Proceedings of the 2007 American Control Conference
Marriott Marquis Hotel at Times Square
New York City, USA, July 11-13, 2007

Integrated Control and Navigation for Omni-directional Mobile Robot Based on Trajectory Linearization

Yong Liu, Robert L. Williams II and J. Jim Zhu

Abstract: In this paper, an integrated navigation and control for omni-directional mobile robot is developed. Both control and navigation algorithms are based on trajectory linearization. The robot control is based on trajectory linearization control (TLC), in which an open-loop kinematic inversion and a closed-loop linear time varying (LTV) stabilizer are combined together to provide robust and accurate trajectory tracking performance. The LTV stabilizer is designed along the nominal trajectory provided by the kinematic inversion. The robot navigation is based on a sensor fusion using nonlinear Kalman filter which is also designed along the nominal trajectory. The sensor fusion combines onboard sensor and vision system measurements together, and provides reliable and accurate location and orientation measurements. Gating technology is employed to remove the inaccurate vision measurement. A real-time hardware-in-the-loop (HIL) simulation system was built to verified the proposed integrated control and navigation. Test results show that the proposed method improves robot location and orientation measurements reliability and accuracy, thus it improves the robot controller performance significantly.

I. INTRODUCTION AND PROBLEM STATEMENT

An omni-directional mobile robot is a holonomic robot [1][2]. The inherent agility of the omni-directional mobile robot makes it widely studied for dynamic environment applications. The annual international Robocup competition in which teams of autonomous robots compete in soccer-like games, is an example where the omni-directional mobile robot can be used. The Ohio University (OU) Robocup Team's entry *Robocat* is a cross-disciplinary research project intended for Robocup small-size league competition. The current OU Robocup team members are comprised of Phase V omni-directional mobile robots, as shown in Fig. 1. The Phase V *Robocat* has three omni-directional wheels, arranged 120° apart. Each wheel is driven by a DC motor installed with an optical shaft encoder. An overhead camera above the field of play can sense the position and the orientation of robots.

In Robocup games, a precise trajectory tracking control for the robot is one of the key areas to improve a team's performance. A nonlinear controller based on trajectory linearization control (TLC) was developed for *Robocat* robots [3][4]. TLC combines nonlinear dynamic inversion and linear time-varying eigenstructure assignment, and provides robust stability and performance along the trajectory without interpolation of controller gains [5]. TLC has been successfully applied to missile and reusable launch vehicle flight control systems [7-10]. Both hardware-in-the-loop (HIL) simulation and the real-time competition performance demonstrate that the TLC controller significantly improves the robot maneuverability. With the same set of controller parameters, the robot is able to follow various three-degree-of-freedom (3DOF) trajectories accurately.

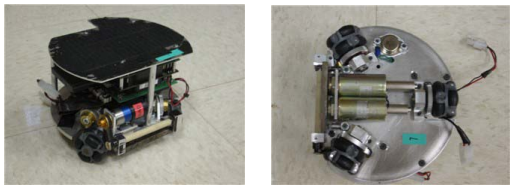


Figure 1 Phase V Robocat Robot

It was observed that the accurate position and orientation measurement is the throttle to further improve the system performance. In the present *Robocat* system configuration, the robot location and orientation can be measured from either onboard sensors or a roof camera. Onboard sensors, including motor shaft encoder, accelerometer and gyroscope, measure the robot body rate, including robot speed and rotation angular rate. The robot location and orientation is estimated by integrating the measured robot body rate. Such estimation has the advantage of high sampling rate. Whereas it has unbounded cumulative errors introduced the body rate measure noise or wheel slippage [12]. If using the onboard sensor alone, the robot drifts away from trajectory commands. The vision system using the roof camera can measure the robot location and orientation directly by image processing. However, the vision system is slow and unreliable due to the roof camera's slow capture rate, the random image processing failure and the delay of the wireless communication between the off-field vision system and the robot. If using merely vision system, the delay and failures of image processing can destabilize robot controller.

In this paper, an integrated control and navigation technique for robot position and orientation control is developed. The developed method augments the existing controller with a navigation component using sensor fusion. The sensor fusion combines vision system and onboard sensor estimation together to provide an accurate and reliable location measurement. It is based on a nonlinear

Manuscript received September 15th, 2007.

Yong Liu (email: yongliu@bobcat.ent.ohiou.edu) and J. Jim Zhu (corresponding author, email: zhuj@ohio.edu) are with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH, 45701.

Robert L. Williams II is with Department of Mechanical Engineering, Ohio University, Athens, Ohio, 45701(email: williar4@ohio.edu).

Kalman filter algorithm designed along the nominal trajectory provided by the controller's dynamic inversion. A gating technique is employed to remove the incorrect vision data.

Kalman filter is a widely used method in sensor fusion [13]. The standard Kalman filter is developed for linear dynamic system with Gaussian noise distribution. For nonlinear systems, techniques such as linearized Kalman filter, extended Kalman filter, unscented Kalman filter and Particle filter have been developed and applied successfully in many practical applications[14][15]. Linearized Kalman filter and extended Kalman filter (EKF) apply standard Kalman filter by linearizing the original nonlinear system [14]. In a linearzed Kalman filter, linearization is along a nominal trajectory. The dynamic system state may diverge from the nominal trajectory over time. Thus the linearized Kalman filter is usually used in short time mission. In EKF, the linearization is about the state estimation. Thus there is a danger that error propagation and filter divergence may occur. Both linearized Kalman filter and EKF are computationally efficient. However, terms neglected in linearization may lead to suboptimal performance. To overcome such disadvantage, unscented Kalman filter (UKF) [15] and particle filters (PFs) [16] are developed. UKF and PFs require much larger computational power to implement, compared to linearized Kalman filter and EKF.

The proposed nonlinear filter is motivated by TLC observer design [11]. Its structure is similar to linearized Kalman filter. In this structure, the nominal system trajectory generated by TLC controller is used to linearize the nonlinear robot kinematics in the filter. The TLC controller drives the robot to follow the nominal trajectory. Thus the divergence problem in the linearized Kalman filter is alleviated. The proposed nonlinear Kalman filter is computationally efficient, and is suitable for real-time implementation on the robot onboard computer.

A real-time HIL simulation system was developed to verify the proposed method. Real-time test results show that the sensor fusion method provides reliable and accurate location and orientation measurements. Thus it improves the robot control system performance.

In section II, the controller structure is briefly reviewed. Then the sensor fusion algorithm for navigation is illustrated. In section III, the real-time HIL system is described. In section IV, the real-time test results are presented.

II. OMNI-DIRECTIONAL MOBILE ROBOT CONTROL AND NAVIGATION BASED ON TLC

In this section, the kinematics model and TLC controller design of omni-directional robot is first reviewed, then the sensor fusion based on nonlinear Kalman filter is described.

A. Robocat Kinematics and TLC Controller

In the *Robocat* mobile robot controller design, a two-loop controller architecture is employed, as shown in Figure 2. The outer-loop controller adjusts the robot position and orientation following command trajectories. The inner-loop is a body rate controller which follows the body rate commands from the outer-loop controller. Both outer-loop and inner-lope controllers employ TLC structure. Detailed design and test result of the omni-directional mobile robot TLC controller is summarized in [3][4]. In this section, only kinematics and outer-loop controller design is briefly reviewed.

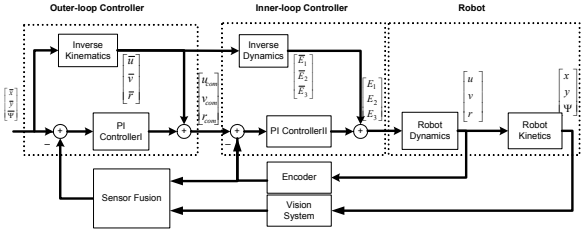
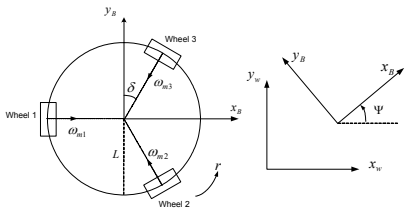


Figure 2 Robot TLC Controller Structure

There are two coordinate frames used in the modeling: the body frame {B} and the world frame {W}. The body frame is fixed on the moving robot with the origin at the center of chassis, as shown in the Figure 3(a). The world frame is fixed on the field of play, as shown in Figure 3(b).



(a) Body frame$\{B\}$ (b) World frame$\{W\}$

Figure 3 Coordinate Frames

Symbols used in the robot dynamic model are listed in Table 1.

Table 1. Symbols

Body Frame	
r	Rotation angular rate
$u,\ v$	Velocity components
$\overline{u}, \overline{v}, \overline{r}$	Nominal body rate
$u_{\text{com}}, v_{\text{com}}, r_{\text{com}}$	Body rate command
$\omega_{m1}, \omega_{m2}, \omega_{m3}$	Motor Shaft Speed
World Frame	
(x, y)	Robot location
Ψ	Robot orientation angle
Mechanical Constants	
R	Wheel radius
L	Radius of robot body
n	Gear ratio
δ	Wheel orientation angle

The Robot kinematics is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} cos(\Psi) & -sin(\Psi) & 0 \\ sin(\Psi) & cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \tag{1}$$

The relationship between body rate $[u\, v\, r]^T$ and motor shaft speed $[\omega_{m1}, \omega_{m2}, \omega_{m3}]^T$ is given by

$$\begin{bmatrix} u \\ v \\ r \end{bmatrix} = \left(\begin{bmatrix} 1 & cos(\delta) & -cos(\delta) \\ -1 & sin(\delta) & sin(\delta) \\ L & L & L \end{bmatrix}^T \right)^{-1} \frac{R}{n} \begin{bmatrix} \omega_{m1} \\ \omega_{m2} \\ \omega_{m3} \end{bmatrix} \tag{2}$$

The outer-loop controller design is based on Eq. (1). First, from (1), the nominal body rate for a desired trajectory $[\overline{x}(t), \overline{y}(t), \overline{\Psi}(t)]^{\mathrm{T}}$ is

$$\begin{bmatrix} \overline{u} \\ \overline{v} \\ \overline{r} \end{bmatrix} = \begin{bmatrix} cos(\overline{\Psi}) & sin(\overline{\Psi}) & 0 \\ -sin(\overline{\Psi}) & cos(\overline{\Psi}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\overline{x}}(t) \\ \dot{\overline{y}}(t) \\ \dot{\overline{\Psi}}(t) \end{bmatrix} \tag{3}$$

where $[\dot{\overline{x}}(t), \dot{\overline{y}}(t), \dot{\overline{\Psi}}(t)]^{\mathrm{T}}$ is calculated from the command $[\overline{x}(t), \overline{y}(t), \overline{\Psi}(t)]^{\mathrm{T}}$ using a pseudo-differentiator. A second order pseudo-differentiator is represented by the following transfer function

$$G_{\text{diff}}(s) = \frac{\omega_{n,\text{diff}}^2 \, s}{s^2 + 2\zeta\omega_{n,\text{diff}} s + \omega_{n,\text{diff}}^2} \tag{4}$$

where ζ is the damping ratio; $\omega_{n,\text{diff}}$ is the low-pass filter bandwidth that attenuates high frequency gain, thereby making the pseudo-differentiator causal and realizable. In the controller realization, the nominal trajectory is replaced by filtered position command taken from the pseudo-differentiator.

Define the robot position tracking error and the tracking error control by

$$[e_x \; e_y \; e_\Psi]^T = [\, x \quad y \quad \Psi \,]^T - [\, \overline{x} \quad \overline{y} \quad \overline{\Psi} \,]^T,$$
$$[\delta u \; \delta v \; \delta r]^T = [\, u_{\text{com}} \quad v_{\text{com}} \quad r_{\text{com}} \,]^T - [\, \overline{u} \quad \overline{v} \quad \overline{r} \,]^T$$

Linearizing (1) along the nominal trajectories $[\overline{x}(t), \overline{y}(t), \overline{\Psi}(t)]^{\mathrm{T}}$ and $[\overline{u}(t), \overline{v}(t), \overline{r}(t)]^{\mathrm{T}}$ yields the linearized error dynamics

$$[\dot{e}_x \dot{e}_y \dot{e}_\Psi]^T = A_1 [e_x \, e_y \, e_\Psi]^T + B_1 [\delta u \, \delta v \, \delta r]^T \tag{5}$$

where

$$A_1 = \begin{bmatrix} 0 & 0 & -sin(\overline{\Psi})\overline{u} - cos(\overline{\Psi})\overline{v} \\ 0 & 0 & cos(\overline{\Psi})\overline{u} - sin(\overline{\Psi})\overline{v} \\ 0 & 0 & 0 \end{bmatrix} B_1 = \begin{bmatrix} cos(\overline{\Psi}) & -sin(\overline{\Psi}) & 0 \\ sin(\overline{\Psi}) & cos(\overline{\Psi}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Secondly, a proportional-integral (PI) feedback control law is designed to stabilize the tracking error.

$$\begin{bmatrix} \delta u \\ \delta v \\ \delta r \end{bmatrix} = -K_{p1} \begin{bmatrix} e_x \\ e_y \\ e_\Psi \end{bmatrix} - K_{I1} \begin{bmatrix} \int e_x(t)dt \\ \int e_y(t)dt \\ \int e_\Psi(t)dt \end{bmatrix} \tag{6}$$

where K_{p1} and K_{I1} are time-varying matrix gains. The body rate command to the inner-loop is given by

$$[u_{\text{com}} \, v_{\text{com}} \, r_{\text{com}}]^T = [\overline{u} \; \overline{v} \; \overline{r}]^T + [\delta u \; \delta v \; \delta r]^T \tag{7}$$

B. Nonlinear Kalman Filter for Omni-directional Robot Sensor Fusion

(1) Location and Orientation Measurement Model

Approximate the robot kinematics (1) using forward Eular method with time interval T

$$\begin{bmatrix} x_k \\ y_k \\ \Psi_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \Psi_{k-1} \end{bmatrix} + \begin{bmatrix} cos(\Psi_{k-1}) \cdot T & -sin(\Psi_{k-1}) \cdot T & 0 \\ sin(\Psi_{k-1}) \cdot T & cos(\Psi_{k-1}) \cdot T & 0 \\ 0 & 0 & 1 \cdot T \end{bmatrix} \begin{bmatrix} u_{k-1} \\ v_{k-1} \\ r_{k-1} \end{bmatrix}$$

In omni-directional mobile robot, body rate can be measured from onboard sensor. Position can be observed from the vision system. The body rate measurement $[\widehat{u}_k \; \widehat{v}_k \; \widehat{r}_k]^T$ at time step k is defined as

$$[\widehat{u}_k \; \widehat{v}_k \; \widehat{r}_k]^T = [u_k \; v_k \; r_k]^T + [w_{1,k}, w_{2,k}, w_{3,k}]^T \tag{8}$$

where $[w_{1,k}, w_{2,k}, w_{3,k}]^T$ is body rate measurement noises. The vision system measurement $z_{1,k}, z_{2,k}$ and $z_{3,k}$ at time step k can be defined as

$$\begin{bmatrix} z_{1,k} \\ z_{2,k} \\ z_{3,k} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \Psi_k \end{bmatrix} + \begin{bmatrix} d_{1,k} \\ d_{2,k} \\ d_{3,k} \end{bmatrix} \tag{9}$$

where $[d_{1,k} \, d_{2,k} \, d_{3,k}]^T$ is vision system noise.

Both $[w_{1,k}, w_{2,k}, w_{3,k}]^T$ and $[d_{1,k} \, d_{2,k} \, d_{3,k}]^T$ are assumed to be white with normal distribution, such that

$$p(w_{1,k-1}, w_{2,k-1}, w_{3,k-1}) \sim N(0, Q)$$
$$p(d_{1,k} \, d_{2,k} \, d_{3,k}) \sim N(0, R)$$

where $Q \in \mathbb{R}^{3\times3}$ is the body rate measurement covariance, and $R \in \mathbb{R}^{3\times3}$ is the vision system observation noise covariance.

(2) Nonlinear Kalman Filter Using Trajectory Linearization

Define $[x_k^- \; y_k^- \; \Psi_k^-]^T$ as the *priori* location estimation from body rate measurement at time step k. It can be calculated as

$$\begin{bmatrix} x_k^- \\ y_k^- \\ \Psi_k^- \end{bmatrix} = \begin{bmatrix} \widehat{x}_{k-1} \\ \widehat{y}_{k-1} \\ \widehat{\Psi}_{k-1} \end{bmatrix} + \begin{bmatrix} cos(\widehat{\Psi}_{k-1}) \cdot T & -sin(\widehat{\Psi}_{k-1}) \cdot T & 0 \\ sin(\widehat{\Psi}_{k-1}) \cdot T & cos(\widehat{\Psi}_{k-1}) \cdot T & 0 \\ 0 & 0 & 1 \cdot T \end{bmatrix} \begin{bmatrix} \widehat{u}_{k-1} \\ \widehat{v}_{k-1} \\ \widehat{r}_{k-1} \end{bmatrix} \tag{10}$$

Then the predicted vision system observation can be defined as

$$[\, z_{1,k}^- \quad z_{2,k}^- \quad z_{3,k}^- \,]^T = [\, x_k^- \quad y_k^- \quad \Psi_k^- \,]^T \tag{11}$$

Define prediction error and innovation error as

$$\begin{bmatrix} e_{x_k} \\ e_{y_k} \\ e_{\Psi_k} \end{bmatrix} = \begin{bmatrix} x_k^- \\ y_k^- \\ \Psi_k^- \end{bmatrix} - \begin{bmatrix} x_k \\ y_k \\ \Psi_k \end{bmatrix}, \begin{bmatrix} e_{z_{1,k}} \\ e_{z_{2,k}} \\ e_{z_{3,k}} \end{bmatrix} = \begin{bmatrix} z_{1,k}^- \\ z_{2,k}^- \\ z_{3,k}^- \end{bmatrix} - \begin{bmatrix} z_{1,k} \\ z_{2,k} \\ z_{3,k} \end{bmatrix} \tag{12}$$

By linearizing equation (10) and along real position $[x_k \, y_k \, \Psi_k]^T$, the prediction error dynamics can be approximated as

$$\begin{bmatrix} e_{x_k} \\ e_{y_k} \\ e_{\Psi_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sin(\Psi_{k-1}) \cdot u_{k-1} \cdot T - cos(\Psi_{k-1}) \cdot v_{k-1} \cdot T \\ 0 & 1 & cos(\Psi_{k-1}) \cdot u_{k-1} \cdot T - sin(\Psi_{k-1}) \cdot v_{k-1} \cdot T \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \widehat{x}_{k-1} \\ \widehat{y}_{k-1} \\ \widehat{\Psi}_{k-1} \end{bmatrix} - \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \Psi_{k-1} \end{bmatrix} \right) + \begin{bmatrix} cos(\Psi_{k-1}) \cdot T & -sin(\Psi_{k-1}) \cdot T & 0 \\ sin(\Psi_{k-1}) \cdot T & cos(\Psi_{k-1}) \cdot T & 0 \\ 0 & 0 & 1 \cdot T \end{bmatrix} \cdot \begin{bmatrix} w_{1,k-1} \\ w_{2,k-1} \\ w_{3,k-1} \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} e_{z_{1,k}} \\ e_{z_{2,k}} \\ e_{z_{3,k}} \end{bmatrix} = \begin{bmatrix} e_{x_k} \\ e_{y_k} \\ e_{\Psi_k} \end{bmatrix} + \begin{bmatrix} d_{1,k} \\ d_{2,k} \\ d_{3,k} \end{bmatrix} \tag{14}$$

In Equation (13), $\left[\widehat{x}_{k-1} \, \widehat{y}_{k-1} \widehat{\Psi}_{k-1}\right] - [x_{k\text{-}1} \, y_{k\text{-}1} \, \Psi_{k\text{-}1}]^T$ is an estimate of $[e_{x_{k\text{-}1}} \, e_{y_{k\text{-}1}} \, e_{\Psi_{k\text{-}1}}]^T$. In (13), the real position $[x_{k-1} \, y_{k-1} \, \Psi_{k-1}]^T$ and the real body rate $[u_{k-1} \, v_{k-1} \, r_{k-1}]^T$ are unknown. In TLC controller, the robot trajectory is driven close to nominal trajectory. Thus the nominal position $[\overline{x}_{k-1} \, \overline{y}_{k-1} \, \overline{\Psi}_{k-1}]^T$ and nominal body rate $[\overline{u}_{k-1} \, \overline{v}_{k-1} \, \overline{r}_{k-1}]^T$ can be used to approximate actual state in (13). Equation (13) and (14) can be rewritten as

$$\begin{bmatrix} e_{x_k} \\ e_{y_k} \\ e_{\Psi_k} \end{bmatrix} = A_k \left(\begin{bmatrix} \widehat{x}_{k-1} \\ \widehat{y}_{k-1} \\ \widehat{\Psi}_{k-1} \end{bmatrix} - \begin{bmatrix} x_{k\text{-}1} \\ y_{k\text{-}1} \\ \Psi_{k\text{-}1} \end{bmatrix} \right) + W_k \begin{bmatrix} w_{1,k-1} \\ w_{2,k-1} \\ w_{3,k-1} \end{bmatrix} \tag{15}$$

where $A_k = \begin{bmatrix} 1 & 0 & -sin(\overline{\Psi}_{k-1}) \cdot \overline{u}_{k-1} \cdot T - cos(\overline{\Psi}_{k-1}) \cdot \overline{v}_{k-1} \cdot T \\ 0 & 1 & cos(\overline{\Psi}_{k-1}) \cdot \overline{u}_{k-1} \cdot T - sin(\overline{\Psi}_{k-1}) \cdot \overline{v}_{k-1} \cdot T \\ 0 & 0 & 1 \end{bmatrix}$

$$W_k = \begin{bmatrix} cos(\overline{\Psi}_{k-1}) \cdot T & -sin(\overline{\Psi}_{k-1}) \cdot T & 0 \\ sin(\overline{\Psi}_{k-1}) \cdot T & cos(\overline{\Psi}_{k-1}) \cdot T & 0 \\ 0 & 0 & 1 \cdot T \end{bmatrix}$$

Equation (15) and (14) are linearized error dynamics along the nominal trajectory. Assuming A_k and W_k are slow-varying, a nonlinear Kalman filter can be constructed based on (15) and (14). It is similar to linearized Kalman filter, except that the TLC controller ensure the actual robot trajectory close to the nominal trajectory. It is different from extended Kalman filter (EKF), in which the estimation at the preceding time step is used to linearize the error dynamics. The nonlinear Kalman filter design is described in Equation(17) to (22)

(3) Gating and Integration with Vision System

In practice, the vision system may lose frames or give wrong measurements due to image processing failure. Gating is a technique for eliminating most unlikely measurement [14]. There are several commonly used gating algorithms. Rectangular gate is the simplest one. Rectangular Gating is defined as the following

$$|e_{z_{1,k}}| \le 3\sqrt{\sigma^2_{R(i)} + \sigma^2_{P(i)}}, i = 1,2,3 \qquad (16)$$

where $\sigma^2_{R(i)}$ is the diagonal element of the vision system noise covariance R, and $\sigma^2_{P_k(i)}$ is the appropriate diagonal element of the prediction covariance P_k^-. If all innovation residues satisfy the above gating condition, then the vision system is considered as valid, and will be used in filter correction. Otherwise, vision system data is determined as invalid.

If there is no valid vision system data, such as when the video frames are lost or vision data is rejected by gating, the correction step of Kalman will not be executed. If the vision system data is delayed, a fast than real-time Kalman filter can be executed using recorded Kalman filter history data once the vision system data is received. The overall nonlinear Kalman filter based on trajectory linearization for mobile robot location sensor fusion is summarized below.

Step 1: Read onboard sensor body rate measurement and estimate robot position and orientation

$$\begin{bmatrix} x_k^- \\ y_k^- \\ \Psi_k^- \end{bmatrix} = \begin{bmatrix} \widehat{x}_{k-1} \\ \widehat{y}_{k-1} \\ \widehat{\Psi}_{k-1} \end{bmatrix} + \begin{bmatrix} cos(\widehat{\Psi}_{k-1}) \cdot T & -sin(\widehat{\Psi}_{k-1}) \cdot T & 0 \\ sin(\widehat{\Psi}_{k-1}) \cdot T & cos(\widehat{\Psi}_{k-1}) \cdot T & 0 \\ 0 & 0 & 1 \cdot T \end{bmatrix} \begin{bmatrix} \widehat{u}_{k-1} \\ \widehat{v}_{k-1} \\ \widehat{r}_{k-1} \end{bmatrix} \qquad (17)$$

$$P_k^- = A_k \cdot P_{k-1} \cdot A_k^T + W_k \cdot Q_{k-1} \cdot W_k^T$$

$$[\, z_{1,k}^- \quad z_{2,k}^- \quad z_{3,k}^- \,]^T = [\, x_k^- \quad y_k^- \quad \Psi_k^- \,]^T \qquad (18)$$

Step 2: Read vision system measurement.

If the vision system data is not available, go to Step 4; If the vision system data is available, calculate innovation residue by (12). If all innovation residues satisfy the gating criterion (16), then the vision system data is valid. Go to Step 3; otherwise, go to Step 4.

Step 3: Correction with valid vision data

$$K_k = P_k^-(P_k^- + R)^{-1} \qquad (19)$$
$$P_k = (I - K_k)P_k^-$$

where R_k is the output measurement noise covariance. The *posteriori* estimation is

$$\begin{bmatrix} \widehat{x}_k \\ \widehat{y}_k \\ \widehat{\Psi}_k \end{bmatrix} = \begin{bmatrix} x_k^- \\ y_k^- \\ \Psi_k^- \end{bmatrix} + K_k \left(\begin{bmatrix} z_{1,k} \\ z_{2,k} \\ z_{3,k} \end{bmatrix} - \begin{bmatrix} x_k^- \\ y_k^- \\ \Psi_k^- \end{bmatrix} \right) \qquad (20)$$

Goto Step 1.

Step 4: Calculate the prediction covariance without correction.

$$[\, \widehat{x}_k \quad \widehat{y}_k \quad \widehat{\Psi}_k \,]^T = [\, x_k^- \quad y_k^- \quad \Psi_k^- \,]^T \qquad (21)$$
$$P_k = P_k^- \qquad (22)$$

Goto Step 1.

III. HARDWARE-IN-THE-LOOP (HIL) SIMULATION

The integrated control and navigation for omni-directional mobile robot was verified and tested in a real-time HIL simulation. The HIL simulation system is shown in Figure 4.

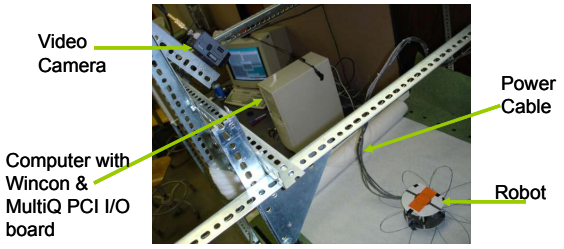


Figure 4 HIL Simulation System

In the HIL simulation, a Phase V robot was used. Quanser's Wincon©, Mathworks' Simulink© and Real-time Workshop© executed on a PC were used to develop a fast prototype of the real-time TLC controller and the sensor fusion system. Motors on the model mobile robot were driven by Quanser's Multi-Q PCI I/O board and power amplifiers. Motor shaft encoder signals were fed to Multi-Q PCI I/O board to measure motor shaft speeds. A Cognachrome 2000 vision system with a YC-100 CCD camera was used to measure robot location. Congnachrome 2000© system identifies robot position and orientation, and transfers these data to PC via a serial port. Vision system data was then calibrated to the world frame using second order polynomials.

The HIL simulation system has the most important features in the real *Robocat* system that hinder the robot controller performance. The robot position and orientation estimate from the motor shaft speeds has cumulative errors when slippage occurs. The power cable connecting the robot generates large drag forces and causes robot wheel slip. It occasionally blocks the camera, which results in loss of image frame. The asynchronous RS-232 communication has randomly mismatched data frame, which results in incorrect vision data.

IV. HIL SIMULATION RESULT

In the HIL simulation, two groups of tests were conducted and compared: one used only the onboard encoder data; whereas the other was augmented with sensor fusion method. In the test, the sampling time interval $T = 0.02$ (second). Over 20 real-time tests were conducted. In all these tests, sensor fusion improved robot tracking performance. In this section, several test results are presented. To emphasis the significant improvement using sensor fusion, a pen attached on the robot was used to plot the actual trajectory.

A. Square Trajectory with Rotation

In this test, the robot was commanded to follow a square trajectory at the speed of 0.2m/s on each side of the square, while rotating 45° during side motion with a fixed rotation rate. The command trajectory is of three degree of freedom (3DOF). There are significant wheel slippage at each corner due to robot acceleration. Real-time test results are illustrated in Figure 5. Figure 5(a) is the controller tracking performance. The controller can accurately follow the command if it is given the correct measurement. Figure 5 (b) shows the sensor fusion result. It can be seen that the encoder estimation diverged slowly, while the vision system had many corrupted data. The sensor fusion used the vision system to calibrate the encoder estimation and discarded the invalid vision system data. Figure 5(c) shows the gating decision. In Figure 5(c) 1 means accept the vision data, 0 means reject the vision data. Figure 5(d) shows the robot trajectory in $x - y$ plan drawn by recorded data. It can be seen from Figure 5(d) that at each corner, the encoder estimation has a large orientation error. The robot cable drag also introduced encoder estimation error, such as in regions near point A and point B. The encoder estimation error accumulates over time. The sensor fusion method corrects the encoder error when the vision system data is available.

B. Square Trajectory with Fixed Orientation

In this test, the command is the same to test 1 except the robot orientation is fixed. The actual robot trajectory was plotted on the field of play by an attached pen. The test results are shown in Figure 6. Figure 6 (b) and (c) are photos of actual robot trajectories drawn by an attached pen.

Experiment using vision system alone was also conducted. The disturbance induced by vision system failure destabilized the robot very soon.

C. Circular Trajectory and Rose Curves

In these tests, the first command is to accelerate from the initial position, and draw a circle of 0.25 m radius at an angular rate of 1 rad/s. The second command is to draw a rose curve, which is a petalled flower curve generated by $r = a\sin(n\theta)$, where r and θ are the radius and the rotation angle in polar coordinate, and n is an integer determining the number of petals. The robot orientation was fixed. Test results are shown in Fig. 7 and 8. As the result of the shift, the trajectory plotted when using encoder only is much lighter than the one using sensor fusion.

IV CONCLUSION AND FUTURE WORK

In this paper, an integrated control and navigation method for omni-directional mobile robot is proposed, and real-time HIL test results are presented. The method augmented the robot controller with a navigation component using sensor fusion, which combines onboard sensor and vision system measurement together. It employs a nonlinear Kalman filter technique based on trajectory linearization. Gating technology is employed to remove the incorrect vision data. Real-time HIL simulation test results show that the proposed method is able to improve robot location and orientation measurement reliability and accuracy, thus it improves the robot controller performance significantly.

The proposed sensor fusion method is an experimental example of the nonlinear Kalman filter based on trajectory linearization. In the future, theoretical study for the general case of nonlinear Kalman filter based on trajectory linearization will be explored. The integrated control and navigation method will be implemented and tested in the real Robocat system.

Reference

[1] F. G. Pin, and S. M. Killough, “A new family of omnidirectional and holonomic wheeled platforms for mobile robots,” IEEE Trans. Robotics Automat. 10(2) (1994), pp480-489.

[2] M.-J. Jung, H.-S. Kim, S. Kim, and J.-H. Kim, “Omni-Directional Mobile Base OK-II”, Proceedings of the IEEE International Conference on Robotics and Automation, 4: 3449-3454, 2000.

[3] Y. Liu, X. Wu, J. J. Zhu and J. Lew, “Omni-Directional Mobile Robot Controller Design by Trajectory Linearization", Proceedings of the American Control Conference. v 4, 2003, p 3423-3428

[4] Y. Liu, J. J. Zhu, R. L. Williams II, and J. Wu, “Omni-Directional Mobile Robot Controller Based on Trajectory Linearization”, submitted to Robotics and Autonomous Systems.

[5] M. C. Mickle, R. Huang, and J. J. Zhu, “Unstable, Nonminimum Phase, Nonlinear Tracking by Trajectory Linearization Control," Proceedings, pp. 812-818, 2004 IEEE Conference on Control Applications, Taipei, Taiwan, Sept. 2004.

[6] J. J. Zhu, B. D. Banker, C. E. Hall, “X-33 Ascent Flight Controller Design by Trajectory Linearization- A Singular Perturbational Approach," 2000 AIAA Guidance, Navigation, and Control Conference, Denver, CO, Aug. 2000.

[7] M. C. Mickle, and J. J. Zhu, “Bank-To-Turn Roll-Yaw-Pitch Autopilot Design Using Dynamic Nonlinear Inversion and PD-eigenvalue assignment”, Proc., 2000 American Control Conference, Chicago, IL, to appear, June 2000

[8] J. J. Zhu, A. S. Hodel, K. Funston, K. and C. E. Hall, “X-33 entry flight controller design by trajectory linearization - a singular perturbational approach,” AAS-01-012, American Astronautical Society Guidance and Control Conference, Breckenridge, Colorado, Jan. 2001.

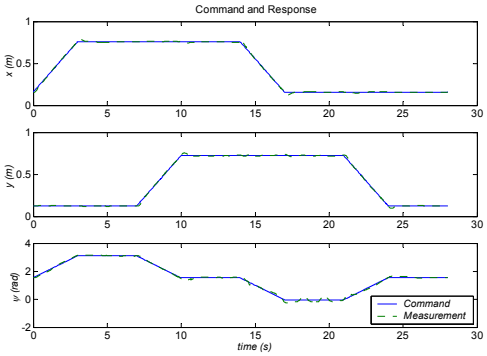
[9] X. Wu, Y. Liu, and J. J. Zhu, “Design and Real-Time Testing of a Trajectory Linearization Flight Controller for the 'Quanser UFO'”, Proceedings of the American Control Conference. v 4, 2003, p 3931-3938.

[10] J. J. Zhu, and A. B. Huizenga, “A type two trajectory linearization controller for a reusable launch vehicle - A singular perturbation approach", Collection of Technical Papers - AIAA Atmospheric Flight Mechanics Conference. v. 2 (2004) p. 1121-1137.

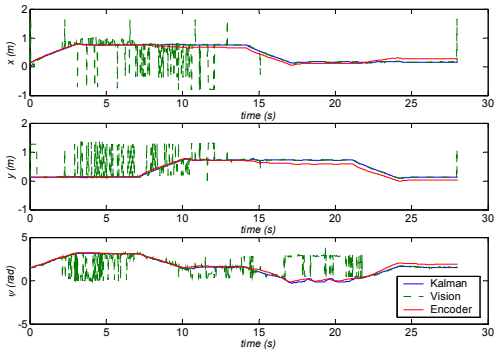
[11] R. Huang, M. C. Mickle, and J. J. Zhu, “Nonlinear Time-varying Observer Design Using Trajectory Linearization”, Proceedings of the American Control Conference. v 6, 2003, p 4772-4778.

[12] R. L. Williams II, B. E. Carter, P. Gallina, P. and G. Rosati, “Dynamic Model with Slip for Wheeled Omni-Directional Robots”, IEEE Transactions on Robotics and Automation, 18(3): 285-293. 2002.

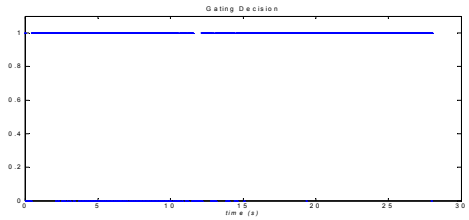
[13] D. L. Hall, and S. A. H. McMullen, Mathematical Techniques in MultiSensor Data Fusion, Artech House, Boston, 2nd ed, 2004.
[14] R. G. Brown, and P. Y. C. Hwang, *Introduction to random signals and applied Kalman filtering : with MATLAB exercises and solutions*, 3rd ed, Wiley, New York.
[15] J. Richard and N. D. Sigpurwalla, "Understanding the Kalman Filter", The American Statistician, May 1983, Volume 37, N0. 2
[16] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking", IEEE Transactions of Signal Processing, Vol. 50(2), pages 174-188.



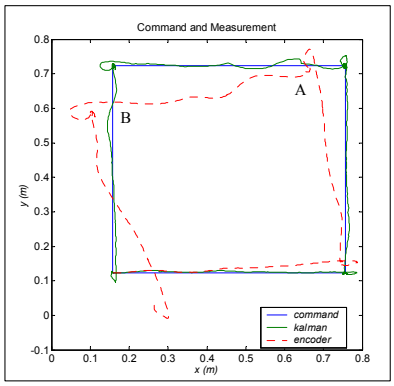
(a) Controller Performance Using Sensor Fusion



(b) Kalman Filter Performance

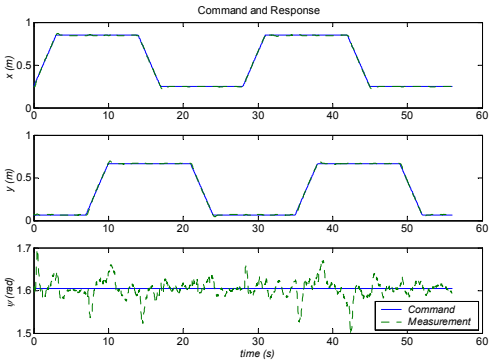


(c) Gating Decision

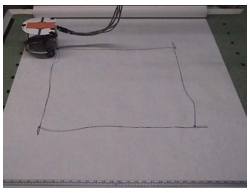


(d) Robot Trajectory Data

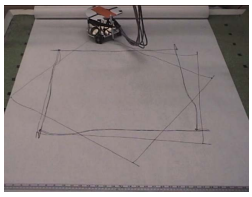
Figure 5 Square Trajectory with Rotation



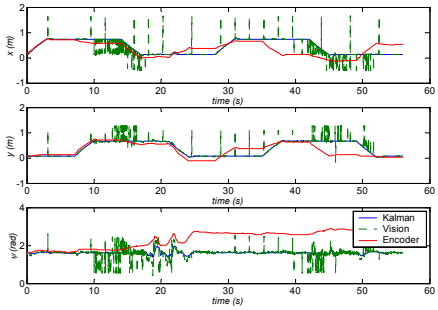
(a) Tracking Performance



(b) Trajectory Using Sensor Fusion

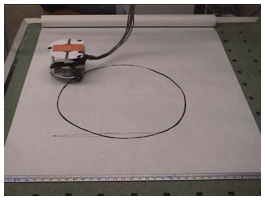


(c) Trajectory Using Encoder

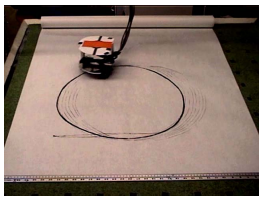


(d) Kalman Filter Performance

Figure 6 Square Trajectory with Fixed Orientation

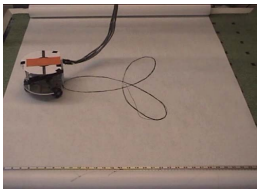


(a) Using Sensor Fusion

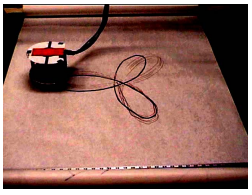


(b)Using Encoder Alone

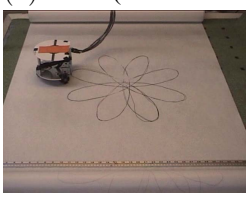
Figure 7 Circular Trajectory



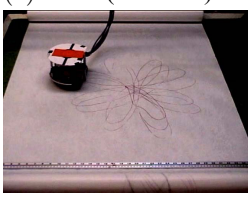
$(a)\ n = 3$ (Sensor Fusion)



$(b)\ n = 3$(Encoder)



$(c)\ n = 4$ (Sensor Fusion)



$(d)\ n = 4$ (Encoder)

Figure 8 Actual Robot Rose Curve Trajectories