Technical Report: GIT-GVU-05-13

An Empirical Evaluation of Context-Sensitive Pose Estimators in an Urban Outdoor Environment[*]

Yoichiro Endo Patrick D. Ulam Ronald C. Arkin Tucker R. Balch Matthew D. Powers

Georgia Tech Mobile Robot Laboratory
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA

{endo, pulam, arkin, tucker, mpowers}@cc.gatech.edu

Abstract – **When a mobile robot is executing a navigational task in an urban outdoor environment, accurate localization information is often essential. The difficulty of this task is compounded by sensor drop-out and the presence of non-linear error sources over the span of the mission. We have observed that certain motions of the robot and environmental conditions affect pose sensors in different ways. In this paper, we propose a computational method for localization that systematically integrates and evaluates contextual information that affects the quality of sensors, and utilize the information in order to improve the output of sensor fusion. Our method was evaluated in comparison with conventional probabilistic localization methods (namely, the extended Kalman filter and Monte Carlo localization) in a set of outdoor experiments. The results of the experiment are also reported in this paper.**

Index Terms – Sensor Fusion, Context-Sensitive Perception, Localization, Extended Kalman Filter, Particle Filter

I. Introduction

Execution of an autonomous mobile robot mission in an urban outdoor environment, such as a reconnaissance mission, often requires a waypoint-following capability (e.g. [1, 2]). Since waypoints are generally specified in a world coordinate system, localizing the robot relative to the world coordinate system is vital in such applications. Furthermore, depending on the urgency of the mission, the localization may have to be accomplished as rapidly as possible. Today, in a typical well-structured indoor environment, laser-based and vision-based SLAM approaches have proven to be useful for localization [3-6]. However, objects in an outdoor environment are less structured, and the lasers or vision systems are vulnerable to severe and dynamic variations in lighting. For outdoor applications, the global positioning system (GPS), compass, gyro-based inertial measurement unit (IMU), and shaft-encoder are standard sensors used to measure where the robot is and what direction it is heading. On the other hand, these sensors are still not perfect, each having its own peculiar strengths and weaknesses. For example in the case of a differential GPS, if satellite signals and/or differential signals are disrupted by the surrounding environment, the positional accuracy provided by the GPS severely degrades. However, our supposition here is that the robot should be able to dynamically assess the qualities of the sensors by monitoring parameters that are known to affect them (e.g., RT-20 value [7] of the GPS, speed of the robot, etc.). We refer to such parameters as contextual information. The contextual information is translated into some quantities that convey the qualities of the sensors by applying some domain knowledge (rules).

The objective of this paper is twofold: (1) To propose a computational scheme that can systematically incorporate contextual information and domain knowledge in order to gauge the robot's best current pose when measurements from multiple sensors are available; and (2) to empirically evaluate whether incorporation of such information/knowledge is indeed useful or not.

It should be noted that, in this paper, the term "pose" we are referring to is a six-dimensional vector (X), whose components include x, y, z, ϕ, θ, and ψ (Equation 1):

$$X = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^{\mathrm{T}} \qquad (1)$$

The values of x, y, and z describe the location of a body in the world coordinate system, and their units are in meters. The x-axis points to East, and the y-axis points to North. The value z describes an altitude. On the other hand, rotations ϕ, θ, and ψ are yaw, pitch, and roll of the body, respectively, in degrees.

The process of collecting data from multiple sensors in order to produce an integrated perceptual output is known as sensor fusion. Murphy [8], for example, proposed a perceptual architecture (SFX) for action-oriented sensor fusion. In the SFX, one of three fusion states is selected based on behavioral needs. In the first fusion state, contributions from all sensors are integrated in order to produce a single perceptual output since no conflict among different sensors is expected. In the second fusion state, a conflict (discordance) among the sensors is expected, and calibration of weak (unreliable) sensors is performed before generating the overall output. The third type is a greedy method. The discordance among the sensors is resolved by suppressing the contributions from all sensors except the most reliable one. In our three proposed fusing methods (Section II.B): two of them (the EKF and Particle Filter) relate to Murphy's first fusion state (cooperative); and one of them (the Maximum Confidence) relates to the third

[*] This research is funded under DARPA/DOI contract #NBCH1020012 as part of the MARS 2020 program.

fusion state (suppressive). Furthermore, our cooperative fusing methods incorporate probabilities, as the employment of probabilities in the context of sensor fusion for mobile robots has proven useful for more than a decade [9]. In fact, Thrun [10] reports in his survey paper that all of the successful approaches to robotics localization today employ probabilistic techniques.

The organization of this paper is as follows. An overview of the context-sensitive pose computation is represented in Section II. The implementation details of the computational scheme are described in Section III. Empirical evaluation via outdoor experimentation is presented in Section IV. Finally, conclusions and future work are discussed in Section V.

II. Context-Sensitive Pose Computation

A. Computational Steps

Suppose that a mobile robot is equipped with multiple pose sensors (e.g., GPS, compass, IMU, shaft-encoder, etc.). Each sensor provides pose information that may be computed in different coordinate systems and generally updated at a different time rate from others. Given these sensors, the objective here is to compute the best pose at any given time in the common world coordinate system.

The first step in the process is to transform the original sensor readings measured by each sensor into the common world coordinate system. Since different sensors operate with different coordinate systems, the transformation function is unique to the sensor type. It should be noted that some sensors, such as shaft-encoders and IMU, compute their poses based on dead-reckoning. While such sensors are prone to cumulative errors, their incremental readings from previous measurement are reasonably accurate. The other types of sensors, such as compass and GPS, compute poses with the absolute location and orientation. Equation 2 expresses this operation mathematically:

$$X_{st} = \begin{cases} f_{\mathrm{T}s}(X_{st-1}, \Delta r_{st}) & \text{if } s = \text{dead-reckoning-based} \\ f_{\mathrm{T}s}(r_{st}) & \text{otherwise} \end{cases} \quad (2)$$

where X_s is the converted pose for sensor s, t is the current instant, $f_{\mathrm{T}s}$ is the transformation function for the sensor, Δr_s is the increment from the previous measurement, and r_s is the measurement itself.

Once a pose from each sensor is computed, the next step is to compute estimated accuracy of the pose; in this paper, we refer to the estimated accuracy as *grade*, denoted with letter G. More specifically, suppose we define vectors Φ and Σ as shown in Equations 3 and 4, respectively:

$$\Phi = [\alpha_x \quad \alpha_y \quad \alpha_z \quad \alpha_\phi \quad \alpha_\theta \quad \alpha_\psi]^{\mathrm{T}} \quad (3)$$

where α_x is a confidence value that is a scalar value ranging from 0 to 1 to quantify how certain the value x is, and:

$$\Sigma = [\sigma_x^2 \quad \sigma_y^2 \quad \sigma_z^2 \quad \sigma_\phi^2 \quad \sigma_\theta^2 \quad \sigma_\psi^2]^{\mathrm{T}} \quad (4)$$

where σ_x^2 is a variance of x. The grade is then defined as a matrix (6×2) that contains both vectors (Equation 5):

$$G = [\Phi \quad \Sigma] \quad (5)$$

Both confidence values and variance are adjusted depending on status of the sensor and/or robot. For example, as mentioned earlier, when the differential signals are disrupted, the pose computed by the GPS should be considered inaccurate; hence, the confidence values should be degraded, and the variances should be increased, accordingly. Indeed, assigning these values requires some real-time information that measures status of the sensor and/or robot. Furthermore, it also requires some rule-based knowledge that translates the sensor/robot status into the grade of the pose. We refer to the real-time information of the sensor/robot as *contextual information* (vector C) and the rule-based knowledge as *domain knowledge* (function f_{D}). G for sensor s is hence computed by Equation 6:

$$G_s = f_{\mathrm{D}s}(C_s) \quad (6)$$

Different sensors require different types of contextual information and domain knowledge to compute the grades. Thus, the contents of C and the rules in f_{D} are unique to the sensor type (s). Specific values/rules of C and f_{D} employed during the experiment conducted for this paper is shown in Section IV.C.

At this point, we have X_s (pose) and G_s (grade) for all available sensors. The next step is to adjust the confidence values (α) in G_s based on the types of sensors. Our assumption here is that even when a particular sensor is operating at its best condition (hence 100% confidence value), this sensor may be known to be unreliable if compared with other sensors. Discounting of the α value in G_s can be done by Equation 7:

$$G'_s = G_s \, \Gamma_s \quad (7)$$

where G'_s is the discounted grade, and Γ_s is an weighting matrix that includes predefined discount factor γ_s ranging from 0 to 1 (Equation 8):

$$\Gamma_s = \begin{bmatrix} \gamma_s & 0 \\ 0 & 1 \end{bmatrix} \quad (8)$$

(See Section IV.C. for specific values used for γ in our experiment.)

Given X_s and G_s' for n pose sensors that have updated their measurements since the previous time cycle, the last step here is to compute the best estimate of the current pose from those. We refer to this step as *fusing*, and it is described by:

$$X_{\zeta t} = f_{\mathrm{F}\zeta}(X_{s1t}, G'_{s1t}, X_{s2t}, G'_{s2t}, \ldots, X_{snt}, G'_{snt}) \quad (9)$$

where $X_{\zeta t}$ is a fused pose (the best estimate of the current pose) for the current instant (t); and $f_{\mathrm{F}\zeta}$ is a function that computes the fused pose using fusing method ζ. Currently, we have implemented three types of fusing methods, which are explained in the following section.

It should be noted that our computational method here supports asynchrony of sensor measurements. In other words, different sensors may have different update frequencies (e.g., the GPS usually updates less frequently than the IMU does). Thus, n in Equation 9 can be different from one cycle to another. Furthermore, it should be also noted that dynamically adjusting the contents of *G*-matrix (Equation 6) is our definition of *context-sensitive pose computation*. In order to determine how effective it is to incorporate dynamically changing contextual information upon pose estimation, we evaluated the computational scheme where the *G*-matrix was dynamically computed through Equation 6 against a control scheme where the *G*-matrix was statically specified (Section IV).

B. Fusing Methods

As mentioned above, function $f_{F\zeta}$ computes the output pose (X_ζ) given the input poses from different sensors and their associated grades using fusing method ζ (Equation 9). In this section, we describe how the fusing function can be constructed in this framework by showing how the three types of our fusing methods (Maximum Confidence, Extended Kalman Filter, and Particle Filter) are implemented.

1. Maximum Confidence

The first fusing method, called *Maximum Confidence*, is essentially a greedy method. Each element in the fused pose is copied from the one whose confidence value is the highest among all candidates. Expressing mathematically, let us first define vector a^m as an *n*-length vector that contains the m^{th} element of confidence values stored in the discounted *G*-matrices (Equation 7) for all *n* sensors:

$$a^m = [G'_{s1}(m,1) \quad G'_{s2}(m,1) \quad \cdots \quad G'_{sn}(m,1)]^{\mathrm{T}} \tag{10}$$

For example, a^1 is a vector that contains confidence values of *x* for all *n* sensors, and a^4 is the same for ϕ. The process of fusing via the Maximum Confidence can be then computed by Equation 11:

$$X_\zeta = [X_{\mathrm{IMax}(a^1)}(1) \quad X_{\mathrm{IMax}(a^2)}(2) \quad \cdots \quad X_{\mathrm{IMax}(a^d)}(d)]^{\mathrm{T}} \tag{11}$$

where IMax is a function that takes a vector and returns the index of the element that has the largest value; *d* is the dimension of *X* (*d* = 6 in our case). In the context-sensitive pose computation, dynamic adjustment of the *G*-matrix (Equation 6) affects the values of a^m, and thus it influences the selection of X_ζ pose-components.

2. Extended Kalman Filter

The second type of fusing methods incorporates the Extended Kalman Filter (EKF) as means to estimate the current pose probabilistically. The EKF, an extension of the simple Kalman filter used to handle nonlinearities, is a recursive filter that estimates the current state of a system. Because sensor fusion can be suitably handled by its mathematical formulation [9], both the simple Kalman filter and EKF have been a popular choice in mobile robot localization [11-14]. The pose computation via the EKF has two distinct phases (*prediction* and *update*), which are explained below. (See [15] for more details including the theoretical background.)

In the prediction phase, the estimated pose (X_p) at the current instant *t* based on the process (motion) model is first projected (Equation 12):

$$X_{pt} = f_{\mathrm{PE}}(X_{\zeta t-1}, u_t, w_{t-1}) \tag{12}$$

f_{PE} is a function that implements the process model; the function takes the fused pose computed in the previous cycle ($X_{\zeta t\text{-}1}$), control input (u_t), and estimated process noise ($w_{t\text{-}1}$). u_t is estimated using shaft-encoder readings (Δr_{shatf}) since the shaft-encoder is a standard sensor that is installed on most of the mobile robots in our laboratory. From simple kinematics, the process model can be described by the following equation (Equation 13):

$$X_{pt} = X_{\zeta t-1} + \begin{bmatrix} \Delta l \cos(\theta_{\zeta t-1} + \frac{\Delta\theta_t}{2})\cos(\phi_{\zeta t-1} + \frac{\Delta\phi_t}{2}) \\ \Delta l \cos(\theta_{\zeta t-1} + \frac{\Delta\theta_t}{2})\sin(\phi_{\zeta t-1} + \frac{\Delta\phi_t}{2}) \\ \Delta l \sin(\theta_{\zeta t-1} + \frac{\Delta\theta_t}{2}) \\ \Delta\phi_t \\ \Delta\theta_t \\ \Delta\psi_t \end{bmatrix} + w_{t-1} \tag{13}$$

where Δl is the Euclidean norm of Δr_{shaft} (Equation 14); $\Delta\phi$, $\Delta\theta$, and $\Delta\psi$ are ϕ, θ, and ψ components of Δr_{shaft}, respectively. The values of *w* are all approximated as zero in this phase:

$$\Delta l = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{14}$$

In the prediction phase, the estimated error covariance matrix (*P*) is also projected (Equation 15):

$$P_t = A\,P_{t-1}\,A^{\mathrm{T}} + W\,Q\,W^{\mathrm{T}} \tag{15}$$

where *A* is the Jacobian of f_{PE} with respect to *X*; *W* is the Jacobian of f_{PE} in relation to *w*; and *Q* is the process noise covariance. It should be noted that the *P*-matrix is recursively computed using values from the previous cycle. More specifically, suppose we split up f_{PE} in terms of the elements of the pose to be computed (Equation 16):

$$f_{\mathrm{PE}} = [f_{\mathrm{PE}\,x} \quad f_{\mathrm{PE}\,y} \quad \cdots \quad f_{\mathrm{PE}\,\phi}]^{\mathrm{T}} \tag{16}$$

By the definition of the Jacobian matrix, the *A*-matrix in Equation 15 can be then expressed as:

$$A = \begin{bmatrix} \frac{\partial f_{\mathrm{PE}\,x}}{\partial x} & \cdots & \frac{\partial f_{\mathrm{PE}\,x}}{\partial \phi} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{\mathrm{PE}\,\phi}}{\partial x} & \cdots & \frac{\partial f_{\mathrm{PE}\,\phi}}{\partial \phi} \end{bmatrix} \tag{17}$$

Because of Equation 13, the *W*-matrix in Equation 15 is a 6×6 identity matrix in this case. The *Q*-matrix is computed by Equation 18:

$$Q = \mathrm{Diag}(\Sigma_{shaft}) = \mathrm{Diag}(G'_{shaft}(:,2)) \tag{18}$$

where Diag is a function that returns a square matrix whose diagonal elements are copies of elements in the input vector (other elements are set to be zero); and Σ_{shaft} is the variances of shaft-encoder readings stored in the second column of the discounted G-matrix (the colon indicates that all elements are taken into account).

In the update phase of the EKF, the Kalman gain (K) and final estimates of the current pose (X_ζ) and the error covariance (P) are computed based on the latest measurements (X_s calculated by Equation 2 and arrived to the fuser via Equation 9).

Let us define a function ($f_{\mathrm{H}s}$) that computes an estimated sensor measurement (X_h) for sensor s from an input pose (X) and its estimated error (v):

$$X_{hs} = f_{\mathrm{H}s}(X, v) \tag{20}$$

Assuming the coordinate systems of X_h and X are the same, Equation 20 can be replaced with a simple linear equation (Equation 21):

$$f_{\mathrm{H}s}(X, v) = X + v \tag{21}$$

Given $f_{\mathrm{H}s}$ and P (from Equation 15), the Kalman gain for sensor s is then computed by Equation 22:

$$K_s = \frac{P\, H_s^{\mathrm{T}}}{H_s\, P\, H_s^{\mathrm{T}} + V_s\, R_s\, V_s^{\mathrm{T}}} \tag{22}$$

where H is the Jacobian of f_{H} with respect to X (i.e., an identity matrix in our case because of Equation 21), V is the Jacobian of f_{H} with respect to v (also an identity matrix), and R is the sensor noise covariance for the sensor (Equation 23):

$$R_s = \mathrm{Diag}(\Sigma_s) = \mathrm{Diag}(G_s'(:,2)) \tag{23}$$

Finally, given the Kalman gains for all n sensors, the fused pose (the final estimate of the current pose) of the EKF is computed by the following equation:

$$X_\zeta = X_p + \sum_{s \in S \,\cap\, s \neq shaft} K_s\,(X_s - f_{\mathrm{H}s}(X_p, v_s)) \tag{24}$$

where S is a set of all n sensors that have updated their measurements since the previous cycle. Substituting $f_{\mathrm{H}s}$ with Equation 21, Equation 24 can be simplified as Equation 25 in our implementation:

$$X_\zeta = X_p + \sum_{s \in S \,\cap\, s \neq shaft} K_s\,(X_s - X_p - v_s) \tag{25}$$

Practically, the values of v_S are approximated as zero in this calculation. The error covariance matrix (P) for the next cycle is then updated with Equation 26:

$$P_{next} = (I - \sum_{s \in S \,\cap\, s \neq shaft} K_s\, H_s)\, P \tag{26}$$

It should be noted that, in the context-sensitive pose computation, dynamic adjustment of the G-matrix affects matrices Q (Equation 18) and R (Equation 23).

3. Particle Filter

The third type of the fusing methods implements the Monte Carlo localization (MCL) method, another probabilistic approach; we call this fuser the Particle Filter. Recently, the MCL has been used extensively in the robotics community to fuse sensor readings and represent likelihood distributions over localization space [5, 16-18]. As in the EKF, the Particle Filter computes the current pose (X_ζ) using both the prediction and update phases. (A good review of the MCL method including the theoretical justification can be found in [5, 19].) To carry out the computation, the Particle Filter utilizes a concept called "particle", which is essentially a six-dimensional pose (o) obtained from sampling methods described below. As a fuser, the Particle Filter retains a large number (N) of particles to compute the final pose. Here, we use O, a 6×N matrix, to denote the set of N particles (Equation 27):

$$O = [o_1 \quad o_2 \quad \cdots \quad o_N] \tag{27}$$

The first step in the prediction phase is to project the pose of each particle based on the process model (Equation 28):

$$o_t = f_{\mathrm{PP}}(o_{t-1}, u_t, w_{t-1}) \tag{28}$$

Similar to f_{PE} in Equation 12, f_{PP} is the process model for the Particle Filter, u is the control input (approximated with Δr_{shaft}), and w is the estimated process noise. f_{PP} can be described by the following equation (Equation 29):

$$o_t = o_{t-1} + \begin{bmatrix} \Delta l' \cos(\theta_{ot-1} + \frac{\Delta\theta_t}{2}) \cos(\phi_{ot-1} + \frac{\Delta\phi_t}{2}) \\ \Delta l' \cos(\theta_{ot-1} + \frac{\Delta\theta_t}{2}) \sin(\phi_{ot-1} + \frac{\Delta\phi_t}{2}) \\ \Delta l' \sin(\theta_{ot-1} + \frac{\Delta\theta_t}{2}) \\ \Delta\phi_t \\ \Delta\theta_t \\ \Delta\psi_t \end{bmatrix} \tag{29}$$

where $\Delta\phi$, $\Delta\theta$, and $\Delta\psi$ are ϕ, θ, and ψ components of Δr_{shaft}, respectively; $\Delta l'$ is the Euclidean norm of Δr_{shaft} with the process noise being taken account. More specifically, $\Delta l'$ can be obtained from this equation (Equation 30):

$$\Delta l' = \mathrm{GaussSample}(\Delta l, \sqrt{\mathrm{Mean}(\Sigma_{shaft})}) \tag{30}$$

where GaussSample is a function that draws a sample from the normal density having the mean and standard deviation specified in its first and second input parameters, respectively; Δl is the Euclidean norm of Δr_{shaft} (Equation 14); Mean is a function that returns an average value of its input vector; and Σ_{shaft} is the variances stored in the second column of the discounted G-matrix for the shaft-encoder. Notice that f_{PE} in the EKF (Equation 13) and f_{PP} in the Particle Filter (Equation 29) are similar. However, the difference is that, while the process noise (w) is linearly added to f_{PE}, the process noise in f_{PP} is already taken into account when computing the Euclidean norm of Δr_{shaft} (i.e., $\Delta l'$).

In the update phase of the Particle Filter, the pose of each particle is refined based on the sensor measurements. It is done by first calculating weight (ω_o) for each particle using Equation 31:

$$\omega_o = \prod_{s \in S \cap s \neq shaft} \left(\prod_{i=1}^{d} f_{\mathrm{L}}(o(i), X_s(i), \sqrt{\Sigma_s(i)}) \right) \quad (31)$$

where S is a set of all n sensors that have updated their measurements since the previous cycle; d is the dimension of o (which is six); and f_{L} is a function that returns the likelihood of a sample (specified in the first input parameter) given a measurement (specified in the second input parameter); our f_{L} assumes the normal density, and hence its standard deviation is specified by the third input parameter. Once the weights are computed for all N particles, they are normalized (ω'_o) and combined as a vector (Ω) as shown in Equation 32:

$$\Omega = [\omega'_{o1} \quad \omega'_{o2} \quad \cdots \quad \omega'_{oN}] \quad (32)$$

The next step is re-sampling; a new particle (o_{new}) is drawn from set O using Equation 33:

$$o_{new} = O(:, \mathrm{IRandom}(\Omega)) \quad (33)$$

where IRandom is a function that returns the index of an element that was picked by weighted random sampling from the input vector where the values of the input vector are sampling weights. Equation 33 is repeated N times to form a new set of N particles (O_{new}).

Finally, each component of the final pose (X_ζ) for the Particle Filter is computed by taking the average of the appropriate element in all new particles (Equation 34):

$$X_\zeta = \begin{bmatrix} \mathrm{Mean}(O_{new}(1,:)) \\ \mathrm{Mean}(O_{new}(2,:)) \\ \vdots \\ \mathrm{Mean}(O_{new}(d,:)) \end{bmatrix} \quad (35)$$

where d is the dimension of X_ζ (which is six).

It should be noted that, in the context-sensitive pose computation, dynamic adjustment of the G-matrix affects the value of $\Delta l'$ (Equation 30) and ω_o (Equation 31).

III. Implementation

The context-sensitive pose computation described above, including the three fusing methods, was implemented within *HServer* (Figure 1), one of the components of *MissionLab* [20, 21]. *HServer* is a UNIX process that communicates with attached hardware devices via serial ports or TCP/IP socket connections. For example, *HServer* can control a physical robot by executing commands that are issued from *Robot Executable*, another *MissionLab* process where behaviors are computed.

Another functionality of *HServer* is to marshal sensory information from attached sensors and report it to *Robot Executable*. Such sensory information includes the pose of the robot. More specifically, pose computation is done in a module called *PoseCalculator* in *HServer*. *PoseCalculator* gathers the latest readings from the GPS, compass, IMU and shaft-encoder (if they are enabled), and attempts to compute the best estimate of the current pose based on those. Indeed, *PoseCalculator* is where the proposed context-sensitive pose computation scheme was integrated. Figure 2 depicts the process of the context-sensitive pose computation in *PoseCalculator*. As shown in the figure, a module called *Pose Manager* implements Equations 2 and 6 in order to compute the converted pose for the sensor (X_s) and its (initial) grade (G_s), respectively. Discounting of the grade (G'_s) is done in *Sensory Situational Context* (Equation 7). All asynchronously computed X_s and G'_s from different sensors arrive at *Sensory Data Bus*, from which *Sensor Fuser* grabs the latest values every computational cycle. *Sensor Fuser* (Equation 9) is where the fused pose (X_ζ) is computed by employing one of the three fuser methods (i.e., Maximum Confidence, EKF, and Particle Filter).

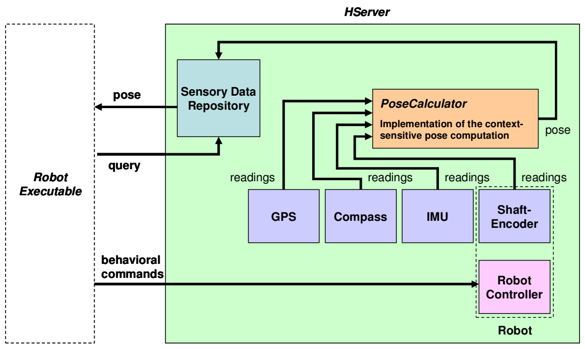

Figure 1: *HServer*

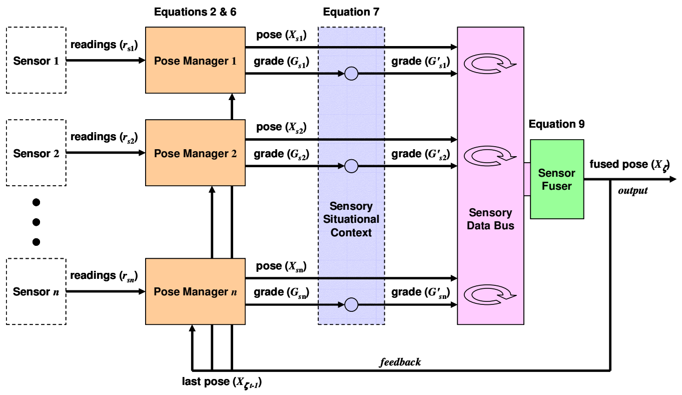

Figure 2: *PoseCalculator*

IV. Evaluation

A. Experimental Hypotheses

An outdoor experiment was conducted in order to determine whether incorporation of contextual information and domain knowledge helps the accuracy of localization. More specifically, the experiment was designed to assess the following hypotheses:

Hypothesis 1:

If adequate real-time contextual information and domain knowledge is incorporated, the robot's pose computed by a simple greedy fusing method can be as accurate as the one computed by the conventional probabilistic localization methods.

Hypothesis 2:

The conventional probabilistic localization methods can improve their accuracy of the robot's pose if real-time contextual information and domain knowledge are incorporated.

B. *Experimental Area*

The outdoor experiment was conducted at the top level of a parking deck in the Georgia Tech campus to test the above hypotheses. As shown in Figure 3, the area was about 60 meters wide and about 90 meters long. Six waypoints were selected in the area: Start Point <53.1, 37.4>, A <53.1, 63.3 >, B <53.1, 97.7 >, C <23.3, 97.7>, D <23.3, 63.3>, and E <23.3, 37.4> (Note: the coordinates <*x*, *y*> are in meters).

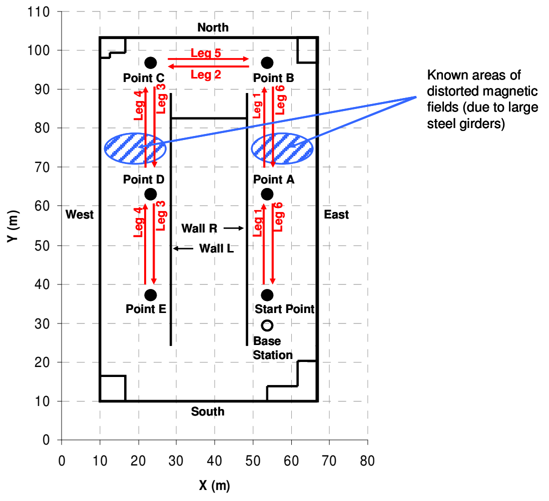

Figure 3: Experimental Area

C. *Contextual Information and Domain Knowledge Used*

Tables 1 and 2 show the values/rules of contextual information (C) and domain knowledge (function f_D) being used during the outdoor experiment when they were dynamically adjusted. The values of the grade (G) when the G-matrix is statically defined are shown in Table 3. Table 4 shows the values of discount factors (Equation 8). All these values used here are determined based on hardware specifications and through trial-and-error during the testing. However, it should be noted that calibration of such parameters is a delicate process, and we do not guarantee that they are perfectly optimized. Nevertheless, they are, to the best of our knowledge, adequately tuned.

Table 1: Real-Time Contextual Information (C)

GPS	Shaft-Encoder
C_1 = RT-20 value C_2 = translational speed of robot C_3 = angular speed of robot	C_1 = translational speed of robot C_2 = angular speed of robot
Compass	**IMU**
C_1 = angular speed of robot	C = ∅ (empty set)

Table 2: Dynamical Adjustment of Grades (G) via Domain Knowledge (f_D)

GPS	Compass

if $C_1 = 0$ then
  $\alpha_{\{x,y,z\}} = 1$, $\sigma^2_{\{x,y,z\}} = 0.2$
  if C_2 >= 0 and $C_3 = 0$
    $\alpha_{\{\phi,\theta,\psi\}} = 1$, $\sigma^2_{\{\phi,\theta,\psi\}} = 1$
  else
    $\alpha_{\{\phi,\theta,\psi\}} = 0$, $\sigma^2_{\{\phi,\theta,\psi\}} = 129600$
  end
else if $C_1 = 1$ then
  $\alpha_{\{x,y,z\}} = 1.0$, $\sigma^2_{\{x,y,z\}} = 0.3$
  if C_2 >= 0 and $C_3 = 0$
    $\alpha_{\{\phi,\theta,\psi\}} = 1$, $\sigma^2_{\{\phi,\theta,\psi\}} = 4$
  else
    $\alpha_{\{\phi,\theta,\psi\}} = 0$, $\sigma^2_{\{\phi,\theta,\psi\}} = 129600$
  end
end
	$\alpha_{\{x,y,z\}} = 0$, $\alpha_{\{\phi,\theta,\psi\}} = 1$
	$\sigma^2_{\{x,y,z\}} = 1000000$
	if $C_1 = 0$ then
	  $\sigma^2_{\{\phi,\theta,\psi\}} = 360$
	else
	  $\sigma^2_{\{\phi,\theta,\psi\}} = 129600$
	End

	Shaft-Encoder
	$\alpha_{\{x,y,z\}} = 0$, $\alpha_{\{\phi,\theta,\psi\}} = 1$ if $C_1 = 0$ and $C_2 = 0$ then $\sigma^2_{\{x,y,z\}} = 0.0001$, $\sigma^2_{\{\phi,\theta,\psi\}} = 0.001$ else $\sigma^2_{\{x,y,z\}} = 0.001$, $\sigma^2_{\{\phi,\theta,\psi\}} = 1$ End
Note: GPS readings are ignored when RT-20 is 2 or greater (same for the static case).	**IMU**
	$\alpha_{\{x,y,z\}} = 0$, $\alpha_{\{\phi,\theta,\psi\}} = 1$ $\sigma^2_{\{x,y,z\}} = 1000000$, $\sigma^2_{\{\phi,\theta,\psi\}} = 1$

Table 3: Static Grades (G)

GPS	Shaft-Encoder
$\alpha_{\{x,y,z\}} = 1$, $\alpha_{\{\phi,\theta,\psi\}} = 0$ $\sigma^2_{\{x,y,z\}} = 0.25$, $\sigma^2_{\{\phi,\theta,\psi\}} = 3600$	$\alpha_{\{x,y,z\}} = 1$, $\alpha_{\{\phi,\theta,\psi\}} = 1$ $\sigma^2_{\{x,y,z\}} = 0.01$, $\sigma^2_{\{\phi,\theta,\psi\}} = 1$
Compass	**IMU**
$\alpha_{\{x,y,z\}} = 0$, $\alpha_{\{\phi,\theta,\psi\}} = 0$ $\sigma^2_{\{x,y,z\}} = 1000000$, $\sigma^2_{\{\phi,\theta,\psi\}} = 8100$	$\alpha_{\{x,y,z\}} = 0$, $\alpha_{\{\phi,\theta,\psi\}} = 1$ $\sigma^2_{\{x,y,z\}} = 1000000$, $\sigma^2_{\{\phi,\theta,\psi\}} = 1$

Table 4: Discount Factors (γ) in the Weighting Matrix (Γ)

GPS	Compass	IMU	Shaft-Encoder
$\gamma_{GPS} = 0.9$	$\gamma_{compass} = 0.8$	$\gamma_{IMU} = 0.75$	$\gamma_{shaft} = 0.5$

D. *Hardware*

Both *HServer* and *Robot Executable* ran on the onboard dual processors (Pentium III, 1 GHz) of an ATRV-Jr (iRobot Corporation) during execution. The ATRV-Jr was equipped with a differential GPS (ProPak by NovAtel, Inc.), a compass (3DM-G by MicroStrain, Inc.), an IMU (IMU400CC-200 by Crossbow Technology, Inc.), and internal shaft-encoders. In addition, two sets of onboard laser scanners (LMS 200-30106 by SICK, Inc.) were used to measure the ground truth of the current pose (explained below). The base station for the differential GPS was placed 8 meters south of Start Point.

E. *Methods*

In order to test the above experimental hypotheses, an autonomous waypoint-following mission was created and executed by *MissionLab*. In this mission, the robot followed the six points by the order of Start Point, A, B, C, D, E, D, C, B, A, and then back to Start Point. Here, the segment from Start Point to Point B is called Leg 1; the segments B → C, C → E, E → C, C → B, and B → Start Point are called Legs 2, 3, 4, 5, and 6, respectively. During the mission, the robot was always commanded to run with its full-speed (approximately 2 m/s) unless making a point-turn at a waypoint.

To ensure GPS disruption, the differential signals from the base station to the robot were physically cut off when the robot was at Leg 2 (from Point B to Point C). This allowed the RT-20 value [7] of the differential GPS to degrade gradually from 0 to 8, simulating realistic deterioration of the GPS accuracy. During the return trip, the transmission of differential signals was resumed at Leg 5 (from Point C to Point B). Furthermore, during Legs 1, 3, 4, and 6, the robot had to go through the areas where the magnetic fields were distorted by steel girders laying underneath the floor, affecting the performance of the compass in a nonlinear manner.

In order to determine the accuracy of the pose computed by the system with respect to the ground truth, the computed pose and readings from laser scanners were recorded for every second during Legs 1, 4, and 6. The set of the two lasers can acquire 722 readings (covering 360°) with an update rate of four times per second. As shown in Figure 3, during Legs 1, 4, and 6, the robot moved along with the flat walls laying in the North-South direction, namely, Wall R and Wall L. Since the coordinates of those walls were known, one can calculate the expected distance from the pose to the wall, and compare it with the actual distance measured by the laser scanners. The difference between the expected distance and the actual distance is defined here as a distance error. Moreover, since the angle of the direction of which the laser scanners found the closest distance to the wall was known, one can also calculate the actual heading of the robot with respect to the wall (i.e., with respect to the ground truth). The difference between the actual heading and the expected heading computed by the system is defined here as a heading error. In others words, in this experiment, the accuracy of the pose computed by the system was determined by the distance and heading errors.

Two conditions were tested for each of the three fusing methods (the Maximum confidence, EKF, and Particle Filter); the first condition is context-free, that is when the *G*-matrix (Equation 5) is fixed; and the second condition is context-sensitive, that is when the *G*-matrix is dynamically adjusted by Equation 6 (explained in Section IV.C). In order to be statistically significant, 20 runs of the waypoint-following mission were recorded for every condition (i.e., the total of 120 runs were recorded for the six conditions). As a standard practice, we consider a difference of two means to be significant if the *p*-value of the associated ANOVA test is less than 0.05 (5%).

F. Results

The average heading and distance errors for all conditions are plotted against the leg number in Figures 4 and 5, respectively. The error bars in the figures denote 95 percent confidence intervals.

Regarding Hypothesis 1, the greedy fusing method (Maximum Confidence) with the context-sensitive condition (dynamic *G*-matrix) was compared against the context-free (fixed *G*-matrix) conventional probabilistic localization methods: namely, the EKF and Particle Filter. At Leg 1, the one-way ANOVA test showed that the context-sensitive Maximum Confidence had significantly less heading error than the context-free EKF (F = 4.167, p < 0.048; the error bars slightly overlapped). However, there was no significant difference if compared to the heading error produced by the context-free Particle Filter. At Leg 4 (the GPS shadow), the average heading error of the context-sensitive Maximum Confidence was not significantly different from the context-free EKF and Particle Filter. At Leg 6 (the final leg), the context-sensitive Maximum Confidence produced significantly less heading error compared to the context-free EKF (F = 9.845, p < 0.003) and the context-free Particle Filter (F = 6.961, p < 0.012; error bars slightly overlapped). However, the context-sensitive Maximum Confidence had no significant distance errors over the context-free EKF and Particle Filter at all legs. Overall, these results support the first hypothesis and indicate that with the addition of the proper contextual information and domain knowledge, a simple greedy fusing method can achieve accuracies meeting and in some cases exceeding that of the two conventional probabilistic filters used in these experiments.

On the other hand, the second hypothesis was not supported by the current data. In other words, in both probabilistic filters, the heading and distance errors when using contextual information in the form of dynamic variances (context-sensitive) did not exhibit significant differences if compared with the context-free ones.

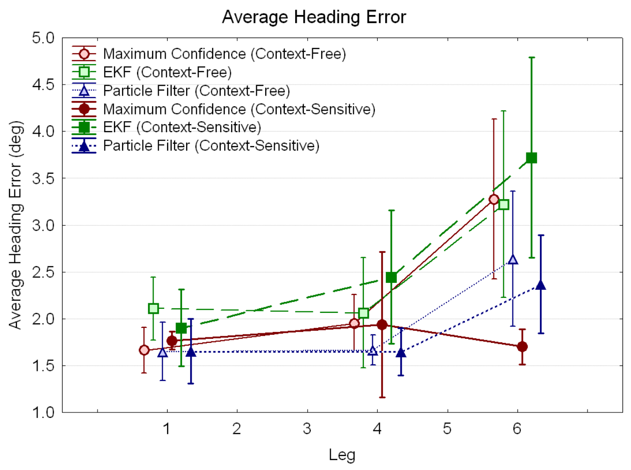

Figure 4: Average Heading Error

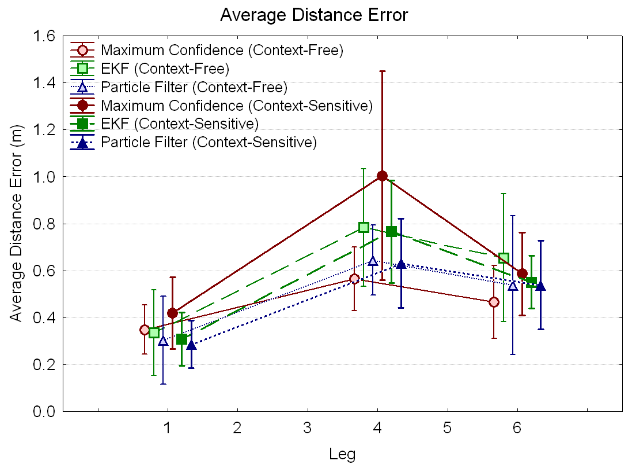

Figure 5: Average Distance Error

V. Conclusions and Future Work

This work details context-sensitive pose computation and empirically evaluates it within the framework of a localization task in an urban environment. In this task, the robot must provide accurate localization information even in the event of sensor drop-out and in the presence of non-linear error sources over the span of numerous waypoint-following missions. The utility of the computational scheme is illustrated by the performance of the Maximum Confidence, based purely on this context-sensitive information, matches or even exceeds the performance of the conventional probabilistic localization methods (i.e., the EKF and Particle Filter). Further, it has been shown to be robust under a wide variety of sensor noise such as that produced by GPS dropout and the non-linear sensor noise produced by the large steel girders present in the experimental arena.

On the other hand, our evaluation determined that the performances of the probabilistic filters are not affected significantly by the utilization of the contextual information in the form of dynamic variances. A few causes are speculated: (1) The domain knowledge (i.e., adjustment of variances) was not adequate; (2) the probabilistic filters were so efficiently formulated that the extra information did not add any value; or (3) the navigational task and/or environment was too simple. An additional set of experiments should be conducted in order to solve this predicament.

Furthermore, in this study, the performance of our computational scheme was measured by the accuracy of the output pose with respect to the ground truth. In a real urban outdoor navigational task, however, how effectively the robot can accomplish the assigned task is also important; such effectiveness includes its ability to arrive to a waypoint quickly or ability to overcome presence of static and/or dynamic obstacles without being disoriented. In other words, the computational scheme should be also evaluated in terms of behavioral accuracies.

A possible extension of this work relates to Murphy's action-oriented perceptual architecture [8] described in Section I. By adding some high-level planning mechanism, dynamical switching or even blending of the fusing methods themselves is also possible, and such an extension may be advantageous in more complex and/or dynamic environments.

Acknowledgment

The authors would like to thank Alan Wagner, Yang Chen and Alex Stoytchev for their support on the experiment.

References

[1] T. R. Collins, R. C. Arkin, M. J. Cramer, and Y. Endo, "Field Results for Tactical Mobile Robot Missions," presented at *Unmanned Systems*, Orlando, Fla., Assoc. for Unmanned Vehicle Systems International, 2000.

[2] L. Chaimowicz, A. Cowley, D. Gomez-Ibanez, B. Grocholsky, M. A. Hsieh, H. Hsu, J. F. Keller, V. Kumar, R. Swaminathan, and C. J. Taylor, "Deploying Air-Ground Multirobot Teams in Urban Environments," presented at *Multirobot Workshop*, Washington, D.C., 2005.

[3] J. S. Gutmann and D. Fox, "An Experimental Comparison of Localization Methods Continued," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and System*, 2002, pp. 454-459.

[4] S. Thrun, "Probabilistic Algorithms in Robotics," in *AI Magazine*, vol. 21, 2000, pp. 93-109.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo Localization for Mobile Robots," *Proc. IEEE Int'l Conf. Robotics and Automation*, 1999, pp. 1322-1328.

[6] S. Se, D. Lowe, and J. Little, "Mobile Robot Localization and Mapping with Uncertainty Using Scale-Invariant Landmarks," *Int'l J. Robotics Research*, vol. 21, 2002, pp. 305-360.

[7] T. J. Ford and J. Neumann, "NovAtel's RT-20^{TM} - A Real Time Floating Ambiquity Positioning System," presented at *Int'l Technical Meeting of the Satellite Division of the Institute of Navigation*, Salt Lake City, Utah, 1994.

[8] R. R. Murphy, *An Architecture for Intelligent Robotics Sensor Fusion*, Ph.D. Thesis, College of Computing, Georgia Institute of Technology, 1992

[9] H. P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots," in *AI Magazine*, 1988.

[10] S. Thrun, "Robotic Mapping: A Survey," in *Exploring Artificial Intelligence in the New Millenium*, Morgan Kaufmann, 2002.

[11] S. I. Roumeliotis and G. A. Bekey, "Bayesian Estimation and Kalman Filtering: a Unified Framework for Mobile Robot Localization," *Proc. IEEE Int'l Conf. Robotics and Automation*, 2000, pp. 2985-2992.

[12] H. Chung, L. Ojeda, and J. Borenstein, "Sensor Fusion for Mobile Robot Dead-Reckoning with a Precision-Calibrated Fiber Optic Gyroscope," *Proc. IEEE Int'l Conf. Robotics and Automation*, 2001, pp. 3588-3599.

[13] R. R. Negenborn, "Robot Localization and Kalman Filters," Utrecht Univ., Utrecht, Netherlands, Master's thesis INF/SCR-0309, 2003.

[14] D. Kurth, "Range-Only Robot Localization and SLAM with Radio," Robotics Institute, Carnegie Mellon Univ., Pittsburgh, Master's thesis CMU-RI-TR-04-29, May 2004.

[15] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," presented at *SIGGRAPH 2001 Course 8*, Los Angeles, ACM Press, 2001.

[16] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel Approach to Nonlinear/non-Gaussian Bayesian State Estimation," *Proc. IEE Proceedings on Radar and Signal Processing*, 1993, pp. 107-113.

[17] M. Isard and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," *Proc. European Conf. Computer Vision*, Cambridge, UK, 1996, pp. 343-356.

[18] J. Carpenter, P. Clifford, and P. Fernhead, "An Improved Particle Filter for Non-Linear Problems," Department of Statistics, Univ. of Oxford, Technical Report 1997.

[19] A. Doucet, "On Sequential Monte Carlo Sampling Methods for Bayesian Filtering," *Statistics and Computing*, vol. 10, 2000, pp. 197-208.

[20] D. C. MacKenzie, R. C. Arkin, and J. Cameron, "Multiagent Mission Specification and Execution," *Autonomous Robots*, vol. 4, 1997, pp. 29-57.

[21] *MissionLab: User Manual for MissionLab 6.0.* Georgia Tech Mobile Robot Laboratory, College of Computing, Georgia Institute of Technology, Atlanta, GA, 2003.