.[16]

Assuming the dynamic model and the measurement function is known, the decentralized system has the form:

$$x_{\mathrm{POI},k+1} = x_{\mathrm{POI},k} + f(x_{\mathrm{POI},k}, \omega_k)\Delta T \tag{21}$$

$$z_{\mathrm{SCR},k+1} = g_{\mathrm{SCR}}\left(x_{\mathrm{POI},k+1}, \begin{bmatrix} x_{\mathrm{NAV},k+1} \\ x_{\mathrm{ATT},k_1} \\ x_{\mathrm{GIM},k+1} \end{bmatrix}, \upsilon_{\mathrm{SCR},k+1}\right) \tag{22}$$

where $x_{\mathrm{POI},k}$ represents state of possible object of interest at time k, ω_k and υ_{SCR} are zero mean white process and sensor noise, x_{NAV}, x_{ATT}, and x_{GIM} represents vehicle position, attitude, and camera gimbaled position, respectively. z_{SCR} represents the pixel information of the center of the detection box (Figure 7(a)), $[u_{\mathrm{pix}}\ v_{\mathrm{pix}}]^T$.

For the prediction step, each of the sigma points generated from the target state propagates through the dynamics model, Equation (21). These predicted sigma points are then augmented with sigma points drawn from current vehicle state and measurement uncertainty. Then each augmented sigma point propagates through the measurement function, Equation (22), which output predicted sigma points of detection. Using the state-measurement cross covariance, the nonlinear measurement function can be linearized. Then, the square root information state and gain are calculated to compute the updated target state and corresponding covariance matrix.

5.3.2 Implementation

Since the range to an object from the robot is difficult to estimate solely based on a camera pixel data, the axis of the covariance circle aligned with the robot's orientation decreases slowly compared to the other axis. As additional measurements are taken from different positions, the target is localized more accurately. However,

laser measurement can improve the tracking performance significantly by adding another measurement state which is orthogonal to the camera measurement. Thus, the augmented measurement vector, z_{SCR}, takes a form:

$$z_{\mathrm{SCR}} = \begin{bmatrix} u_{\mathrm{pix}} \\ v_{\mathrm{pix}} \\ r_{\mathrm{laser}} \end{bmatrix} \tag{23}$$

where r_{laser} represents the range return from the laser scanner.

The performance of tracking algorithm heavily depends on the initial ENU location of the target, which is computed using laser return data. The range to the target is found by projecting the laser points on the camera image using projection matrices. Laser points within the bounding box (Figure 7(a)) are divided into clusters of laser points based on the continuity along with the neighboring points. Among the clusters, the largest one is selected and the range measurement of its middle laser point is returned.

One problem to arise is data association, i.e. how to tell whether a detected target is one which the robot has already seen. Doing so with only ENU location and a distance threshold tends to cause spawning of new targets near existing targets due to pose uncertainties. Thus, the association is accomplished by predicting the states of existing targets based on current pose uncertainty using Equation (22). The only difference is that the input of this measurement function for prediction is $[r_{\mathrm{laser}}\ \theta_{\mathrm{laser}}]^T$ and the output is predicted target states in ENU frame, not in camera space, with a modified measurement function, g'_{SCR}. A nearest neighboring gating test[17] is done with this predicted state and covariance matrix in ENU coordinate for the new detection. If the new detection point is sitting within 3σ bounding circle of a existing target, the detection point is associated to the current target.

$$(z_i - s_j)^T \bar{\Sigma_j}^{-1} (z_i - s_j) - g^2 = f(z_i, s_j) \tag{24}$$

In the gating equation, Equation (24), g equals 3 in this 3σ gating test, z_i represents the new detection ENU measurement, and s_j represents predicted ENU state of existing target j. The covariance keeps decreasing with additional measurement updates. However, once the covariance ellipse is smaller than the actual target, the algorithm has high potential to spawn new targets. Therefore, a floor was placed on the 3σ bound such that it cannot shrink smaller than the actual target size. This effect is later shown in the example experiment section (Section 8).

6. PLANNING

Planning cooperative behaviors for multiple robots is done at two levels: global and local. The global planner is responsible for dynamically allocating tasks to robots, while a local planner on each robot is responsible for executing assigned tasks and feeding the state of the local planner back to the global level. Two types of tasks can be sent to a robot by the global planner: waypoint or state. If a local planner receives a waypoint task, it uses path planning and following to achieve that point in space. The state task, on the other hand, allows the global planner to command a robot into a state that corresponds to a specialized behavior. Specialized behaviors at the local level allow a robot to perform tasks that cannot be accomplished with the delay of communicating with the global planner.

The global planner is composed of a data layer and a state layer. Data from the collaborative world state and the local planners is assembled in the data layer. Information in the data layer is used by the state layer to determine the state of the robots and objects of interest. Each robot state is associated with an algorithm that has access to all information known to the system, and it uses that information to assign a task to a robot. For example, in the process of exploration a robot is represented in the state layer as having an exploration state with a self-loop that is triggered when the robot completes a waypoint. When the state messages from the local planner indicate that a waypoint has been completed, the state layer re-enters the exploration state and calculates the next waypoint using information from the other robot planners and the collaborative world state.

The goals of the global planner are exploration and neutralization. Two different types of robots are used to complete these goals: explorers and destructors. Explorer robots participate in exploration and neutralizing

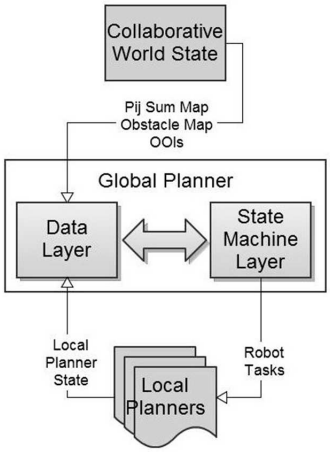


Figure 10. The two tiered planning system separates task allocation from path planning and following

mobile objects of interest while destructor robots are only responsible for neutralizing static objects of interest. From the perspective of the global planner, neutralization is the process of sending an appropriate robot to a place where it can neutralize the target and then putting the robot in a specialized neutralization mode. Exploration is the process of maximizing map discovery while finding objects of interest in an unknown environment with dynamic obstacles.

Traditional multi-robot exploration algorithms, such as the multi robot exploration algorithm in Probabilistic Robotics,[19] have robots set waypoints in nearby unexplored areas and then prohibit other robots from setting their waypoints nearby. This approach does not inherently prevent conflicts, and methods for reducing or preventing conflicts have to be added. For this reason, a new exploration algorithm called Finite Space Horizon (FiSH) exploration was developed.

The Finite Space Horizon exploration algorithm is a grid based exploration algorithm that works by establishing a finite space around a robot and then choosing an optimal waypoint in that space. Therefore, choosing a waypoint for a robot is a two-step process. First, a set of possible finite spaces, S*, are examined. Possible finite spaces are those that contain the robot and do not overlap with the finite space of any other robot. The possible finite spaces can be hardcoded as functions of the robot's current position. From S*, the best finite space, S', is chosen. The distance between the possible finite space and the other location of the other robots and the unexplored value of the space (represented by the pij sum of the cells in that space) are used to choose the best finite space. This process is represented by the following function:

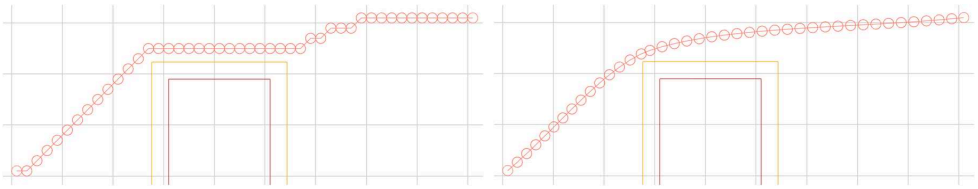
$$\min\left\{\frac{\sum_{(E,N)\in S} Pr(E,N)}{\max\{Pr(E,N)|(E,N)\in S\}} - \frac{\sum_{R\in R^*} Distance(S,R)}{\max\{Distance(S,R)|R\in R^*\}}|S\in S^*\right\} \tag{25}$$

where R* is the set of the locations of the other robots. Note that distance and pij values are normalized before being compared.

Second, choosing the optimal point within S' can be done in many ways. While D*, A*, or Dijkstra's algorithms can all be used, it can be done simply by finding the point in S' furthest away from the robot. It appears that choosing the furthest possible finite space with a point path to the robot is sufficient.

The FiSH algorithm reduces the likelihood of confliction because robots are likely to operate in a finite area in which there are no other robots. However, because path planning is done by the local planners, there is no guarantee that robots will stay within their finite space. Feedback from the local planner allows the global planner to detect when the robot has stopped moving and plan a new waypoint.

The local planner constantly runs the path planning algorithm with the waypoint from the global planner as the goal. To plan a path, a collection of bloated polygons extracted from a short history of laser returns are painted to an occupancy grid. The occupancy grid is then blurred to add additional traversal cost to cells in the immediate vicinity of polygons. Based on the occupancy grid, an 8-connected A* graph search algorithm[21] iterates over the occupancy grid to return an optimal path to a given waypoint. However, because the A* algorithm can only explore in 8 discrete directions, we smooth the A* path by collapsing redundant waypoints returned by the A* algorithm and constructing Bezier curve based paths from the collapsed paths.



(a) Raw output from the A* algorithm when planning from the bottom left corner to the upper right corner with an obstacle in between

(b) Segmented Bezier curve built from the output from the A* algorithm

The planned paths are then fed into a modified vector polar histogram (VPH) controller which provides motor actuation commands to the Segway platform. The VPH controller continuously aims for a look-ahead point on the path a set distance away. Various concepts from the Enhanced Vector Field Histogram method (VFH+)[22] and the Vector Polar Histogram method (VPH)[23] are employed, including LIDAR measurement history, candidate valleys, threshold hysteresis, and polar histogram smoothing.

Further development on the planning system will focus on optimizing planning for faster exploration. Instead of waiting for robots to complete waypoints, the global planner will continuously re-plan exploration waypoints so that the robots do not stop at the end of their paths. The local planner will use the D* path planning algorithm instead of A* because D* dynamically re-plans and does not have to be run continuously. Finally, the local planner will plan over the map used by the global planner so that it has greater information about the environment and can plan more effectively.

7. HUMAN-MACHINE INTERFACE

The human-machine interface (HMI) serves three main purposes: to maintain operator situation awareness (SA), to facilitate natural and efficient interaction, and to support system flexibility and scalability. The current HMI solution consists of a double-monitor display (Figure 11) and the hardware E-STOP described in Section 3.5. One screen (referred to here as "Map Form"; left of Figure 11) displays a global view of the environment in which the robots are operating, including all robot poses and current statuses, detected obstacles, and dangerous areas such as Object of Interest (OOI) blast zones. The other screen (called "Camera Form"; right of Figure 11) facilitates more specialized interaction, such as viewing robot camera feeds, examining the local environment around an individual robot, teleoperating robots, and OOI confirmation and neutralization.

Using the Map Form shown in Figure 11 (Left), the operator can send commands to the robot team using several available tools, which are activated by clicking a button at the top of the screen or by right-clicking and choosing the tool from the context menu. `Path Tool` is used to send a desired trajectory to selected robots by placing waypoints in global coordinates. This enables the operator to help the Global Planner (described in Section 6) carry out the mission objectives, such as sending robots into areas that operator wishes to explore or placing a destructor robot in a strategic location for future neutralization. `Ruler Tool` and `Angle Tool` can be used to measure relative locations of objects in the environment, for example to determine how far a robot is from a potentially dangerous area. The operator can also place reference points on the map using `Point Tool` to keep track of objects or locations in the environment. Lastly, `Sketch Tool`, first introduced in [11, Shah2010], is a fully probabilistic command interface for controlling robots using multi-stroke sketches. After the operator sketches a series of commands (for example, drawing an "X" sets a single waypoint), an online sketch recognition

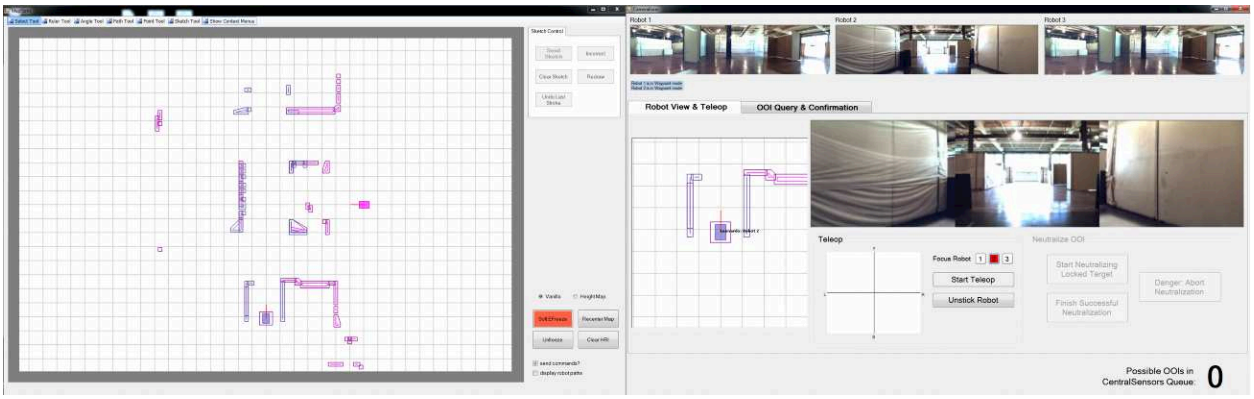
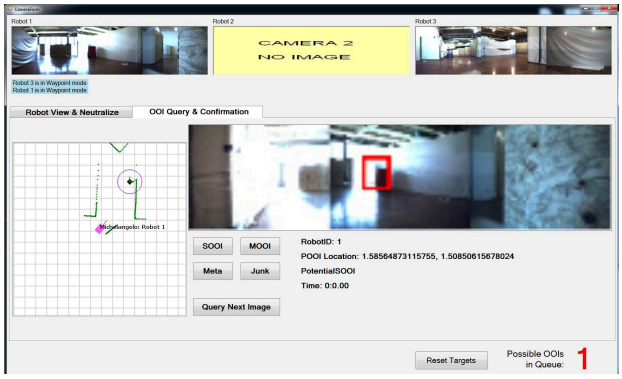
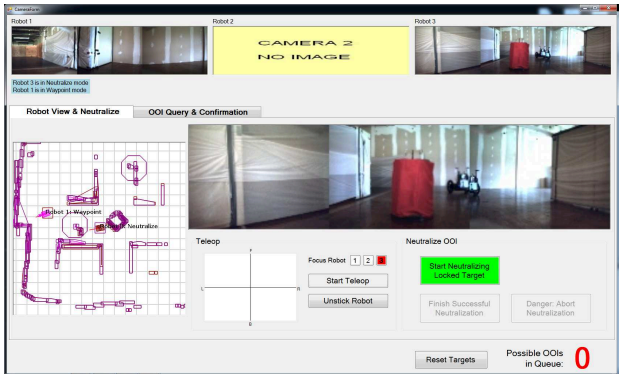


Figure 11. Double-monitor human machine interface (HMI) display.

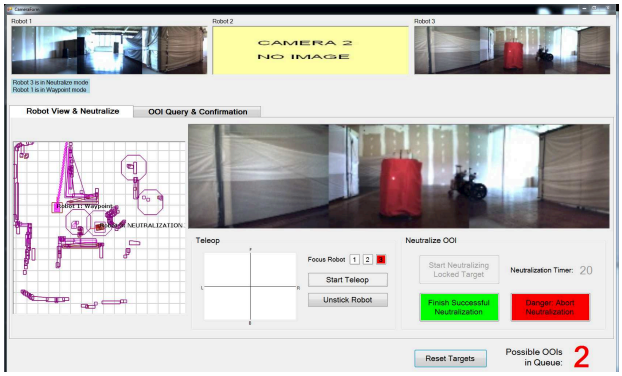
algorithm finds the most likely sketch and displays to the operator for confirmation. If the sketch has been incorrectly interpreted, the operator can either re-draw the sketch or press the "Incorrect" button to display the next-most likely sketch. Upon confirmation, the corresponding commands are executed. The distributions used by the sketch recognition algorithm are learned from training data, and can therefore be trained on many users for robustness, or on a single user for increased recognition accuracy if the intended operator is known. Currently, `Sketch Tool` can be used to select robots, send single waypoints or paths, and place points on the map. In the future, higher-level "playbook-style" commands will be available via this interface, to take advantage of humans' ability to strategically plan.



(a) Object of Interest Confirmation



(b) Robot locking on to a SOOI



(c) Neutralization Process

Figure 12. Camera Form of HMI view in confirmation, before neutralization, and during neutralization state

The Camera Form, shown in Figure 11 (right), is used for lower-level interaction with individual robots.

Streaming camera images from all robots are displayed at the top of the screen; the operator can click a robot's camera view to select it as the Focus Robot. The Focus Robot's camera stream is enlarged and displayed in the center of the screen, as well as a local scrolling map centered at the robot's location. This allows the operator to gain a better understanding of the environment around a particular robot. In rare cases when the operator must control a robot directly, a teleoperation panel also provides joystick-like control of the Focus Robot. Teleoperation is most helpful when the robot is "stuck" or can otherwise not be safely controlled by the on-board planner.

Confirmation and neutralization of OOIs is also handled using the Camera Form. If the Focus Robot is locked on and prepared to neutralize a confirmed OOI, the "Start Neutralizing Locked Target" button is activated, which the operator presses to initiate neutralization. Once neutralization has commenced, the operator can abort the neutralization process (for example, if a non-combatant enters the blast zone) by pressing the "Danger: Abort Neutralization" button. When the OOI has been neutralized, the operator presses the "Finish Successful Neutralization" button to signal to the Focus Robot that the OOI is no longer a threat and to remove the corresponding blast zone from the map.

The number of unconfirmed or unlabeled OOIs is displayed on the bottom-right of the Camera Form. At any time, the operator may choose to label these potential OOIs in a separate tab, shown in Figure 7(a). On the right, the camera frame is shown from the time the object was first detected, and a red box is drawn around the object. On the left, the lidar returns from the robot's local environment at the time of detection is also displayed with a black diamond indicating the estimated location of the object. The operator must then classify the potential OOI as a mobile Object of Interest ("MOOI"), static Object of Interest ("SOOI"), meta-data object ("Meta"), or false alarm ("Junk"). If the potential object is associated improperly (i.e. the black diamond indicating the object's location is incorrect), the object is also labeled as "Junk". When a potential OOI is labeled, the next object in the queue is automatically displayed.

8. EXAMPLE SCENARIOS AND RESULTS

This section demonstrates the system running in both indoor and outdoor environments. The objectives of the experiments were to successfully map the environment, and to detect, find, and neutralize detected objects of interest (OOIs), while avoiding obstacles. Two different types of robots exist; an explorer which maps the unknown environment and detects any potential OOIs, and a destructor which neutralizes the detected OOIs.

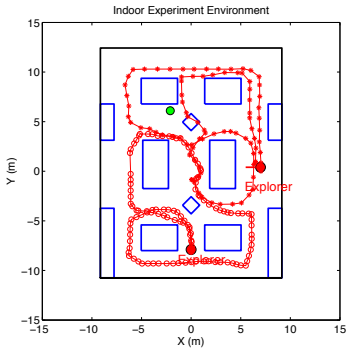
8.1 Indoor Experiment

Two robots are deployed in an indoor experiment setup shown in Figure 13. There are two static OOIs with obstacles for robots to avoid. The Robots were using integrated odometry and IMU measurement for pose solution. Two experiments were conducted for indoor set-up; one for indoor mapping and the other for target neutralization.

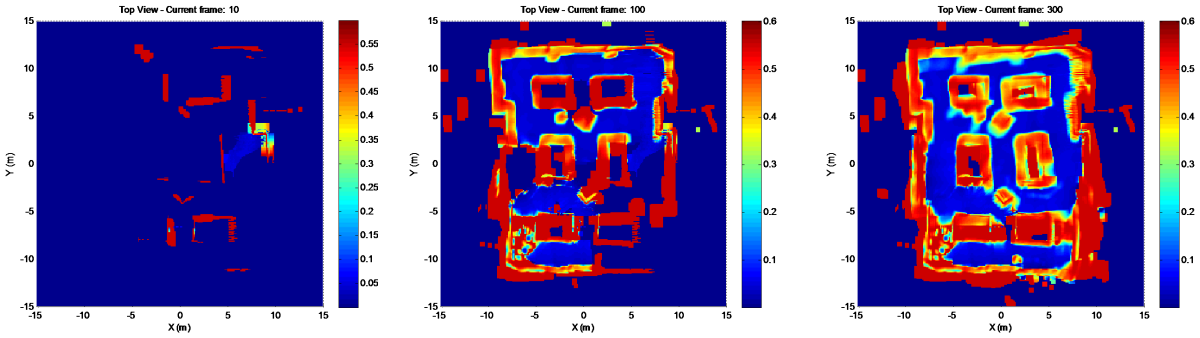
8.1.1 Indoor Mapping

Two robots equipped with two different LRFs (Figure 2) are set up in the environment as shown in Figure 13. In this mapping scenario, there is no destructor; both are explorer robots to demonstrate cooperative mapping. The two robots start at two different locations, as shown in Figure 13(b). Red circles denote robots' initial positions with a line for initial heading. The figure also shows the paths followed by each robot. Each robot is given a path from an operator to efficiently explore the environment.

Figure (c - e) show mapping results at three different times: 0, 300, and 600 seconds. Over the 10 minutes data run, the evolution of the map is quite clear, with the major components such as walls and obstacles clear. Because the consequence odometry and IMU combined pose was used, the localization degrades over time. The consequence is that the pose of one robot begins to accumulate on offset in yaw angle over time, leading to a slight smearing of the map. An additional movie showing real-time updates of mapping is available at `http://www.youtube.com/watch?v=7ojH1Tz02KM`. The SICK LRF initially causes the map to be updated with constant height. However, the Hokuyo LRF data, because it is inclined, clears the area with the correct height as the robot moves. Note that the bottom of the map in the range of $y \approx [-9, -6]$ is not correctly updated by Hokuyo LRF. This is because the floor texture of the indoor space of that area is different from the other area



(a) Picture of the indoor environment (b) Schemetic of the indoor experiment environment for mapping



(c) Mapping at $t \approx 20s$ (d) Mapping at $t \approx 200s$ (e) Mapping at $t \approx 600s$

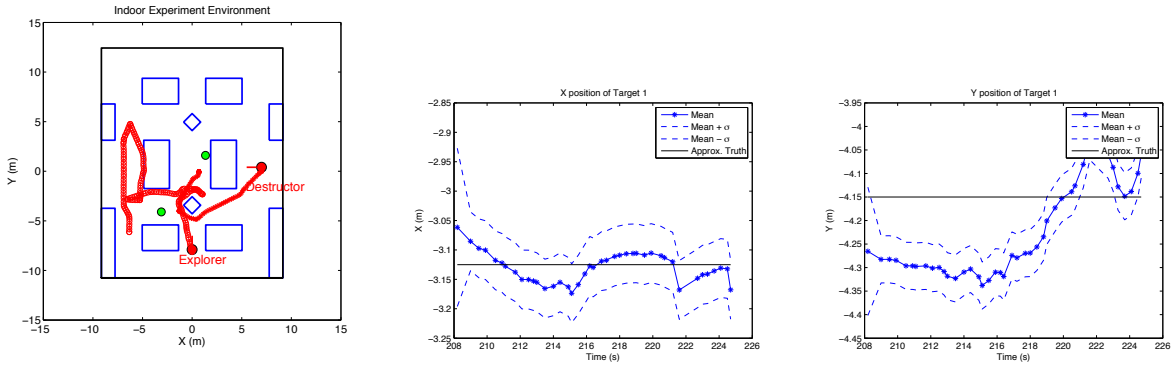
Figure 13. A picture(a) and a schematic(b) of the indoor environment with mapping results. Red circles denote two robots' initial positions and heading, and green circles represent targets. A path with circles and a star is drawn for each robot. The bottom figures (c - e) show an overhead-view map generated by two robots at three different moments.

(Figure 13(a)), causing the Hokuyo LRF to have no range returns. Figure 13(e) shows a complete full map after about 600 seconds, which suggests the generated map well represents the actual environment (Figure 13(b)) quite well.

8.1.2 Target tracking and neutralization

For the neutralization scenario, one explorer and one destructor were deployed in the indoor field and two static OOIs (SOOIs) were positioned, as shown in Figure 13(b). Both robots were controlled by a central planner, as described in Section 6. Once the explorer locates a potential target and the human operator confirms it, the destructor approaches and neutralizes the target with an onboard laser pointer.

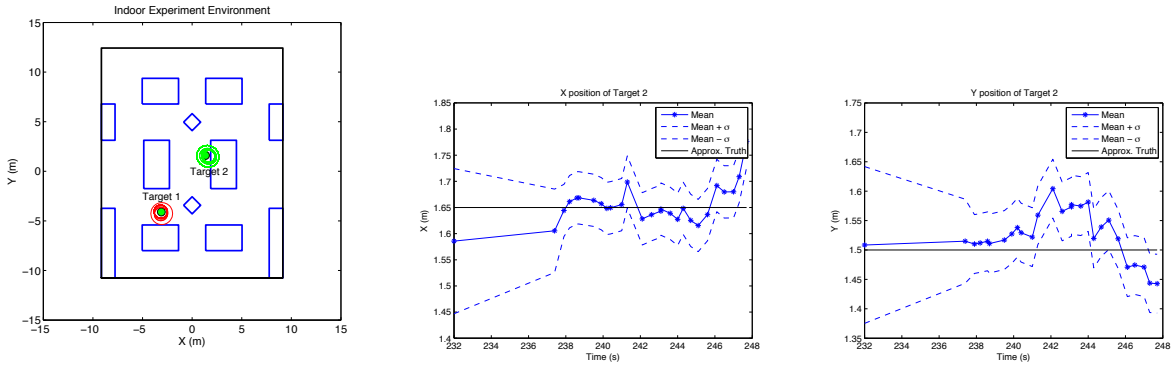
Figure 14 shows the neutralization process utilizing one explorer and one destructor. Figure 14(a) shows paths followed by each robot while green circles represent static OOIs. Once the explorer robot locates the target, it continues to collect data and localize. Overtime, the OOIs are more accurately localized. Figure 14(d) shows covariance circles for both targets shifting and converging over time. Figure 14(b - c, and e - f) show the estimates and 1σ bounds for the X, Y positions of both targets converging over time. Due to the covariance floor implemented to match with physical size of OOI, $\pm 1\sigma$ boundary remains constant after a short period of time. The mean values for target locations are not consistent over time, primarily because of non-point targets and laser point segmentations. However, the estimated locations of targets are within 20cm at all times which is reasonable for the targets of size diam. $= 50cm$. The localized targets' positions well match with the schematic presented.



(a) Schematic of the indoor experiment environment for neutralization process

(b) Mean and Covariance for target 1 in X-direction

(c) Mean and Covariance for target 1 in Y-direction



(d) Tracking showing covariance circles for two targets

(e) Mean and Covariance for target 2 in X-direction

(f) Mean and Covariance for target 2 in Y-direction

Figure 14. Neutralization environment schematic, path followed, and targets' mean and covariance plots

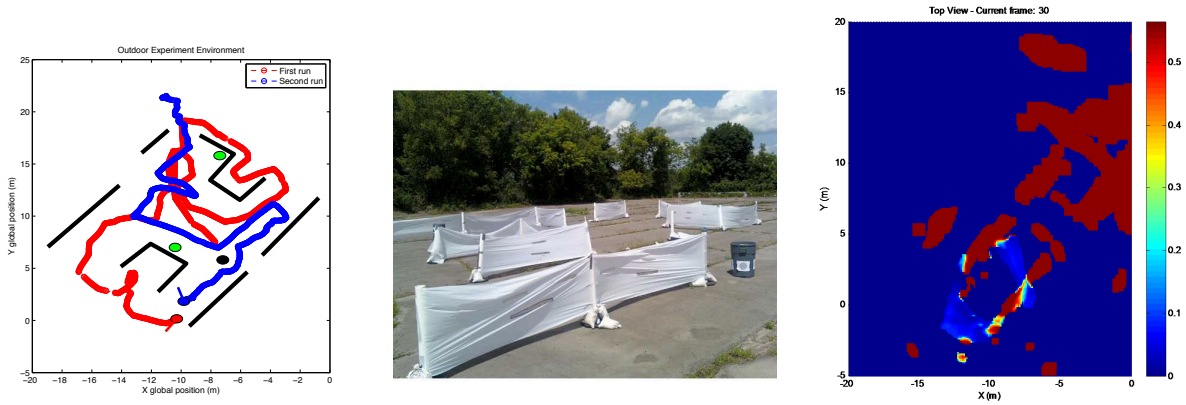
8.2 Outdoor Experiment

An outdoor experiment was conducted to demonstrate mapping and target tracking performance. Note that the only difference between indoor and outdoor experiments is the pose solution which uses GPS as an additional measurement. The outdoor environment is built using cloth wrapped around a number of plastic poles in order to construct temporary barriers as shown in Figure 15(b). The environment is set up as shown in Figure 15(a). One robot was deployed in the field, and two sets of data were collected. The two robots' data were broadcasted to the cooperative mapper and a joint map was generated.

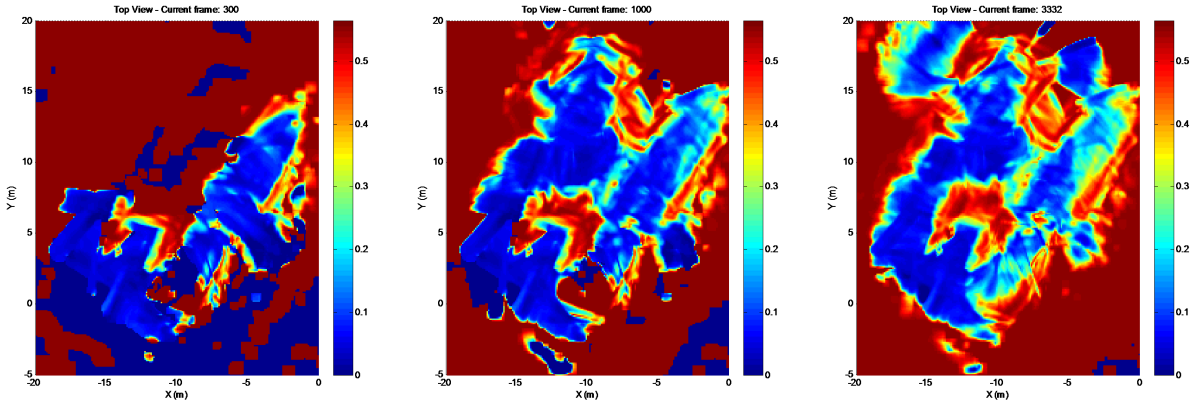
Figure 15(c - f) show the map generated from the two data sets fused together at four different instants. Figure 15(a) shows the robot trajectories of two data sets. Green circles represent static OOIs, and a black circle represents an object similar to the OOIs, but with a different color in order to test the robustness of the detection capability. Red and blue circles with lines represent initial positions and headings of the two different data sets. Figure 15(c - f) show the evolving map over time. Figure 15(f) shows the final map after the both data sets fused together. The outer red edges are trees, dirt piles, and grass hills, which are correctly classified as high terrain in the map. The final map shows good correspondence to the schematic shown in Figure 15(a).

Occasionally, GPS measurements cause the estimated position to shift approximately one meter, resulting in a smearing of the map compared to the indoor case. However, GPS measurements maintain global pose solution even after long periods of time; much longer than the indoor cases. As in the indoor test, the Hokuyo LRF had problems in measuring the range due to the low intensity return off the ground under certain lighting conditions. This causes some areas to be not updated.

Figure 16(a) shows covariance ellipse estimates of the two targets. Note that the converged position of Target 2 is slightly in error from the actual position due to GPS measurement update with a large relative error. Figure 16(b - e) show the mean position with $\pm 1\sigma$ boundaries for both targets. The dots on the mean represent the actual detections occurred. For example, during $t \approx [805, 850]$, there was no detection for target 1, and during that period robot's pose drifted a little, resulting the mean to shift. The overall means of the estimated target location are not consistent with the approximated truth position. This is mainly from the error in yaw angle of robot pose, since target location is calculated based on the current robot pose using SICK LRF. Compared to the indoor experiment, outdoor tracking performance shows more shifting in the target localization, typically correlated with large jumps in the GPS measurements. Also, due to the varying light condition during the outdoor experiment, more false target detections occurred for human operators to confirm as junk objects.



(a) Outdoor experiment environment (b) Outdoor experiment environment (c) Mapping at $t \approx 7.5s$

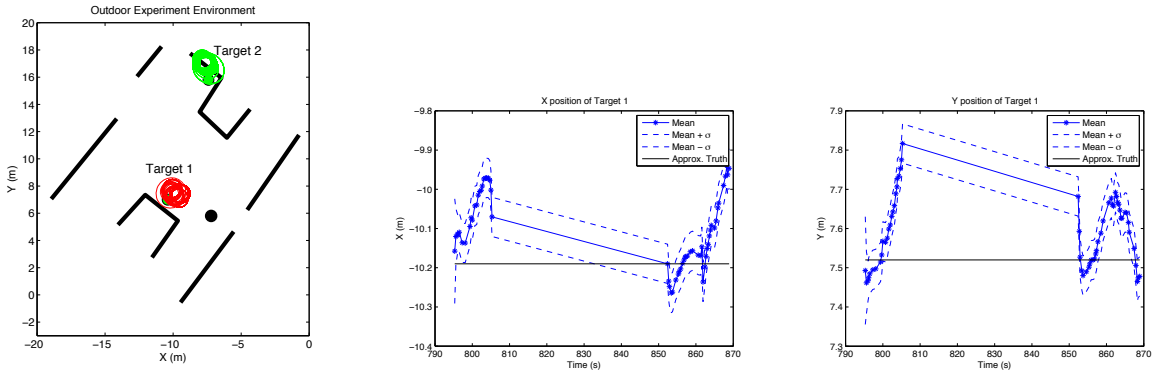


(d) Mapping at $t \approx 75s$ (e) Mapping at $t \approx 250s$ (f) Mapping at $t \approx 825s$

Figure 15. Outdoor experiment environment schematic and pictures including robots' trajectory. Mapping results are shown at four different times.

9. CONCLUSIONS

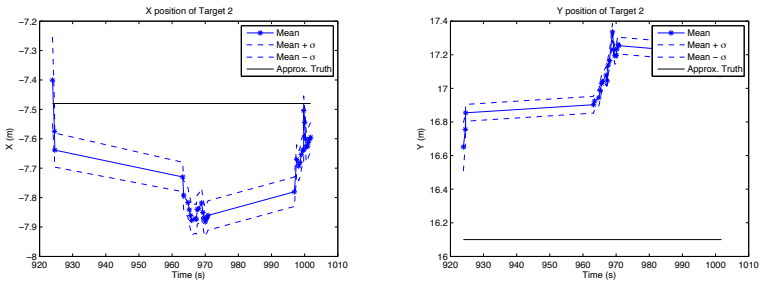
This paper presents an overview of a human-robotic system under development at Cornell which is capable of mapping an unknown environment, and discovering, tracking, and neutralizing several static and dynamic objects of interest. The paper details the system hardware, indoor and outdoor localization approach, cooperative mapping and tracking algorithms, multi-level planning approaches, and human-machine interface. Experimental results indicate good performance for cooperative mapping, tracking and indoor/outdoor operation. The primary challenges include long term localization while indoors for long term operations such as mapping; indoor to outdoor transitions of localization; cooperative tracking with low bandwidth communication; decentralized planning over uncertain graph/map/tasks; and efficient human-robotic interfacing.



(a) Outdoor tracking showing covariance circles for two targets

(b) Mean and covariance for target 1 in X-direction

(c) Mean and covariance for target 1 in Y-direction



(d) Mean and covariance for target 2 in X-direction

(e) Mean and covariance for target 2 in Y-direction

Figure 16. Target tracking performed in an outdoor experiment Mean and $\pm 1\sigma$ boundaries from two targets are shown

REFERENCES

1. N. Ahmed and M. Campbell, "Variational bayesian data fusion of multi-class discrete observations with applications to cooperative human-robot estimation," in *IEEE International Conference on Robotics and Automation*, 2010.
2. I. Miller and M. Campbell, "A mixture-model based algorithm for real-time terrain estimation," *Journal of Field Robotics* **23**, pp. 755–775, 2006.
3. J. Schoenberg and M. Campbell, "Distributed terrain estimation using a mixture-model based algorithm," in *FUSION*, 2009.
4. M. E. Campbell and W. W. Whitacre, "Cooperative tracking using vision measurement on seascan uavs," *IEEE Transactions on Control Systems Technology* **15**, pp. 613–626, 2007.
5. I. Miller and M. Campbell, "Rao-blackwellized particle filtering for mapping dynamic environments," in *International Conference on Robotics and Automation*, pp. 3862 – 3869, April 2007.
6. I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, F.-R. Kline, P. Moran, N. Zych, B. Schimpf, S. Lupashin, E. Garcia, M. Catlin, J. Kurdziel, and H. Fujishima, "Team Cornell's Skynet: Robust perception and planning in an urban environment," *Journal of Field Robotics* **25**, pp. 493–527, August 2008.
7. J. Ousingsawat and M. Campbell, "Optimal cooperative reconnaissance using multiple vehicles," *Journal of Guidance, Control, and Dynamics* **30**, pp. 122–132, Jan and Feb 2007.
8. H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal logic-based reactive mission and motion planning," *IEEE Transactions on Robotics* **25**(6), pp. 1370–1381, 2009.
9. H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured english to robot controllers," *Advanced Robotics Special Issue on Selected Papers from IROS 2007* **22**(12), pp. 1343–1359, 2008.
10. T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Conference on Decision and Control*, 2009.

11. D. Shah, J. Schneider, and M. Campbell, "A robust sketch interface for natural robot control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010.
12. B. Jones and L. Tong, "Information exchange in multi-rover slam," in *NSBE Aerospace Systems Conference*, 2010.
13. M. E. Campbell and M. Wheeler, "Vision-based geolocation tracking system for uninhabited aerial vehicles," *Journal of Guidance, Control, and Dynamics* **33**(2), pp. 521–532, 2010.
14. "Multi autonomous ground-robotic international challenge (magic 2010)." Website. `http://www.dsto.defence.gov.au/MAGIC2010/`.
15. K. Iagnemma, M. Buehler, and S. Singh, eds., *Special Issue on the 2007 DARPA Urban Challenge, Journal of Field Robotics*, vol. 25 (No. 8,9,10), pp. 423–860. Wiley Periodicals, 2008.
16. S. Brunke and M. E. Campbell, "Square root sigma point filter for real-time, nonlinear estimation," *AIAA Journal of Guidance, Control, and Dynamics* **27**, pp. 314–317, 2003.
17. Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley, 2001.
18. J. L. Crassidis and F. L. Markley, "Unsceted filtering for spacecraft attitude estimation," *AIAA Journal of Guidance, Control, and Dynamics* **26**, pp. 536–542, August 2003.
19. S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press, 2005.
20. J. Xavier, M. Pachcco, D. Castro, A. Ruano, and U. Nunes, "Fast line, arc/circle and leg detection from laser scan data in a player driver," in *IEEE International Conference on Robotics and Automation*, 2005.
21. P. E. Hart, N. J. Milsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics* **4**, pp. 100–107, July 1968 1968.
22. I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," *IEEE International Conference on Robotics and Automation* **2**, pp. 1572–1577, May 1998 1998.
23. D. An and H. Wang, "Vph: A new laser radar based obstacle avoidance method for intelligent mobile robots," *Fifth World Congress on Intelligent Control and Automation, WCICA 2004.* **5**, pp. 4681–4685, June 2004 2004.