Improving Occupancy Grid FastSLAM by Integrating Navigation Sensors

Christopher Weyers
Sensors Directorate
Air Force Research Laboratory
Wright-Patterson AFB, OH 45433

Gilbert Peterson
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

Abstract— **When an autonomous vehicle operates in an unknown environment, it must remember the locations of environmental objects and use those object to maintain an accurate location of itself. This vehicle is faced with Simultaneous Localization and Mapping (SLAM), a circularly defined robotics problem of map building with no prior knowledge. The SLAM problem is a difficult but critical component of autonomous vehicle exploration with applications to search and rescue missions. This paper presents the first SLAM solution combining stereo cameras, inertial measurements, and vehicle odometry into a Multiple Integrated Navigation Sensor (MINS) path. The FastSLAM algorithm, modified to make use of the MINS path, observes and maps the environment with a LIDAR unit. The MINS FastSLAM algorithm closes a 140 meter loop with a path error that remains within 1 meter of surveyed truth. This path reduces the error 79% from an odometry FastSLAM output and uses 30% of the particles.**

I. INTRODUCTION

One of the fundamental properties of an autonomous vehicle is its perception of the environment, even when initially unknown. This becomes increasingly difficult without the aid of external signals such as the Global Positioning System (GPS), which is often unavailable indoors. The challenge of navigating through an unknown environment while building a map is the Simultaneous Localization and Mapping (SLAM) problem. Its solutions include not only where a vehicle is, but also a representation of where it has traveled. The SLAM problem is fundamentally difficult, as these capabilities are codependent by themselves. However, an autonomous vehicle implementing a SLAM solution has incredible potential without relying on GPS or external communication.

Nearly all SLAM solutions construct the map using precise range measurements, but this precision is unhelpful if the pose estimate is uncertain. Vehicle odometry is often sufficient to obtain an accurate estimate over a short path with few turns. However, this becomes exceedingly difficult traveling a farther distance or exploring a larger environment [2]. Odometry error is cumulative and often grows quickly. An algorithm must include more variance in its estimate to account for this, increasing computation and decreasing solution certainty. This research seeks to produce a more accurate vehicle pose estimate allowing for less variance and a more consistent solution.

Previous research has developed a navigation system using an Extended Kalman Filter (EKF) to combine cameras and inertial measurements [3]. It consists of stereo cameras tightly coupled with an Inertial Measurement Unit (IMU) to produce an accurate position for different platforms in real-time, but is not designed for mapping [4].

This approach integrates pose estimates from stereo camera egomotion, IMU measurement integration, and vehicle odometry in a linear Kalman filter (KF). This creates a Multiple Integrated Navigation Sensor (MINS) path. A particle filter then combines the MINS path with LIDAR ranges and creates an occupancy grid map in a FastSLAM solution [5]. Because the MINS system calculates its position outside the particle filter, the SLAM implementation does not increase in complexity.

In real world tests, a vehicle operating all necessary sensors travels a 140 m loop around indoor hallways and collects data, which the MINS system and FastSLAM implementation process. With more sensors to estimate position, this MINS and FastSLAM strategy produces a more accurate path with 79% less error than the FastSLAM implementation using only vehicle odometry.

The following section gives a background on related SLAM strategies and stereo image navigation. The next section discusses the methods used in incorporating IMU data and image features with odometry to achieve control input u_t, and specifics of the SLAM implementation. The final sections describe the testing procedure used and results obtained, as well as propose further topics to explore.

II. BACKGROUND AND RELATED WORK

This section reviews existing SLAM and navigation techniques utilized in and related to this work.

A. SLAM Solutions

The SLAM problem is often represented as a Dynamic Bayes Network (DBN) in Fig. 1. Online solutions make the Markov assumption, saving all useful information in the current state [6]. In the DBN structure, observed variables (controls u_t and measurements z_t) are inputs and hidden variables (poses $\mathbf{s}_t$ and map Θ) form the SLAM solution.

Since algorithms make localization the first aspect of SLAM, the most important aspect is to accurately represent the vehicle position and orientation in 2D with coordinates x, y, and heading θ. Added geometrically, these are the elements of vehicle pose $\mathbf{s}$, depicted in (1) as a vector.

$$\mathbf{s} = \begin{bmatrix} x & y & \theta \end{bmatrix}^{\mathrm{T}} \tag{1}$$

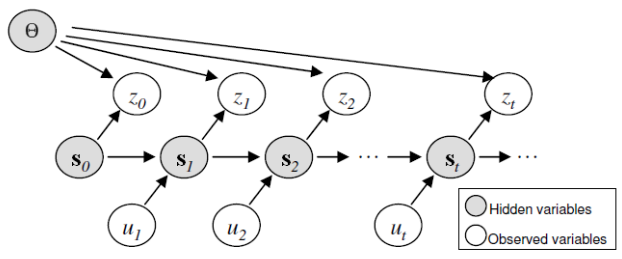

Fig. 1. SLAM as a DBN. Controls u_t and measurements z_t are observed at each time t, as vehicle poses $\mathbf{s}_t$ and a single map Θ are hidden [7].

To solve the probabilistic DBN, vehicle localization algorithms redefine the SLAM solution to be the probability that a vehicle occupies any given location. To localize in an environment, the Bayesian filtering problem executes a motion model, then represents an estimate of the posterior probability in (2) using the Bayes theorem [8].

$$p(\mathbf{s}_t|z_{0:t}) = \frac{p(z_t|\mathbf{s}_t)\ p(\mathbf{s}_t|z_{0:t-1})}{p(\mathbf{s}_t|z_{0:t-1})} \tag{2}$$

SLAM solutions iterate by first estimating a pose within an existing map, then updating the map to incorporate new measurements [9]. The SLAM posterior is similar to the localization posterior with more variables. SLAM requires the current pose $\mathbf{s}_t$, all measurements $z_{0:t}$ and controls $u_{1:t}$ to maintain the map Θ. The posterior is calculated as an integral in (3), where η is a normalization constant.

$$\begin{aligned} p(\mathbf{s}_t, \Theta|z_{0:t}, u_{1:t}) &= \eta\ p(z_t|\mathbf{s}_t, \Theta) \cdot \\ &\int p(\mathbf{s}_t|\mathbf{s}_{t-1}, u_t)\ p(\mathbf{s}_{t-1}, \Theta|z_{0:t-1}, u_{1:t-1})\ d\mathbf{s}_{t-1} \end{aligned} \tag{3}$$

Due to its occupancy grid map storage, this research does not include a term for data association. Rather, it considers data association part of the measurement model. Because evaluating this integral is computationally prohibitive, SLAM solutions use statistical estimating techniques to approximate the Bayesian filter problem [10].

A common approach to approximating this posterior is to use a particle filter to maintain the error distribution through M samples, each one storing a pose estimate $\mathbf{s}^{[m]}$ and map $\Theta^{[m]}$ [10]. FastSLAM operates a Rao-Blackwellized Particle Filter (RBPF), which samples from a Gaussian proposal distribution [11]. It then calculates an importance weight $w_t^{[m]}$ for each particle in (4), as the ratio of distributions at each sample location.

$$w_t^{[m]} = \frac{target\ distribution}{proposal\ distribution} = \frac{p(\mathbf{s}_t^{[m]}|z_t, u_t)}{p(\mathbf{s}_t^{[m]}|z_{t-1}, u_t)} \tag{4}$$

With the weighting determined, FastSLAM uses a Sampling Importance Resampling (SIR) algorithm to select a new set of samples based on each $w_t^{[m]}$. This replaces low weighted particles with higher weighted ones much like the operation of a Genetic Algorithm.

B. SLAM Improvements

To reduce the chances of replacing good particles, a SLAM solution can implement a selective resampling process to only resample when necessary [12]. This reduces the risk of removing accurate particles by only resampling the particle set once the weighting indicates a need, instead of every time. This strategy is also implemented in this research, as both the RBPF and metric map techniques are similar.

Another FastSLAM enhancement uses scan matching, which leverages the accuracy of LIDAR units; this reduces the number of particles required, cutting memory requirements and computation time [12]. Using the principle of scan matching, the most recent measurement is considered in addition to vehicle movement, resulting in a particle distribution with less variance. The RBPF then preceeds with weighting the particles by comparing z_t to each $\Theta^{[m]}$.

A common map storing strategy is metric representation, where each stores objects by their absolute position. Storing a global metric map such as this carries a significant memory requirement which can affect computation more than the algorithm complexity [5]. Other work maintains several local metric maps roughly the size of the sensor range [7]. This work also maintains an overall topological map (storing locations relative to each other), and places the local metric maps within the global topological one. This results in less of an overall storage requirement and helps to keep local areas correctly placed relative to each other.

C. Machine Vision SLAM

Many SLAM implementations have presented results using odometry and LIDAR, but others present the use of other input sensors. Extracting SIFT features from cameras has been used as an alternative SLAM input, removing the requirement for other sensors [13]. Later work used a RBPF implementation with features as input and resulted in a more accurate SLAM map than using pose estimates from odometry measurements [14].

Cost is a significant factor in these situations, as cameras are smaller, lighter, cheaper, and require less power than LIDAR units. Additionally, cameras are passive sensors, as they do not project energy into the environment. This is safer and less detectable than using active sensors like ranging devices.

Other work uses cameras alongside other sensors, as used extensively for rover navigation when paired with odometry [15]. The process of using sequential images for navigation is called egomotion (also visual odometry or optical flow). Egomotion relies on epipolar geometry to calculate focal points from stereo cameras [16]. This mathematical solution provides the basis for the feature processing used in this research.

The algorithms match image features over time, then determine their position using stereopsis, the ability to approximate distance. Egomotion translates feature movement into vehicle motion, and has been used for SLAM applications with different map representation on simulated data [1].

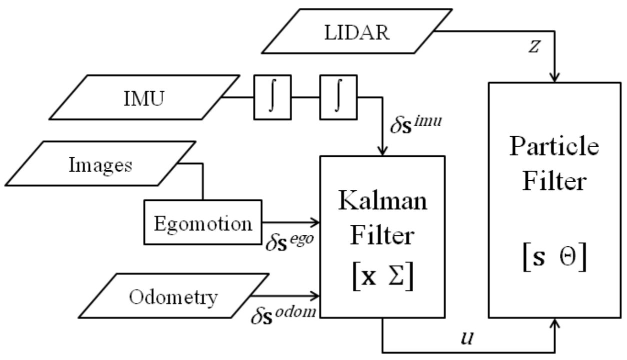

Fig. 2. The KF takes input from IMU integration, stereo image egomotion, and odometry, then provides u to a RBPF with LIDAR ranges z.

III. METHODOLOGY

Where previous research used separate sensors to create separate paths, this work combines stereo image egomotion, integrated IMU measurements, and odometry using a linear KF to generate a single MINS path. A FastSLAM implementation uses this MINS path as u_t and LIDAR ranges as z_t. It carries out the necessary steps to construct occupancy grid maps from these inputs. Fig. 2 shows an overall system diagram of MINS and FastSLAM. It has four inputs from the four sensors, and maintains pose $\mathbf{s}$ and map Θ as output.

A. Stereo Egomotion

To take advantage of stereopsis, MINS begins by matching features between the left and right camera images using SIFT [17]. MINS calculates a metric similar to Euclidean distance from of each pair of left and right camera descriptors $\mathbf{d}$ in (5) to find the distance between each descriptor pair.

$$\arccos(\mathbf{d}^{left} \cdot \mathbf{d}^{right}) \tag{5}$$

If the closest match is less than 0.6 times the second closest, MINS matches the feature and saves its pixel coordinates. To reduce the number of false matches, MINS discards feature matches in the other image that measure greater than 1.5° from horizontal. Before the coordinates can be accurately used, MINS removes lens distortion effects through the CalTech distortion model [3].

MINS then uses epipolar geometry to measure the physical location of each feature taking advantage of stereopsis [16]. It uses Direction Cosine Matricies (DCMs) to convert pixels to physical locations in the correct reference frame relative to the vehicle.

To implement egomotion, MINS assumes that the only thing moving in the environment is the vehicle. This makes it possible to extract a vehicle pose change, because the algorithm interprets feature motion as a result of vehicle motion. MINS removes outlying feature matches by eliminating those that travel more than 0.75 standard deviations from the mean distance traveled.

MINS then uses a polar representation to measure movement by subtracting the current locations from the previous locations in polar form to obtain position differences. This is opposite typical motion, as the algorithm is translating movement in one direction as vehicle movement in the other. Finally, MINS measures egomotion pose difference $\delta\mathbf{s}_t^{ego}$ in (6) by converting the mean of the differences $\bar{\theta}$ and $\bar{r}$ back to rectangular form.

$$\delta\mathbf{s}_t^{ego} = \begin{bmatrix}\bar{r}\cos(\bar{\theta}) & \bar{r}\sin(\bar{\theta}) & \bar{\theta}\end{bmatrix}^{\mathrm{T}} \tag{6}$$

B. Inertial Integration

Obtaining $\delta\mathbf{s}_t^{ego}$ from the images is merely one of the three measured paths. Vehicle odometry measures a second path $\delta\mathbf{s}_t^{odom}$ that MINS does not need to process. MINS computes an IMU pose difference $\delta\mathbf{s}_t^{imu}$ for the third path, as the IMU measures linear accelerations and rotational velocities. These measurements, shown in (7) must be integrated twice and once, respectively [18].

$$\begin{bmatrix}\ddot{x}_t & \ddot{y}_t & \dot{\theta}_t\end{bmatrix}^{\mathrm{T}} \tag{7}$$

Due to specific IMU hardware, MINS does not compensate for coning and sculling effects. However, to limit the effect of drifting, MINS tracks and removes a bias in $\dot{\theta}_t$ when the vehicle is stationary [3]. MINS first integrates $\ddot{x}_t$ and $\ddot{y}_t$ to current velocity $\mathbf{v}_t$ in (8). To mitigate growing integrated errors, MINS limits $|\mathbf{v}_t|$ to vehicle maximum 1 m/s.

$$\mathbf{v}_t = \begin{bmatrix}\dot{x}_{t-1} + (dt)\ddot{x}_t \\ \dot{y}_{t-1} + (dt)\ddot{y}_t \\ \dot{\theta}_t\end{bmatrix} \tag{8}$$

MINS performs numeric integration on the measured values in (7) to determine current IMU pose difference $\delta\mathbf{s}_t^{imu}$ in (9), where dt is the time interval since the previous measurement [18].

$$\delta\mathbf{s}_t^{imu} = \frac{dt}{2}\Big(\mathbf{v}_{t-1} + \mathbf{v}_t\Big) \tag{9}$$

C. MINS Path

MINS combines $\delta\mathbf{s}_t^{imu}$ from the IMU, $\delta\mathbf{s}_t^{ego}$ from the cameras, and $\delta\mathbf{s}_t^{odom}$ from odometry to produce the combined pose difference $\delta\mathbf{s}_t$. It does this by implementing a linear KF, advantageous as it handles asynchronous updates and extends easily to include different sensors and uncertainties.

KFs are often described as a series of predictions and observations that maintain state $\mathbf{x}_t$ and covariance $\mathbf{\Sigma}_t$ [19]. MINS provides each $\delta\mathbf{s}_t^{imu}$ as a prediction with $\delta\mathbf{s}_t^{ego}$ and $\delta\mathbf{s}_t^{odom}$ as separate observations. MINS sets the associated variance of each from hardware specifications and tested path accuracy. MINS calculates the KF functions through the Bayes++ filtering library [20]. The result gives MINS a pose difference $\delta\mathbf{s}_t$ from KF state $\mathbf{x}_t$.

D. FastSLAM Implementation

MINS provides $\delta\mathbf{s}_t$ as control input u_t to FastSLAM as a 2D path. MINS also sends FastSLAM the KF covariance $\mathbf{\Sigma}_t$ shown in (10) to represent its certainty.

$$\mathbf{\Sigma}_t = \begin{bmatrix}\sigma_{xx} & \sigma_{xy} & 0 \\ \sigma_{xy} & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{\theta\theta}\end{bmatrix} \tag{10}$$

1) Motion Model: The first FastSLAM step uses a motion model to propagate each particle to an updated pose using u_t. Four parameters determine the amount of the Gaussian noise applied. They represent the motion uncertainty and allow the RBPF to model specific errors in u_t [2].

Uncertainty in translation by translation α_3 and rotation by rotation α_1 correspond respectively with the upper and lower blocks of $\mathbf{\Sigma}_t$. Therefore, FastSLAM obtains its motion model in (11) by modifying these parameters without changing its uncertainty in rotation by translation α_2 or uncertainty in translation by rotation α_4 from configured constants. In this way, the MINS covariance affects motion uncertainty in the FastSLAM implementation.

$$\begin{bmatrix} (\sigma_{\theta\theta})\alpha_1 \\ \alpha_2 \\ (\sigma_{xx} + \sigma_{xy} + \sigma_{yy})\alpha_3 \\ \alpha_4 \end{bmatrix} \tag{11}$$

2) Measurement Model: The measurement model satisfies the second FastSLAM step to apply z_t to each $\Theta^{[m]}$. Necessarily different for each map representation and sensor, the measurement model compares z_t to each $\Theta^{[m]}$, accumulating line errors e_{line} into scan errors e_{scan} that measure inconsistencies. This research uses a LIDAR unit and maintains a metric occupancy grid, storing probabilities that locations are open or occupied [21].

Algorithm 1 gives an overview of this process, making calculations similar to the map building process. Each LIDAR scan has a valid range r_{min} to r_{max}, and the occupancy grid has 0.1 m resolution and a 256 value range. Additionally, the model decays each previous map error $e_{t-1}^{[m]}$ by discount factor 0.99 to weigh recent observations higher in determining current map error $e_t^{[m]}$ [6].

3) Particle Weighting: After executing the measurement model, FastSLAM assigns a weight $w_t^{[m]}$ to each particle from its map error $e_t^{[m]}$. It first subtracts the minimum error value from each $e_t^{[m]}$, then imposes a lower limit of 2 to prevent a spike in the particle distribution. FastSLAM calculates weight $w_t^{[m]}$ in (12) by taking the reciprocal of its map error $e_t^{[m]}$ and normalizing the result.

$$w_t^{[m]} = \frac{1}{e_t^{[m]}} \left(\sum_{m=1}^{M} \frac{1}{e_t^{[m]}} \right)^{-1} \tag{12}$$

4) Resampling: Before resampling, FastSLAM performs a test to determine if resampling is necessary [12]. This reduces both computation and the chances of a good particle being replaced by resampling too often. The test calculates the number of effective particles by determining a sum of the squared weights. FastSLAM only resamples if this number is less than $M/2$, half the number of particles.

IV. TESTING AND RESULTS

The test data is collected from the Pioneer P2-AT8 vehicle shown in Fig. 3. The P2-AT8 provides internal odometry on skid steering wheels, and a SICK LMS 200 LIDAR unit. The vehicle also carries two PixeLINK PL-A741 cameras with a 90° field of view and 1280 × 960 resolution, and a MicroRobotics MIDG II consumer grade IMU with 50 Hz sampling rate, both connected to an external PC. This PC connects to the internal one via Ethernet cable to record collected data.

A manually driven vehicle test run provides a reusable data set. The vehicle begins in the northeast corner before traveling left around the 30×40 meter rectangular loop. After returning to its starting location, the vehicle makes a right turn into a room and stops.

The implementation first computes the feature extraction, stereopsis, and egomotion using a SIFT feature executable

Algorithm 1 LIDAR Grid Measurement Model

```
for all particles $[m]$ do
  $e_{scan} = 0$
  for all scans with range $r_s$ at angle $\theta_s$ do
    if $r_{min} < r_s < r_{max}$ then
      $e_{line} = 0$
      $x_t^q = r_s \cos(\theta_s)$
      $y_t^q = r_s \sin(\theta_s)$
      $\mathbf{q}_t = \mathbf{s}_t^{[m]} + [x_t^q \; y_t^q \; 0]^{\mathrm{T}}$
      $\rho = \lceil r_s/0.1 \rceil$
      $x^d = (x^q - x_t^{[m]})/\rho$
      $y^d = (y^q - y_t^{[m]})/\rho$
      for $i = 0$ to $\rho + 1$ do
        $x^p = x_t^{[m]} + (i)x^d$
        $y^p = y_t^{[m]} + (i)y^d$
        if $i \leq \rho$ and $\Theta[x^p][y^p]^{[m]} > 0$ then
          $e_{line}$ += $\Theta[x^p][y^p]^{[m]}(\rho + 1 - i)(0.1/256)$
          break
        else if $i > \rho$ and $\Theta[x^p][y^p]^{[m]} < 0$ then
          $e_{line}$ −= $\Theta[x^p][y^p]^{[m]}(0.1/256)$
        end if
      end for
      $e_{scan} = e_{scan} + e_{line}$
    end if
  end for
  $e_t^{[m]} = 0.99e_{t-1}^{[m]} + e_{scan}$
end for
```

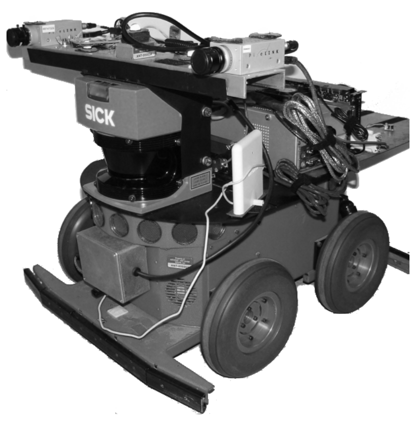
Fig. 3. The P2-AT8 vehicle with cameras and IMU above the LIDAR.

TABLE I

Path Error (meters)

	Odometry	MINS	Odometry FastSLAM	MINS FastSLAM
Fig.	5 left	5 right	6 top	6 bottom
Start	0.0	0.0	0.0	0.0
NE	0.345	0.261	0.359	0.269
NW	6.587	0.482	0.128	0.584
SW	40.751	3.716	0.898	0.742
SE	62.627	5.413	2.452	0.863
NE	81.935	5.311	3.709	0.096
Finish	91.203	6.185	6.000	0.298
mean	40.499	3.052	1.935	0.408

and MATLAB® script files. This portion requires hours of computation if calculated directly; improvements have been discussed in previous research [4]. The main implementation includes the remainder of MINS and all of FastSLAM as an application built on the Bayes++ filtering library. The remaining discussion concerns only this main implementation and not the image calculations.

As the true vehicle path was unknown at all times, path errors can be approximated using a set of surveyed points in the environment corresponding to start, finish, and each of the corner locations. To compare the error of each path, this research measures the distance between the current pose and the survey point location at that time. Table I displays each error for odometry, the MINS system, and FastSLAM using odometry and FastSLAM using MINS for control input u_t.

After taking the first corner, vehicle odometry drifts away and compounds error much faster than the MINS system, which is only six meters away upon returning to the starting location. The FastSLAM implementation greatly improves on each of these paths by incorporating LIDAR scans. When using the more accurate MINS path as input, FastSLAM remains within a meter at all points.

Table I does not factor errors in heading or inaccuracies at other points along the path, so Fig. 4 displays vehicle odometry and the MINS path alongside approximate truth from the surveyed points.

MINS is suitable for navigation requirements in local environments without using GPS. It displays accurate corners with a smooth path and straight hallways. MINS does not suffer problems from loss of image features, as consistent odometry provides an additional KF input.

The goal of SLAM is more about building maps than producing a path, so applying LIDAR scans produces a better visual result. The occupancy grids indicate open space as white and occupied space as black; unobserved areas are grey. Fig. 5 displays maps from the odometry and MINS paths, revealing more details about their accuracy.

The odometry map (Fig. 5 left) contrasts the severity of its drift with the accuracy of its measured distances. MINS (Fig. 5 right) combines the accuracy of each sensor to produce a path close to closing the loop. These paths do not use the LIDAR scans to improve pose information.

Since the FastSLAM implementation seeks to improve on these paths, Fig. 6 displays the maps generated by the best FastSLAM particle. Odometry FastSLAM with 100 particles (Fig. 6 top) improves the path remarkably, but is unable to remove all drift and does not close the loop; MINS FastSLAM (Fig. 6 bottom) needs only 30 particles to produce its map.

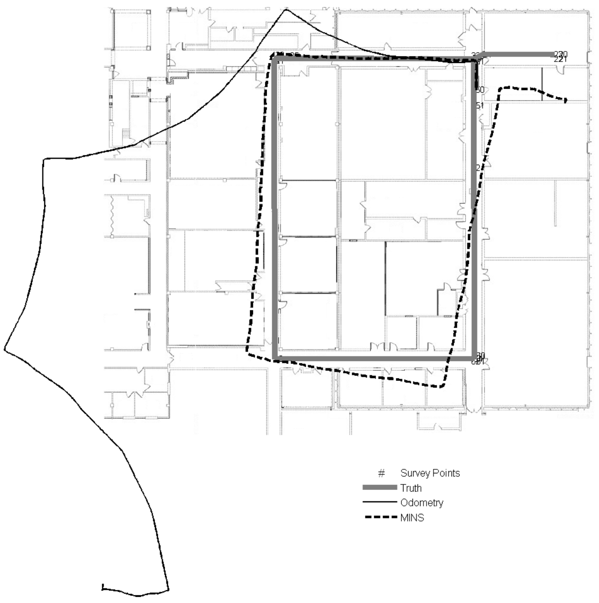

Fig. 4. Approximate truth, odometry, and MINS over a building floor plan.

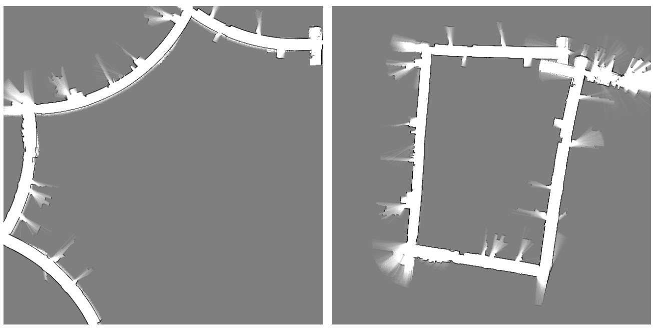

Fig. 5. Maps produced from LIDAR scans using odometry (left) and using the MINS path (right)

Because the implementation calculates the MINS path outside the FastSLAM algorithm, its computation time is also reduced linearly with number of particles to 30%. Furthermore, the main MINS implementation (consisting of IMU, path, and KF calculations) adds no measurable computation time to the FastSLAM algorithm. This keeps overall computation linear in both analysis and in results.

The end result of this research is the MINS system connected to the FastSLAM implementation. Improving on the MINS path and odometry FastSLAM results, using MINS with FastSLAM successfully closes the 140 m loop. It displays straight hallways with minimal inaccuracies from LIDAR scans, which are also present in the other maps.

V. CONCLUSIONS AND FUTURE WORK

This research presents a SLAM solution for exploring indoor 2D environments. It integrates multiple navigation

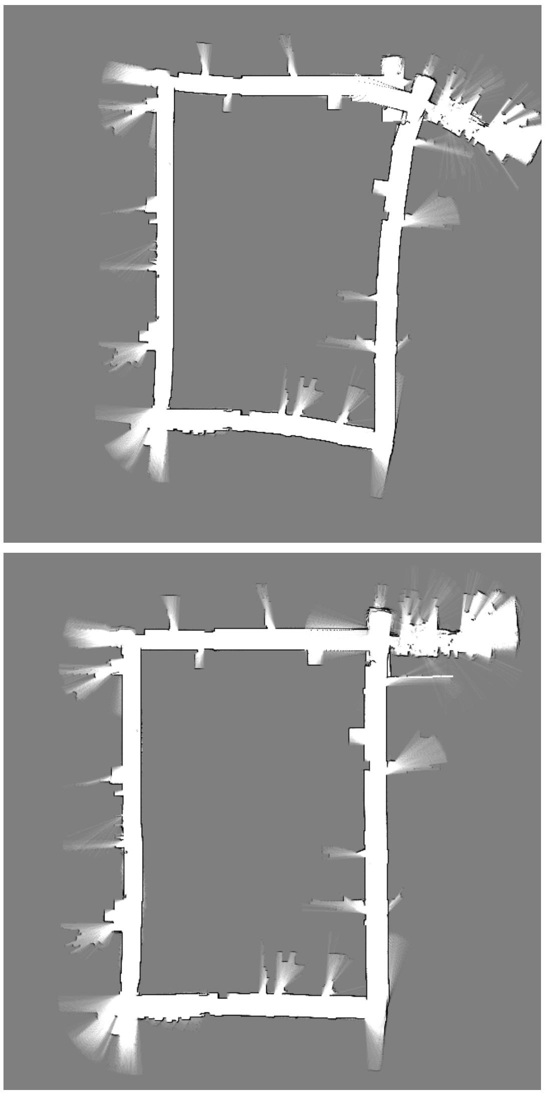

Fig. 6. The best particle maps from 100 particle odometry FastSLAM (top) and 30 particle MINS FastSLAM (bottom).

sensors to provide an improved solution for FastSLAM. By combining sensors in MINS and using its path in a FastSLAM implementation using LIDAR and occupancy grid maps, this research achieved a more accurate path than odometry in real-world tests around a large loop.

The most time intensive aspect of the implementation involves processing the stereo image features for egomotion, which is also the least consistent path. In comparison, the odometry and IMU paths do not require advanced calculation. To provide feasible real-time navigation, simple sensors like these should be used with accurate ranges. LIDAR scan matching appears to be the most likely strategy to improve this research, improving the motion distribution and particle accuracy [12]. Alternatively, MINS provides an opportunity to apply new sensors and combinations to SLAM solutions, but it may be more complicated than necessary. Implementing a simple KF, it can likely be replaced with a weighted average system without losing much effectiveness, rather than incorporating SLAM into a more complex KF.

ACKNOWLEDGMENTS

The authors would like to thank the Advanced Navigation Technology Center staff for all their supporting efforts.

This work was supported in part through AFRL/RYMN Lab Task 06SN02COR from the Air Force Office of Scientific Research. The views expressed in this article are those of the authors and do not reflect the official policy or position of the United State Air Force, Department of Defense, or the U.S. Government.

REFERENCES

[1] G. Bleser and G. Hendeby, "Using optical flow as lightweight slam alternative," in *IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 175–176.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[3] M. Veth, "Fusion of image and inertial sensors for navigation," Ph.D. dissertation, Air Force Institute of Technology, 2006.

[4] J. Fletcher, "Real-time gps-alternative navigation using commodity hardware," Master's thesis, Air Force Institute of Technology, 2007.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *AAAI National Conference on Artificial Intelligence*, 2002, pp. 593–598.

[6] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[7] J.-L. Blanco, J.-A. Fernández-Madrigal, and J. Gonzalez, "Towards a unified bayesian approach to hybrid metric-topological slam," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 259–270, 2008.

[8] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1322–1328.

[9] M. G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[10] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *International Joint Conference on Artificial Intelligence*, 2003, pp. 1151–1156.

[11] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 176–183.

[12] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

[13] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *International Journal of Robotics Research*, vol. 21, no. 8, pp. 735–758, 2002.

[14] R. Sim, P. Elinas, and J. Little, "A study of the rao-blackwellised particle filter for efficient and accurate vision-based slam," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 303–318, 2007.

[15] C. Olson, L. Matthies, M. Schoppers, and M. Maimone, "Rover navigation using stereo ego-motion," *Robotics and Autonomous Systems*, vol. 43, no. 4, pp. 215–229, 2003.

[16] X. Armangué, H. Araújo, and J. Salvi, "A review on egomotion by means of differential epipolar geometry applied to the movement of a mobile robot," *Pattern Recognition*, vol. 36, no. 12, pp. 2927–2944, 2003.

[17] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 2, no. 60, pp. 91–110, 2004.

[18] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*, 2nd ed. The Institution of Electrical Engineers, 2004.

[19] P. Maybeck, *Stochastic Models, Estimation, and Control*. Academic Press, 1979, vol. 1 and 2.

[20] M. Stevens, "Bayes++ bayesian filtering library," Australian Centre for Field Robotics, 2005.

[21] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.