© ARTVILLE

Accurate Position Estimation and Propagation in Autonomous Vehicles

State Your Position

BY CARLOS ALBORES BORJA, JOSEP M. MIRATS TUR, AND JOSÉ LUIS GORDILLO

Autonomous vehicles (AVs) and mobile robots are developed to carry out specific tasks autonomously, ideally with no human intervention at all [1], [2]. These kinds of vehicles, whether AVs or mobile robots, has generated great interest in recent years, for instance, because of their capacity to work in remote or harmful environments where humans cannot enter because of the extreme operating conditions [3]. We are differentiating a mobile robot (designed from scratch for a specific task) from an AV (a commercial vehicle with proper sensory and control systems added so as to be autonomous). However, in the sequel, for the sake of simplicity, we will use indistinctly one or the other term. Several applications are found in the literature, varying from material transportation in a common industrial environment [4] to the exciting exploration of a far planet surface [5], [6].

A common issue for any application using mobile robots is navigation [7], including localization, path planning, path execution, and obstacle avoidance [8]. To correctly perform the assigned task, the robot must estimate its position and orientation, i.e., its pose, in the environment. Whichever the sensory system used, the computation of the robot's pose estimate will never be error free; there is always an associated uncertainty to this estimate due to different factors such as the inaccuracy of the used sensors or incomplete environment information. Therefore, localization involves two different challenges: computing a pose estimate and its associated uncertainty. The first problem has received a lot of attention in the literature during the last decades; for instance, see [9] for a recent approach.

This article is centered on how we can accurately estimate the robot pose uncertainty and how this uncertainty is propagated to future states (the robot state, in this context, is its pose). Some attention has been paid to this issue in the past. The first approach considering the uncertainty of position estimation was given in [10]. A min/max error bound approach is proposed resulting in bigger and bigger circles in the x-y plane representing the possible positions for the robot. Those circles are computed as projections of cylinders in the configuration space. Basically, the same approach was independently derived in [11] using a scalar as an uncertainty measure in the plane position but without reference to the orientation error. Then the idea of representing this uncertainty came up

Digital Object Identifier 10.1109/MRA.2009.932523

1070-9932/09/$25.00©2009 IEEE

by means of a covariance matrix [12]. In [13], a method for determining the pose uncertainty covariance matrix using a Jacobian approach is given for the special case of circular arc motion with a constant radius of curvature. In [14], the odometry error is modeled for a synchronous drive system depending on parameters characterizing both systematic and nonsystematic components. An important work addressing the problem of understanding the relation between odometry sensor readings and the resultant error in the computed vehicle pose is [15].

In [16] and [17], we provided a general expression for the uncertainty in the pose estimate of a mobile vehicle using the covariance definition. This covariance matrix was derived from a given vehicle kinematic model, and for each time step, the total covariance was calculated under the assumption that the cross-covariance between the previous position and the actual increment of position was zero. This assumption, although widely considered, is not generally true, because the orientation errors in the robot's previous position do not affect the computation of the actual increment of position, leading to inaccurate pose uncertainty estimation when propagating it to future states.

The main contribution of this article is the proposal of a novel solution for the calculation of such cross-covariance terms. In this way, we are able to catch the highly nonlinear behavior of the pose uncertainty while accurately estimating it. The basic idea is to fit the covariance matrix for the previous pose using a set of equations obtained by eigen decomposition. The cross-covariance terms are then derived using these set of equations together with the already-known expressions for the vehicle pose increments.

To validate this method, we compare our pose uncertainty calculation and propagation approach to widely accepted existing propagation methods such as the classical Jacobian approach (used, for instance, by the extended Kalman filter) and the scaled unscented transformation (UT) [used by the unscented Kalman filter (UKF)].

The article is organized as follows. The "Current Pose Uncertainty: Accurate Computation and Propagation" section develops the cross-covariance terms solution. Next, in the "Approach Validation" section, the results for the pose uncertainty obtained in different vehicle paths are compared using three methods: the Jacobian matrix method [18], which is the classical way to obtain the estimation for the incremental pose uncertainty, the scaled UKF [19], [20], and the presented formulation. The comparison is made to the uncertainty computed using a Monte Carlo simulation of the run paths. Finally, the last section addresses conclusions and future work.

Current Pose Uncertainty: Accurate Computation and Propagation

We are interested on how the robot's pose uncertainty can be accurately estimated and propagated to future states. In [16], this uncertainty was computed using (1), considering the errors of the previous pose estimate, $\hat{P}_{k-1}$, and the actual estimated increments of pose, $\Delta\hat{P}_k$, to be independent. However, as the angle (and its error) of the previous pose is considered in the computation of both, $\Delta\hat{x}_k$ and $\Delta\hat{y}_k$, there is a nonmodeled cross-covariance between $\hat{P}_{k-1}$ and $\Delta\hat{P}_k$. The fact of ignoring the dependency between $\hat{P}_{k-1}$ and $\Delta\hat{P}_k$ causes the considered error model to not completely catch the high nonlinear behavior of the pose uncertainty.

$$\begin{aligned}\mathrm{cov}(\hat{P}_k) = \mathrm{cov}(\hat{P}_{k-1}) + \mathrm{cov}(\Delta\hat{P}_k) + \mathrm{cov}(\hat{P}_{k-1}, \Delta\hat{P}_k^T) \\ + \mathrm{cov}(\Delta\hat{P}_k, \hat{P}_{k-1}^T).\end{aligned} \tag{1}$$

We propose here an efficient method to compute such cross-covariance between the previous pose $\hat{P}_{k-1}$ and the actual pose increment $\Delta\hat{P}_k$. Obviously, the optimal way to calculate the covariance of $\hat{P}_k$ is to consider, at each sample time k, the full mathematical expressions for all the pose increments $\Delta\hat{P}_1$, $\Delta\hat{P}_2$, $\Delta\hat{P}_3, \ldots, \Delta\hat{P}_{k-2}$, $\Delta\hat{P}_{k-1}$, and $\Delta\hat{P}_k$, which may be obtained using

$$\begin{aligned}\mathrm{cov}(\hat{P}_k) &= \mathrm{cov}(\hat{P}_0 + \Delta\hat{P}_1 + \cdots + \Delta\hat{P}_k) \\ &= \mathbf{E}[(\hat{P}_0 + \Delta\hat{P}_1 + \cdots + \Delta\hat{P}_k) \\ &\quad \times (\hat{P}_0 + \Delta\hat{P}_1 + \cdots + \Delta\hat{P}_k)^T] \\ &\quad - \mathbf{E}[(\hat{P}_0 + \Delta\hat{P}_1 + \cdots + \Delta\hat{P}_k)] \\ &\quad \mathbf{E}[(\hat{P}_0 + \Delta\hat{P}_1 + \cdots + \Delta\hat{P}_k)^T].\end{aligned} \tag{2}$$

Since the computational complexity of such expression increases exponentially, it is not feasible for a real-time performance to calculate the covariance in this way. Hence, an incremental approach is preferred, as stated in (1).

To obtain an expression for the cross-covariance terms and because of the mathematical complexity of the problem, an equation representation of the covariance at step $k-1$ is obtained, by assuming an error vector for the robot's pose in time $k-1$.

$$\hat{P}_{k-1} = P_{k-1} + \varepsilon_{P_{k-1}}, \tag{3}$$

$$\varepsilon_{P_{k-1}} = [\,\varepsilon_{x_{k-1}} \quad \varepsilon_{y_{k-1}} \quad \varepsilon_{\theta_{k-1}}\,]^T, \tag{4}$$

$$\mathbf{E}[\varepsilon_{P_{k-1}}] = [\,0 \quad 0 \quad 0\,]^T. \tag{5}$$

The error $\varepsilon_{P_{k-1}}$ can be seen as a vector composed of three different independent (zero mean, Gaussian, and orthogonal) errors γ_1, γ_2, and γ_3. Note that these three independent errors, when multiplied by certain matrix Q, equal the error in each of the considered axes x, y, θ:

$$\gamma = [\,\gamma_1 \quad \gamma_2 \quad \gamma_3\,]^T, \tag{6}$$

$$\varepsilon_{P_{k-1}} = \begin{bmatrix}\varepsilon_{x_{k-1}} \\ \varepsilon_{y_{k-1}} \\ \varepsilon_{\theta_{k-1}}\end{bmatrix} = Q\gamma = \begin{bmatrix}A\gamma_1 + B\gamma_2 + C\gamma_3 \\ D\gamma_1 + E\gamma_2 + F\gamma_3 \\ G\gamma_1 + H\gamma_2 + I\gamma_3\end{bmatrix}. \tag{7}$$

Since the considered errors are zero mean, $\mathbf{E}[\varepsilon_{P_{k-1}}] = 0$, from the covariance definition, the expression for the covariance matrix associated to $\hat{P}_{k-1}$ is obtained:

$$\begin{aligned}\mathrm{cov}(\hat{P}_{k-1}) &= \mathbf{E}[(\hat{P}_{k-1} - \mathbf{E}[\hat{P}_{k-1}])(\hat{P}_{k-1} - \mathbf{E}[\hat{P}_{k-1}])^T], \\ &= \mathbf{E}[(P_{k-1} + \varepsilon_{P_{k-1}} - \mathbf{E}[P_{k-1} + \varepsilon_{P_{k-1}}]) \\ &\quad \times (P_{k-1} + \varepsilon_{P_{k-1}} - \mathbf{E}[P_{k-1} + \varepsilon_{P_{k-1}}])^T],\end{aligned}$$

$$
\begin{aligned}
&= \mathbf{E}[(P_{k-1} + \varepsilon_{P_{k-1}} - \mathbf{E}[P_{k-1}] - \mathbf{E}[P_{k-1} + \varepsilon_{P_{k-1}}]) \\
&\quad \times (P_{k-1} + \varepsilon_{P_{k-1}} - \mathbf{E}[P_{k-1}] - \mathbf{E}[P_{k-1} + \varepsilon_{P_{k-1}}])^T], \\
&= \mathbf{E}[(P_{k-1} + \varepsilon_{P_{k-1}} - P_{k-1} - (0)) \\
&\quad \times (P_{k-1} + \varepsilon_{P_{k-1}} - P_{k-1} - (0))^T], \\
&= \mathbf{E}[(\varepsilon_{P_{k-1}})(\varepsilon_{P_{k-1}})^T]. \qquad (8)
\end{aligned}
$$

On multiplying these terms and reducing,

$$
\begin{aligned}
\mathrm{cov}(\hat{P}_{k-1}) &= \mathbf{E}[(\varepsilon_{P_{k-1}})(\varepsilon_{P_{k-1}})^T] \\
&= \mathbf{E}[Q\gamma(Q\gamma)^T] \\
&= \mathbf{E}[Q\gamma\gamma^T Q^T], \\
&= \mathbf{E}\left[Q \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix}^T Q^T \right] \\
&= \mathbf{E}\left[Q \begin{bmatrix} \gamma_1\gamma_1 & \gamma_1\gamma_2 & \gamma_1\gamma_3 \\ \gamma_2\gamma_1 & \gamma_2\gamma_2 & \gamma_2\gamma_3 \\ \gamma_3\gamma_1 & \gamma_3\gamma_2 & \gamma_3\gamma_3 \end{bmatrix} Q^T \right], \\
&= Q\left(\mathbf{E}\begin{bmatrix} \gamma_1\gamma_1 & \gamma_1\gamma_2 & \gamma_1\gamma_3 \\ \gamma_2\gamma_1 & \gamma_2\gamma_2 & \gamma_2\gamma_3 \\ \gamma_3\gamma_1 & \gamma_3\gamma_2 & \gamma_3\gamma_3 \end{bmatrix} \right) Q^T. \qquad (9)
\end{aligned}
$$

Recall that the considered errors γ_1, γ_2, and γ_3 are independent and zero mean; hence, the resultant covariance matrix can be written as follows:

$$
\begin{aligned}
\mathrm{cov}(\hat{P}_{k-1}) &= Q\left(\mathbf{E}\begin{bmatrix} \gamma_1\gamma_1 & \gamma_1\gamma_2 & \gamma_1\gamma_3 \\ \gamma_2\gamma_1 & \gamma_2\gamma_2 & \gamma_2\gamma_3 \\ \gamma_3\gamma_1 & \gamma_3\gamma_2 & \gamma_3\gamma_3 \end{bmatrix} \right) Q^T, \\
&= Q\begin{bmatrix} \sigma_{\gamma_1}^2 & 0 & 0 \\ 0 & \sigma_{\gamma_2}^2 & 0 \\ 0 & 0 & \sigma_{\gamma_3}^2 \end{bmatrix} Q^T. \qquad (10)
\end{aligned}
$$

Now, since $\mathrm{cov}(\hat{P}_{k-1})$ is a positive semidefinite Hermitian matrix, it can be decomposed in the form

$$
\begin{aligned}
\mathrm{cov}(\hat{P}_{k-1}) &= Q\lambda Q^T \\
&= Q\begin{bmatrix} \lambda_{1,1} & 0 & 0 \\ 0 & \lambda_{2,2} & 0 \\ 0 & 0 & \lambda_{3,3} \end{bmatrix} Q^T, \qquad (11)
\end{aligned}
$$

where Q is the eigenvector matrix and λ is a diagonal matrix containing the eigenvalues of $\mathrm{cov}(\hat{P}_{k-1})$. This form is equal to the one in (9); thus, the matrix Q and the three variances $\sigma_{\gamma_1}^2$, $\sigma_{\gamma_2}^2$, and $\sigma_{\gamma_3}^2$, which equals $\lambda_{1,1}$, $\lambda_{2,2}$, and $\lambda_{3,3}$, respectively, can be obtained by the eigen decomposition of the covariance matrix. Note that the product $Q\gamma$ could also be seen as a rotation of the three orthogonal errors considered in $\varepsilon_{P_{k-1}}$ with respect to the axes x, y, and θ. So this term is capable of modeling the rotation of the robot pose uncertainty as it runs its path.

Now, we are ready to obtain an expression for the cross-covariance terms

$$
\begin{aligned}
&\mathrm{cov}(\hat{P}_{k-1}, \Delta\hat{P}_k^T) \\
&\quad = \mathbf{E}[(\hat{P}_{k-1} - \mathbf{E}[\hat{P}_{k-1}])(\Delta\hat{P}_k - \mathbf{E}[\Delta\hat{P}_k])^T], \\
&\quad = \mathbf{E}[(P_{k-1} + \varepsilon_{P_{k-1}} - \mathbf{E}[P_{k-1}] - \mathbf{E}[\varepsilon_{P_{k-1}}])(\Delta\hat{P}_k - \mathbf{E}[\Delta\hat{P}_k])^T], \\
&\quad = \mathbf{E}[(P_{k-1} + \varepsilon_{P_{k-1}} - P_{k-1} - (0))(\Delta\hat{P}_k - \mathbf{E}[\Delta\hat{P}_k])^T], \\
&\quad = \mathbf{E}[(\varepsilon_{P_{k-1}})(\Delta\hat{P}_k - \mathbf{E}[\Delta\hat{P}_k])^T] \\
&\quad = \mathbf{E}[\varepsilon_{P_{k-1}}\Delta\hat{P}_k^T - \varepsilon_{P_{k-1}}\mathbf{E}[\Delta\hat{P}_k]^T], \\
&\quad = \mathbf{E}[\varepsilon_{P_{k-1}}\Delta\hat{P}_k^T] - \mathbf{E}[\varepsilon_{P_{k-1}}]\mathbf{E}[\Delta\hat{P}_k]^T \\
&\quad = \mathbf{E}[\varepsilon_{P_{k-1}}\Delta\hat{P}_k^T] - (0)\mathbf{E}[\Delta\hat{P}_k]^T, \\
&\quad = \mathbf{E}[\varepsilon_{P_{k-1}}\Delta\hat{P}_k^T], \qquad (12)
\end{aligned}
$$

$$
\mathbf{E}[\varepsilon_{P_{k-1}}\Delta\hat{P}_k^T] = \mathbf{E}\begin{pmatrix} \varepsilon_{x_{k-1}}\Delta\hat{x}_k & \varepsilon_{x_{k-1}}\Delta\hat{y}_k & \varepsilon_{x_{k-1}}\Delta\hat{\theta}_k \\ \varepsilon_{y_{k-1}}\Delta\hat{x}_k & \varepsilon_{y_{k-1}}\Delta\hat{y}_k & \varepsilon_{y_{k-1}}\Delta\hat{\theta}_k \\ \varepsilon_{\theta_{k-1}}\Delta\hat{x}_k & \varepsilon_{\theta_{k-1}}\Delta\hat{y}_k & \varepsilon_{\theta_{k-1}}\Delta\hat{\theta}_k \end{pmatrix}. \qquad (13)
$$

Just as an example, we develop here the complete calculation for the first term in (13). See the "Approach Validation" section and "Kinematic Model of the AV" for details on this particularization.

$$
\mathbf{E}[\varepsilon_{x_{k-1}}\Delta\hat{x}_k^T] = \mathbf{E}\left[\varepsilon_{x_{k-1}}\Delta\hat{d}_k \left[\cos(\hat{\theta}_{k-1}) - \frac{\Delta\hat{\theta}_k}{2}\sin(\hat{\theta}_{k-1}) \right] \right]. \qquad (14)
$$

Now we need to consider that the estimate of orientation at previous step $\hat{\theta}_{k-1}$ has an associated error, which, without loss of generality, is modeled as Gaussian. If we consider independency between previous position errors and the sensor's increment measurements, we can group them separately and then separate the estimates.

$$
\begin{aligned}
\mathbf{E}[\varepsilon_{x_{k-1}}\Delta\hat{x}_k^T] &= \cdots = \mathbf{E}[\varepsilon_{x_{k-1}}\cos(\varepsilon_{\theta_{k-1}})] \\
&\quad \mathbf{E}\left[\Delta\hat{d}_k \left[\cos(\theta_{k-1}) - \frac{\Delta\hat{\theta}_k}{2}\sin(\theta_{k-1}) \right] \right] \\
&\quad - \mathbf{E}[\varepsilon_{x_{k-1}}\sin(\varepsilon_{\theta_{k-1}})]\mathbf{E}\left[\Delta\hat{d}_k \left[\sin(\theta_{k-1}) + \frac{\Delta\hat{\theta}_k}{2}\cos(\theta_{k-1}) \right] \right]. \qquad (15)
\end{aligned}
$$

The issue now is to deal with nonlinear functions of stochastic, Gaussian in our case, variables [21]. Expanding $\varepsilon_{x_{k-1}}\cos(\varepsilon_{\theta_{k-1}})$ and $\varepsilon_{x_{k-1}}\sin(\varepsilon_{\theta_{k-1}})$ terms and using Taylor expansion,

$$
\begin{aligned}
\mathbf{E}[(\varepsilon_{x_{k-1}})\cos(\varepsilon_{\theta_{k-1}})] &= \mathbf{E}[(A\gamma_1 + B\gamma_2 + C\gamma_3) \\
&\quad \times (\cos(G\gamma_1 + H\gamma_2 + I\gamma_3)], \qquad (16) \\
\mathbf{E}[(\varepsilon_{x_{k-1}})\sin(\varepsilon_{\theta_{k-1}})] &= \mathbf{E}[(A\gamma_1 + B\gamma_2 + C\gamma_3) \\
&\quad \times (\sin(G\gamma_1 + H\gamma_2 + I\gamma_3)], \qquad (17) \\
\mathbf{E}[(\varepsilon_{x_{k-1}})\cos(\varepsilon_{\theta_{k-1}})] &= 0, \qquad (18) \\
\mathbf{E}[(\varepsilon_{x_{k-1}})\sin(\varepsilon_{\theta_{k-1}})] &= \mathbf{E}[(A\gamma_1 + B\gamma_2 + C\gamma_3) \\
&\quad \times (\sin(G\gamma_1)\cos(H\gamma_2)\cos(I\gamma_3) \\
&\quad - \sin(G\gamma_1)\sin(H\gamma_2)\sin(I\gamma_3) \\
&\quad + \cos(G\gamma_1)\sin(H\gamma_2)\cos(I\gamma_3) \\
&\quad + \cos(G\gamma_1)\cos(H\gamma_2)\cos(I\gamma_3)]
\end{aligned}
$$

$$= (AG\lambda_{1,1} + BH\lambda_{2,2} + CI\lambda_{3,3}) \times \exp\left(-\frac{G^2\lambda_{1,1}\sigma_{\dot{\gamma}_1}^2 + H^2\lambda_{2,2} + I^2\lambda_{3,3}}{2}\right)$$
$$= \text{cov}(x_{k-1}, \theta_{k-1}^T)\exp\left(-\frac{\sigma_{\theta_{k-1}}^2}{2}\right). \quad (19)$$

Finally, the expression for the first term in (13) is

$$\begin{aligned}\text{cov}(\varepsilon_{x_{k-1}}, \Delta x_k^T) &= -\mathbf{E}[\varepsilon_{x_{k-1}}\sin(\varepsilon_{\theta_{k-1}})] \times \mathbf{E}\left[\Delta\hat{d}_k\left[\sin(\theta_{k-1}) + \frac{\Delta\hat{\theta}_k}{2}\cos(\theta_{k-1})\right]\right] \\ &= -\text{cov}(x_{k-1}, \theta_{k-1}^T)\exp\left(-\frac{\sigma_{\theta_{k-1}}^2}{2}\right) \times \left(\Delta d_k \sin\theta_{k-1} + \frac{k_1}{2L}\cos\theta_{k-1}\tan\phi_k\right) \\ &= -\text{cov}(x_{k-1}, \theta_{k-1}^T)\mathbf{E}[\Delta\hat{\gamma}_k]. \quad (20)\end{aligned}$$

Although we do not provide here a full development for all the terms in (13), similar expressions can be found for them.

$$\mathbf{E}[\varepsilon_{x_{k-1}}\Delta\gamma_k^T] = \text{cov}(x_{k-1}, \theta_{k-1}^T)\mathbf{E}[\Delta\hat{x}_k], \quad (21)$$
$$\mathbf{E}[\varepsilon_{x_{k-1}}\Delta\theta_k^T] = 0, \quad (22)$$
$$\mathbf{E}[\varepsilon_{\gamma_{k-1}}\Delta x_k^T] = -\text{cov}(\gamma_{k-1}, \theta_{k-1}^T)\mathbf{E}[\Delta\hat{\gamma}_k], \quad (23)$$
$$\mathbf{E}[\varepsilon_{\gamma_{k-1}}\Delta\gamma_k^T] = \text{cov}(\gamma_{k-1}, \theta_{k-1}^T)\mathbf{E}[\Delta\hat{x}_k], \quad (24)$$
$$\mathbf{E}[\varepsilon_{\gamma_{k-1}}\Delta\theta_k^T] = 0, \quad (25)$$
$$\mathbf{E}[\varepsilon_{\theta_{k-1}}\Delta x_k^T] = -\text{cov}(\theta_{k-1}, \theta_{k-1}^T)\mathbf{E}[\Delta\hat{\gamma}_k], \quad (26)$$
$$\mathbf{E}[\varepsilon_{\theta_{k-1}}\Delta\gamma_k^T] = \text{cov}(\theta_{k-1}, \theta_{k-1}^T)\mathbf{E}[\Delta\hat{x}_k], \quad (27)$$
$$\mathbf{E}[\varepsilon_{\theta_{k-1}}\Delta\theta_k^T] = 0. \quad (28)$$

Approach Validation

To assess the validity and accuracy of the proposed covariance computation method, a series of simulated and real experiments were performed. Real experiments have been done using an AV with a nonholonomic Ackerman architecture. Figure 1 shows the standard vehicle for mining operations, which has been automated at the Center for Intelligent Systems (CSI) of the Tecnológico de Monterrey (ITESM). The vehicle is electrically powered, and steer and driven wheels were provided with independent sensors.

Both simulated and real experiments consisted of computing the total accumulated covariance of the vehicle's pose along a run path as well as the increment of this covariance obtained at each sample time.

Figure 1. *ITESM AV's mining vehicle.*

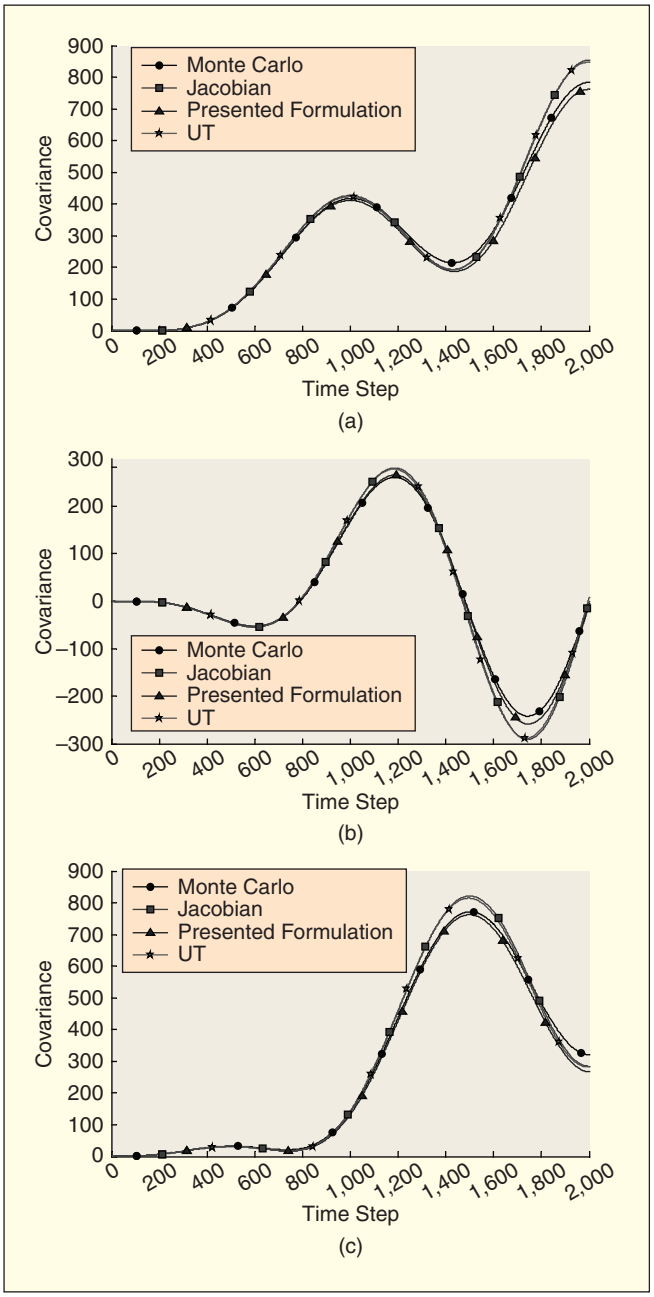

Figure 2. *Comparison for state (x,y,θ) covariance values of the circular trajectory between Monte Carlo simulation, Jacobian, the presented formulation, and UT. Values used in this run are* $\Delta d_k = 0.2$ m, $\sigma_{\Delta d_k} = 0.01$ m, $\phi_k = 1.125°$, $\sigma_{\phi_k} = 3°$, $L = 1.25$ m, *number of steps = 2,000. (a) cov(x,x). (b) cov(x,y). (c) cov(y,y). All covariance values are given in* m^2.

Three different paths have been performed. The first two are simulated paths: a circle with a total length of 400 m, and an *S*-type trajectory, in which the steering angle (φ) value changes sign at the half of the trajectory. The third path is a real outdoors experiment, with data (Δd_k and ϕ_k) being gathered from the ITESM-automated vehicle performing a rectangular 60 m $\times$ 40 m trajectory. In this third experiment, the vehicle was manually driven while following a marked path on the ground; so for this case, a ground truth was available. For all the experiments, we considered the following values for sensor variances: $\sigma_{\Delta d_k} = 0.01$ m and $\sigma_{\phi_k} = 3^\circ$. We use a high value for the steering variance to make evident the difficulty onto computing the orientation's uncertainty.

For each experiment, four different methods to obtain the pose uncertainty have been used. First, a Monte Carlo simulation for the path the vehicle performs has been made. Monte

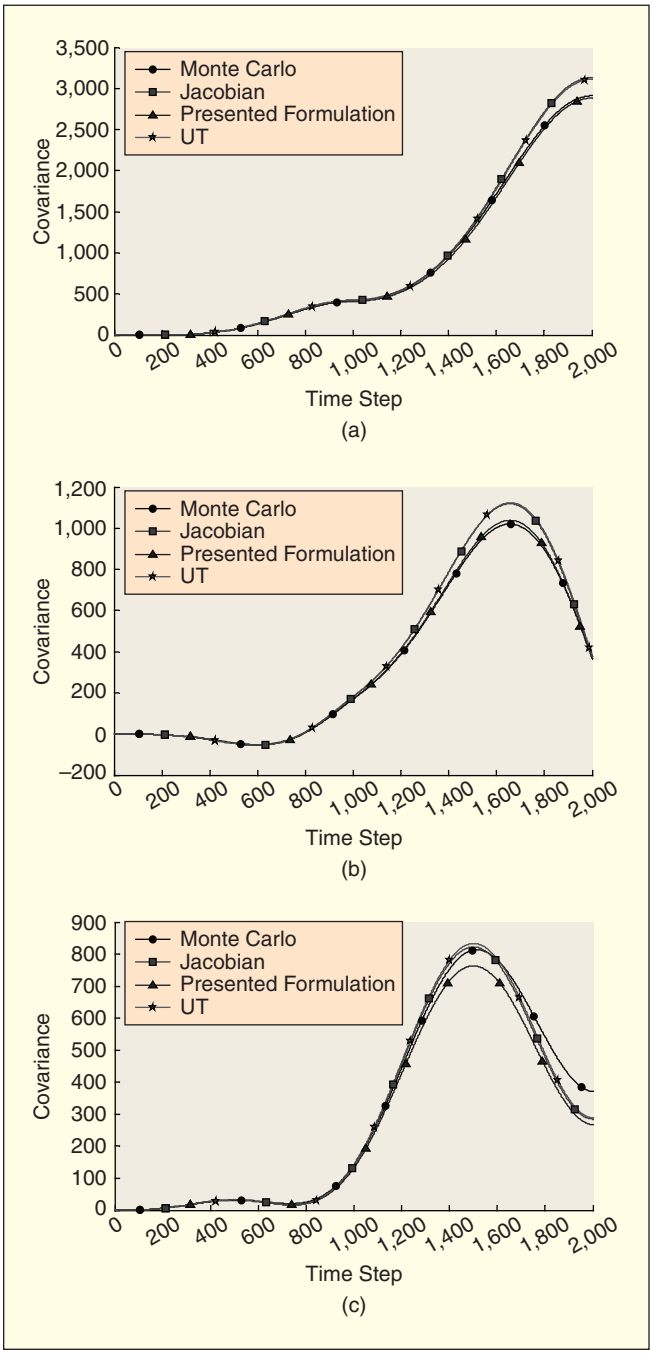

Figure 3. *Comparison for the state (x,y,θ) covariance values of the S trajectory (x,y,θ) between Monte Carlo simulation, Jacobian, the presented formulation, and UT. Values used in this run are L = 1.25 m, 2,000 steps, $\Delta d_k = 0.2$ m, $\sigma_{\Delta d_k} = 0.01$ m, $\phi_{1\ldots1,000} = 1.125^\circ$, $\phi_{1,001\ldots2,000} = -1.125^\circ$, $\sigma_{\phi_k} = 3^\circ$. (a) cov(x,x). (b) cov(x,y). (c) cov(y,y). All covariance values are given in m^2.*

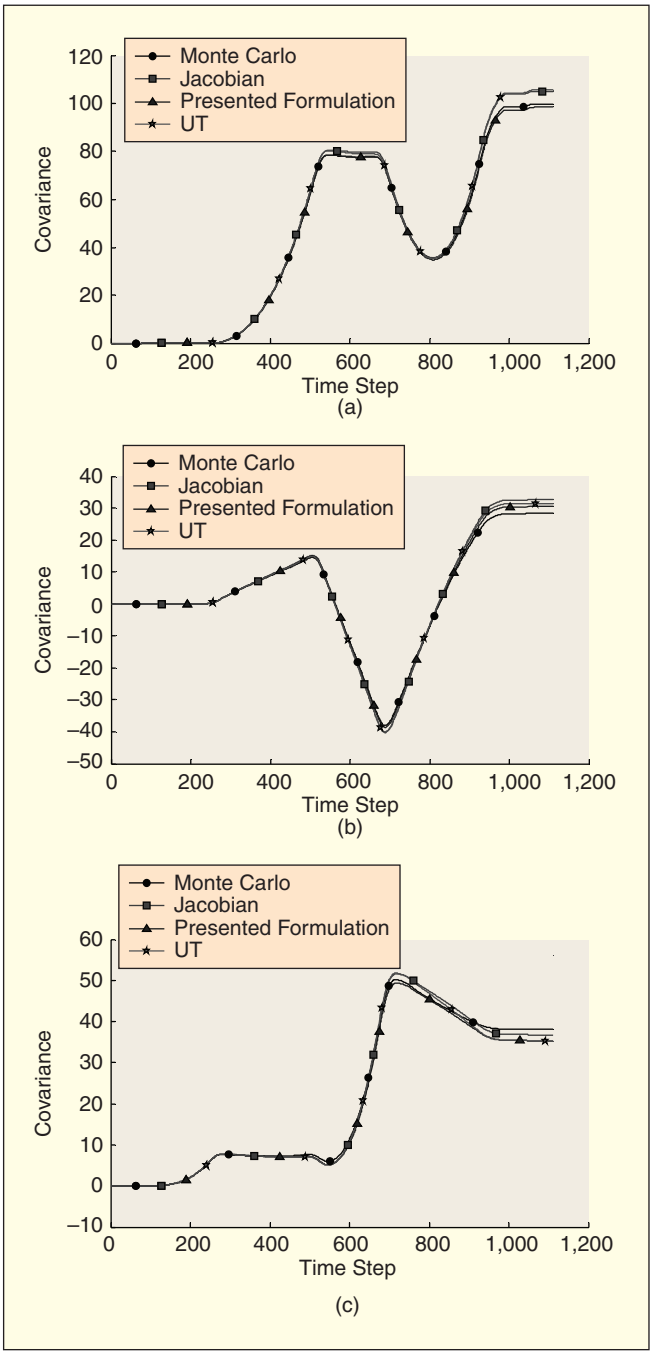

Figure 4. *Comparison for the state (x,y,θ) covariance values of the rectangular trajectory using real data between Monte Carlo simulation, Jacobian, the presented formulation, and UT. Values used in this run are $\sigma_{\phi_k} = 3^\circ$, L = 1.25 m. Δd_k, $\sigma_{\Delta d_k}$, and ϕ_k vary according the measured data. (a) cov(x,x). (b) cov(x,y). (c) cov(y,y). All covariance values are given in m^2.*

Table 1. Covariance differences for the circular trajectory (Figure 2) between Monte Carlo simulation and the other three approaches.

	Jacobian			Presented Formulae			UT		
k	*Cxx*	*Cxy*	*Cyy*	*Cxx*	*Cxy*	*Cyy*	*Cxx*	*Cxy*	*Cyy*
140	−0.01	0	0	−0.01	0	−0.01	0	0	0
280	−0.05	−0.11	0.12	−0.11	−0.02	0	−0.02	−0.12	0.1
420	0.38	−0.79	0.33	−0.24	−0.31	−0.05	0.51	−0.76	0.18
560	2.65	−1.69	−0.52	−0.03	−0.69	−0.92	2.87	−1.38	−0.85
700	6.95	−1.05	−2.77	0	−0.69	−2.91	6.9	−0.14	−3.04
840	10.24	3.3	−4.49	−2.06	0. 15	−5.67	9.06	4.78	−4.03
980	10.15	11.26	−1.33	−5.31	1.81	−8.28	6.97	12.38	0.44
1,120	3.08	18.65	10.53	−10.37	3.97	−9.6	−1.93	17.77	13.18
1,260	−9.59	17.34	28.76	−17.2	4.72	−10.02	−14.64	13.27	30.22
1,400	−20.03	1.78	46.33	−25.15	2	−8.49	−22.4	−4.68	43.71
1,540	−15.94	−25.03	47.53	−30.95	−5.71	−10.81	−14.04	−30.59	39.2
1,680	8.86	−46.76	24.08	−31.26	−14.51	−21.7	13.44	−47.32	11.8
1,820	45.11	−43.19	−12.15	−26.26	−16.6	−38.03	47.62	−37.05	−23.27
1,960	69.04	−9.57	−35.55	−21.97	−8.53	−52.11	64.52	0.25	−40.05

Carlo simulation was performed by averaging the covariances for each sample time in a total of 30,000 simulations to have a good estimation of the real value for the pose uncertainty. Foundations of this technique can be found in [22] and [23]. The other three methods shall be compared to this simulation.

The second method computes the uncertainty of the robot's pose using the classical way, i.e., computing the Jacobian of the vehicle's kinematic model [18]. The third method uses the scaled UKF [20], which is based on the UT [19] and particle filters, using specific and selected points to represent a given probability distribution. The fourth method computes the pose uncertainty using the proposed formulae in this article.

Figure 2 compares results for the different methods when the vehicle performs the 400 m circular trajectory. Figure 3 is for the S-type trajectory, and Figure 4 is for the real rectangular path.

More concretely, the figures show a comparison between Monte Carlo, Jacobian method, UT, and the presented formulation for the covariance of the pose at each sample time. In all the figures, the abscissa axis represents time steps, while the ordinate axis measures the pose uncertainty (units are specified under each graphics). Looking at the figures, it can be seen that, as long as the value of the accumulated uncertainty in the orientation of the vehicle, θ, is relatively small ($<10°$), the Jacobian-based method, the UKF, and the proposed formulae give similar results to those provided by the Monte Carlo simulation.

On the other hand, as the covariance in the orientation of the vehicle grows (as the case at hand, where the robot is trying to perform paths involving high changes in orientation using measures from internal noisy sensors), the computed uncertainty by either the Jacobian method, UKF method, or the one presented here differs from the Monte Carlo simulation results.

Table 2. Covariance differences for the S trajectory (Figure 3) between Monte Carlo simulation and the other three approaches.

	Jacobian			Presented Formulae			UT		
k	*Cxx*	*Cxy*	*Cyy*	*Cxx*	*Cxy*	*Cyy*	*Cxx*	*Cxy*	*Cyy*
140	0	−0.03	0.09	0	−0.03	0.08	0.01	−0.03	0.08
280	0.14	−0.36	0.52	0.08	−0.27	0.39	0.17	−0.37	0.49
420	1.34	−1.46	0.78	0.72	−0.98	0.41	1.47	−1.43	0.63
560	3.9	−2.53	−0.14	1.22	−1.54	−0.54	4.12	−2.22	−0.47
700	7.39	−1.84	−2.65	0.43	−1.48	−2.79	7.33	−0.93	−2.92
840	11.32	2.58	−4.67	−0.99	−0.57	−5.85	10.13	4.06	−4.21
980	10.53	10.06	−1.9	−4.93	0.62	−8.85	7.35	11.19	−0.12
1,120	5.25	20.11	7.26	−12.51	2.66	−12.87	−0.25	19.87	10.61
1,260	4.35	37.29	16.37	−23.89	5.77	−22.42	−4.29	36.14	22.28
1,400	15.47	62.41	18.28	−37.24	10.07	−36.59	2.98	61.81	27.34
1,540	51.63	88.44	3.7	−44.44	15.39	−54.72	35.52	90.26	15.1
1,680	113.17	97.88	−29.14	−41.82	17.92	−75.01	94.71	103.32	−17.56
1,820	179.71	71.64	−68.55	−33.58	10.79	−94.51	160.37	80.53	−59.29
1,960	218.77	9.32	−87.26	−27.94	−6.97	−103.84	199.34	20.53	−81.91

Table 3. Covariance differences for the real rectangular trajectory (Figure 4) between Monte Carlo simulation and the other three approaches.

	Jacobian			Presented Formulae			UT		
k	*Cxx*	*Cxy*	*Cyy*	*Cxx*	*Cxy*	*Cyy*	*Cxx*	*Cxy*	*Cyy*
75	0	0	0	0	0	0	0	0	0
150	0	0	0	0	0	0	0	0	0
225	−0.01	0	0.03	−0.01	0	0.01	−0.01	0	0.03
300	−0.01	0.05	0.08	−0.04	0.02	0.02	−0.01	0.05	0.05
375	0.15	0.21	0.01	−0.08	0.12	−0.04	0.15	0.16	−0.06
450	0.63	0.44	−0.24	−0.16	0.28	−0.28	0.63	0.28	−0.33
525	1.73	0.65	−0.71	−0.13	0.5	−0.74	1.7	0.3	−0.82
600	2.17	−0.57	−0.68	0.22	0.1	−1.01	1.68	−0.86	−0.55
675	1.94	−2.08	0.46	0.03	−0.52	−1.26	1	−2.06	0.82
750	0.3	−1.2	1.73	−0.48	−0.48	−0.51	−0.31	−0.83	1.61
825	0.23	0.66	1.67	−0.76	0.17	−0.31	0.13	0.88	0.94
900	2.18	2.67	0.46	−1.07	1.13	−1.23	2.47	2.26	−0.74
975	5.07	4.14	−1.2	−1.33	2.02	−2.68	5.55	2.97	−2.66
1,050	5.3	4.23	−1.33	−1.34	2.08	−2.79	5.79	3.01	−2.8

Kinematic Model of the AV

We provide in this section an overview of the method proposed in [16] and [17]. The formulation is particularized for an AV for which a bicycle kinematic model is considered (see Figure A1) [24]. The vehicle's pose can be fully determined from $\hat{P}_k = [\hat{x}_k \quad \hat{y}_k \quad \hat{\theta}_k]^T$, where

$$\begin{bmatrix} \hat{P}_k \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \hat{x}_{k-1} \\ 0 & 1 & 0 & \hat{y}_{k-1} \\ 0 & 0 & 1 & \hat{\theta}_{k-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\hat{x}_k \\ \Delta\hat{y}_k \\ \Delta\hat{\theta}_k \\ 1 \end{bmatrix} = \begin{bmatrix} \hat{x}_{k-1} + \Delta\hat{x}_k \\ \hat{y}_{k-1} + \Delta\hat{y}_k \\ \hat{\theta}_{k-1} + \Delta\hat{\theta}_k \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} \hat{x}_{k-1} + \Delta\hat{d}_k \left[\cos(\hat{\theta}_{k-1}) - \frac{\Delta\hat{\theta}_k}{2}\sin(\hat{\theta}_{k-1})\right] \\ \hat{y}_{k-1} + \Delta\hat{d}_k \left[\sin(\hat{\theta}_{k-1}) + \frac{\Delta\hat{\theta}_k}{2}\cos(\hat{\theta}_{k-1})\right] \\ \hat{\theta}_{k-1} + \frac{\Delta\hat{d}_k \tan\hat{\phi}_k}{L} \\ 1 \end{bmatrix}. \quad \text{(A1)}$$

Variables Δd_k, ϕ_k, are the measurements the automated vehicle senses. Δd_k is the measured displacement, and ϕ_k is the measured steering angle. It is worth to mention that $\Delta\hat{\theta}_k$ is not obtained from the rear wheel's displacement; hence, independency can be assumed between $\Delta\hat{d}_k$ and $\Delta\hat{\theta}_k$. Independent errors with Gaussian probability distributions are added to these measurements:

$$\Delta\hat{d}_k = \Delta d_k + \varepsilon_{\Delta d_k}, \quad \text{(A2)}$$

$$\hat{\phi}_k = \phi_k + \varepsilon_{\phi_k}. \quad \text{(A3)}$$

In order that (A2) and (A3) perform correctly, the systematic errors of the vehicle need to be corrected and compensated [25]. Now, suppose that, for time $k-1$, the pose of the vehicle and its associated uncertainty are known $\{\hat{P}_{k-1} = [\hat{x}_{k-1} \quad \hat{y}_{k-1} \quad \hat{\theta}_{k-1}]^T, \text{cov}(\hat{P}_{k-1})\}$. We are using $k-1$ for $t-\Delta t$, $k-2$ for $t-2\Delta t$, and so on. Then, for time t, after the vehicle has performed a certain movement on the plane and sensors on the robot have noisily measured this displacement, the pose of the vehicle is obtained using $\hat{P}_k = \hat{P}_{k-1} + \Delta\hat{P}_k$, $\Delta\hat{P}_k = [\Delta\hat{x}_k \quad \Delta\hat{y}_k \quad \Delta\hat{\theta}k]^T$.

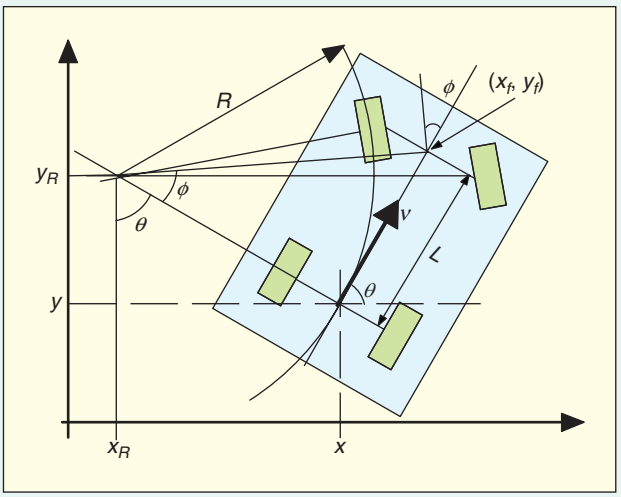

Figure A1. *Kinematic model for a common car-like vehicle.*

Uncertainty of the pose estimate is then given by the covariance matrix $\text{cov}(\hat{P}_k)$, which can be further decomposed into the following expression:

$$\text{cov}(\hat{P}_k) = \text{cov}(\hat{P}_{k-1}) + \text{cov}(\Delta\hat{P}_k) + \text{cov}(\hat{P}_{k-1}, \Delta\hat{P}_k^T) + \text{cov}(\Delta\hat{P}_k, \hat{P}_{k-1}^T). \quad \text{(A4)}$$

The term $\text{cov}(\hat{P}_{t-1})$ is recursively defined. The first term to obtain is $\text{cov}(\Delta\hat{P}_k)$:

$$\text{cov}(\Delta\hat{P}_k) = \mathsf{E}\left[\begin{bmatrix} \Delta\hat{x}_k \\ \Delta\hat{y}_k \\ \Delta\hat{\theta}_k \end{bmatrix}\begin{bmatrix} \Delta\hat{x}_k \\ \Delta\hat{y}_k \\ \Delta\hat{\theta}_k \end{bmatrix}^T\right] - \mathsf{E}\begin{bmatrix} \Delta\hat{x}_k \\ \Delta\hat{y}_k \\ \Delta\hat{\theta}_k \end{bmatrix}\left[\mathsf{E}\begin{bmatrix} \Delta\hat{x}_k \\ \Delta\hat{y}_k \\ \Delta\hat{\theta}_k \end{bmatrix}\right]^T, \quad \text{(A5)}$$

where E stands for the expected value of the corresponding function.

As can be noticed, the presented formulation is less affected by this error than the Jacobian and UKF approaches, respectively; thus, the estimation of the vehicle's pose uncertainty, when computed with the presented formulae, is more accurate.

To summarize the information provided in the given figures, we present in Tables 1–3 the comparison between the Monte Carlo simulation and the other three methods for the run experiments. The tables show the results of subtracting the Monte Carlo covariance values to the ones obtained by the other approaches. It is worth to mention that, at certain time steps, UT and Jacobian behave better for some covariance value, i.e., $\mathrm{cov}(x, x)$, but worse in all the other. In general, the presented formulae give the lowest error.

Finally, to obtain a better evaluation of these methods, the covariance eigenvalues obtained with the presented formulation, Jacobian, and UT are compared to those obtained with the Monte Carlo simulation (always equal to 0). From (11), λ is a diagonal matrix containing the eigenvalues of $\mathrm{cov}(\hat{P}_k)$:

$$\lambda = \left[\begin{array}{ccc} ev_{1,1} & 0 & 0 \\ 0 & ev_{2,2} & 0 \\ 0 & 0 & ev_{3,3} \end{array} \right] \Bigg|_{ev_{1,1} > ev_{2,2} > ev_{3,3}} . \tag{29}$$

Eigenvalues are compared by using

$$\mathrm{diff}^{*} = \sum_{i=1}^{3} \left| ev_{i,i}^{\mathrm{MC}} - ev_{i,i}^{*} \right|, \tag{30}$$

where (*) could be Jacobian, UT, or the developed formulation, and diff represents how close the different methods are to the covariance obtained with Monte Carlo approach. Results are shown in Figure 5, in which the developed formulation is closer than the other two techniques.

Conclusions

The localization problem for a mobile platform involves two different issues: computing a pose estimate and computing its associated uncertainty. This article has focused on how the robot's pose uncertainty can be accurately estimated and how this uncertainty is propagated to future states. This is important since, in any outdoor mobile robot application, we need to know as accurate as possible the error in the robot's pose estimation.

In contrary to other approaches, we have considered here that the cross-covariance between the previous position and the actual increment of position is not zero. This consideration leads to accurate pose uncertainty estimation when propagating it to future states.

So our main contribution is a novel solution for the calculation of such cross-covariance terms. In this way, we are able to catch the highly nonlinear behavior of the pose uncertainty while accurately estimating it.

Experiments presented in the "Approach Validation" section show that the computed pose uncertainty estimation with the proposed formulation captures the motion phenomena with a higher degree of accuracy than that obtained with traditional methods such as the Jacobian computation and the scaled UT. In this way, the presented method is less sensible to

Autonomous vehicles and mobile robots are developed to carry out specific tasks autonomously.

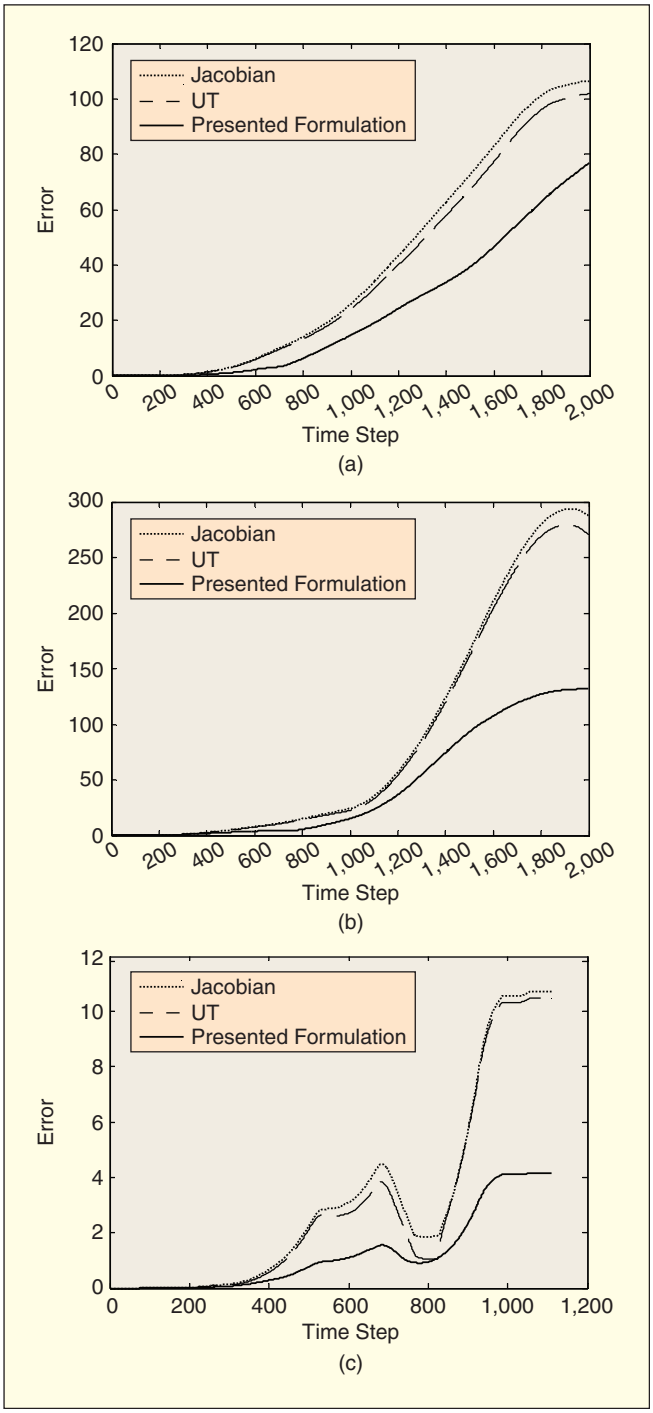

Figure 5. *Eigenvector comparison of Monte Carlo approach against Jacobian, UT, and the presented formulation approaches for the three trajectories performed in the experiments: (a) circular, (b) S, and (c) rectangular real trajectories.*

the accumulated error in the robot orientation, which turns out to be the most important error when using internal sensors to obtain a pose estimate.

Although the presented method has been particularized to an Ackerman vehicle platform, it is general and valid for all types of robotic vehicles.

Acknowledgment

The authors thank Gerardo González and Fernando Rivero for their support in this project. This work has been partially supported by Consejo Nacional de Ciencia y Tecnología (CONACyT) under grant 35396 and by the ITESM Special Research Program on AVs (CAT-049).

Keywords

Uncertainty position estimation, robot localization, robot navigation.

References

[1] A. Ollero, *Robótica: Manipuladores Y robots Móviles*. Marcombo, Spain: Aníbal Ollero Baturone, 2001.

[2] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[3] J. Gutiérrez, "Configuration and construction of an autonomous vehicle for tunnel profiling tasks," Ph.D. dissertation, ITESM, Campus Monterrey, 2004.

[4] F. J. Rodríguez, M. Mazo, and M. A. Sotelo, "Automation of a industrial fork lift truck, guided by artificial vision in open environments," *Autonom. Robot.*, vol. 5, no. 2, pp. 215–231, May 1998.

[5] S. M. Stevenson, "Mars pathfinder Rover-Lewis Research Center technology experiments program," in *Proc. Energy Conversion Engineering Conference (IECEC)*, July–Aug. 1997, vol. 1, no. 27, pp. 722–772.

[6] G. Hizinger and B. Brunner, "Space robotics—DLRs telerobotic concepts, lightweight arms and articulated hands," *Autonom. Robot.*, vol. 14, no. 2, pp. 127–145, May 2003.

[7] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning—Sensors and techniques," *J. Robot. Syst. (Special Issue on Mobile Robots)*, vol. 14, no. 4, pp. 231–249, Apr. 1997.

[8] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[9] A. C. Murtra, J. M. Mirats Tur, and A. Sanfeliu, "Active global localization for mobile robots in cooperative and large environments," *Robot. Autonom. Syst.*, vol. 56, no. 10, pp. 807–818, Oct. 2008.

[10] R. A Brooks, "Visual map making for a mobile robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1985, pp. 824–829.

[11] R. Chatila and J. P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 1985, pp. 138–145.

[12] C. M Wang, "Location estimation and uncertainty analysis for mobile robots," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1988, pp. 1230–1235.

[13] K. S Chong and L. Kleeman, "Accurate odometry and error modeling for a mobile robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 1997, pp. 2783–2788.

[14] A. Martinelli, "Evaluating the odometry error of a mobile robot," in *Proc. Int. Conf. Intell. Robot and Systems (IROS02)*, Lausanne, Switzerland, Sept. 30–Oct. 4 2002, pp. 853–858.

[15] A. Kelly, "Linearized error propagation in odometry," *Int. J. Robot. Res.*, vol. 23, no. 2, pp. 179–218, Feb. 2004.

[16] J. M. Mirats-Tur, J. L. Gordillo, and C. Albores, "A closed form expression for the uncertainty in odometry position estimate of an autonomous vehicle," *IEEE Trans. Robot. Automat.*, vol. 21, no. 5, pp. 1017–1022, Oct. 2005.

[17] J. M. Mirats-Tur, C. Albores, and J. L. Gordillo, "Erratum to: A closed form expression for the uncertainty in odometry position estimate of an autonomous vehicle," *IEEE Trans. Robot. Automat.*, vol. 23, no. 6, p. 1302, Dec. 2007.

[18] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Robot. Res.*, vol. 5, no. 4, pp. 56–68, 1986.

[19] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *Proc. American Control Conf.*, June 1995, vol. 3, pp. 1628–1632.

[20] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. Symp. Adaptive Systems for Signal Processing, Communications, and Control*, 2000, pp. 153–158.

[21] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 3rd ed. New York: McGraw-Hill, 1991.

[22] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York: Springer-Verlag, 2004.

[23] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*. Hoboken, NJ: Wiley, 2007.

[24] F. González-Palacios, "Control de Dirección de un Vehículo Autónomo," Master's thesis, ITESM, Campus Monterrey, 2005.

[25] N. L Doh and W. K. Chung, "A systematic representation method of the odometry uncertainty of mobile robots," *Intell. Automat. Soft Comput.*, vol. 11, no. 10, pp. 1–13, 2005.

Carlos Albores Borja graduated as an electronic systems engineer in 1997 from ITESM in Mexico. He also received his M.Sc. (with honors) and Ph.D. degrees at ITESM in 2001 and 2007, respectively. Currently, he works full time at the Research and Development Department, IDZ, in Monterrey. His areas of interest include mobile robotics in outdoor environments and sensor data fusion.

Josep M. Mirats Tur graduated as an electronic engineer in 1995 at the Universitat Politècnica de Catalunya (UPC). In 2000, he obtained his Ph.D. degree at the Institute of Robotics in Barcelona. In 2002, he was awarded with the mention of European Doctor. He has been the invited professor at the ITESM in Mexico (2004), the University of Cambridge (2008), and the University of California in San Diego (UCSD) in 2008 and 2009. Currently, he is a full-time researcher at the Institute of Robotics in Barcelona. His main areas of interest include mobile cooperative robotics in outdoor environments, knowledge integration, sensor data fusion, and tensegrity-based robots.

José Luis Gordillo graduated in industrial engineering from the Technological Institute of Aguascalientes, Mexico. He obtained both his D.E.A. and Ph.D. degrees in computer science from the National Polytechnic Institute of Grenoble, France, in 1983 and 1988, respectively. Currently, he is full professor at the Center for Intelligent Computing and Robotics at the Tecnológico de Monterrey. He has been a visiting professor in the Computer Science Robotics Laboratory at Stanford University (1993), at INRIA Rhone-Alpes in France (2002 and 2004), and at LAAS-CNRS in Toulouse, France (2007–2008). His research interests include computer vision for robotics applications, in particular, autonomous vehicles, and the development of virtual laboratories for education and manufacturing.

Address for Correspondence: Josep M. Mirats Tur, Institut de Robòtica i Informàtica Industrial (UPC-CSIC), Parc Tecnològic de Barcelona, Edifici U, C. Llorens i Artigas, 4-6, 2a Planta, 08028 Barcelona, Spain. E-mail: jmirats@iri.upc.edu.