J Electr Eng Technol Vol. 8, No. 3: 613-620, 2013
http://dx.doi.org/10.5370/JEET.2013.8.3.613

ISSN(Print) 1975-0102
ISSN(Online) 2093-7423

Mobile Robot Destination Generation by Tracking a Remote Controller Using a Vision-aided Inertial Navigation Algorithm

Dang Quoc Khanh* and Young-Soo Suh†

Abstract – A new remote control algorithm for a mobile robot is proposed, where a remote controller consists of a camera and inertial sensors. Initially the relative position and orientation of a robot is estimated by capturing four circle landmarks on the plate of the robot. When the remote controller moves to point to the destination, the camera pointing trajectory is estimated using an inertial navigation algorithm. The destination is transmitted wirelessly to the robot and then the robot is controlled to move to the destination. A quick movement of the remote controller is possible since the destination is estimated using inertial sensors. Also unlike the vision only control, the robot can be out of camera's range of view.

Keywords: Remote control, Inertial navigation algorithm, Position and orientation estimation, Mobile robot.

1. Introduction

For remote control of a mobile robot, normally a joystick is often used to give commands to the robot. It is, however, difficult to control sometimes: for example, to move a robot in an environment with many obstacles with a joystick is not easy. For easier control, a force-feedback joystick can be used [1, 2].

In this paper, we propose a more intuitive remote control method, where a robot moves to the destination the remote controller pointing to. A camera is installed on the remote controller and circle landmarks are installed on the mobile robot. Similar approaches have been proposed in [3, 4]. In these papers, G. Novak *et al.* introduce a small soccer robot which plays in a playground supervised by a camera mounted above. The image of the camera is processed by a host computer located beside the playground and the result of processing is a trajectory for the soccer robot. The movement of the soccer robot is estimated and controlled using the camera. The soccer robot is always within the camera's range of view.

The combination of inertial sensors and vision has been received a lot of attentions recently [5-7]. In [5], the accurate position information of a quad-copter is estimated using an off-board stereo camera setup consisting of two web cameras and an on-board IMU (Inertial Measurement Unit). A marker-based video tracking system and an IMU are used to obtain an absolute position [6]. For another application, in [7], an IMU is incorporated into the object and a robot manipulator with a stereo camera was tasked to track the moving object with the help of the object's inertial information. In [8, 9], inertial sensors are combined with vision information to estimate the motion.

In this paper, we combine inertial sensors with vision information. Four circle landmarks are installed on a mobile robot and the relative position and orientation of the remote controller are computed using pose estimation algorithms [10, 11]. The movement of the remote controller is estimated using an inertial navigation algorithm [12]. A preliminary result is reported in [13]. In [13], it is assumed that the position of the remote controller is motionless. Only the orientation of the remote controller can be changed and only gyroscopes are used to track the orientation change. In this paper, this restriction is solved and an inertial navigation algorithm is used to track the remote controller.

The paper is organized as follows. Section 2 introduces the problem formulation. A destination computation method and a robot control algorithm are explained in Section 3 and Section 4, respectively. Error analysis for the proposed method is given in Section 5. Section 6 is for verifying the advantages of proposed system through some experiments and some conclusion is given in Section 7.

2. Problem Formulation

2.1 System hardware

The proposed system (Fig. 1) consists of a mobile robot and a remote controller. The mobile robot is a two DC motor driven wheel vehicle and four circle landmarks are attached on the top of the robot. The mobile robot receives destination commands from the remote controller through wireless communication.

† Corresponding Author: School of Electrical Engineering, University of Ulsan, Namgu, Ulsan Korea. (yssuh@ulsan.ac.kr)
* School of Electrical Engineering, University of Ulsan, Namgu, Ulsan Korea. (khanhdq86@yahoo.com)

Revised: December 27, 2011; Accepted: November 27, 2012

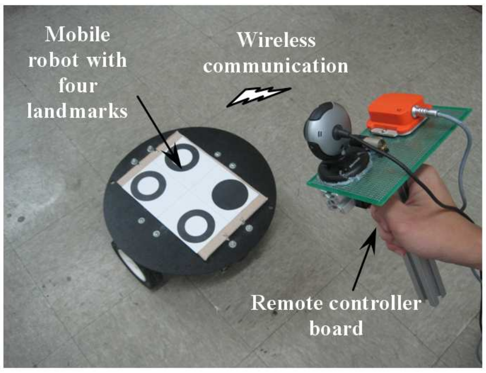

Fig. 1. System overview.

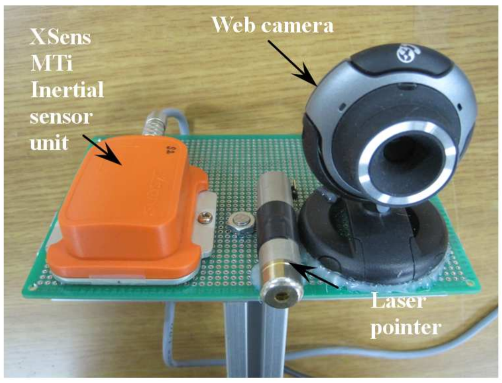

Fig. 2. Remote controller board.

The remote controller (Fig. 2) consists of a web camera (640 × 480 pixel resolution) and an inertial sensor unit (XSens MTi), where the three axes of the camera coincide with those of the inertial sensor unit. A laser pointer is also attached on the remote controller. We note that the pointer is not used for controlling the mobile robot. It is used to visually check where the remote controller is pointing to.

2.2 Overview of the proposed remote control algorithm

An overview of the proposed remote control algorithm is illustrated in Fig. 3. A basic idea is that a robot is remotely controlled so that it moves to a destination where the remote controller is pointing to.

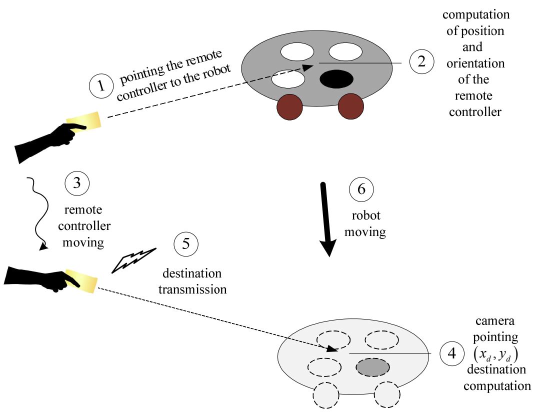
1 pointing the remote controller to the robot
2 computation of position and orientation of the remote controller
3 remote controller moving
6 robot moving
5 destination transmission
4 camera pointing (x_d, y_d) destination computation

Fig. 3. Overview of the remote control algorithm

If we point the remote controller to the mobile robot ①, relative position and orientation of the remote controller are computed using landmarks on the robot ②. If we move the remote controller and stop ③, the destination (where the remote controller is pointing to) is computed using an inertial navigation algorithm ④. The computed destination is transmitted to the mobile robot ⑤ and the robot moves toward the destination ⑥.

3 Destination Computation

In this section, the destination computing method is explained in detail.

The destination computation algorithm can be divided into mainly four parts: a zero velocity detection algorithm, a position and orientation computation algorithm using landmarks, a remote controller movement computation algorithm using inertial sensors, and a camera pointing destination computation algorithm.

3.1 Zero velocity detection algorithm

The remote control process can be divided into three different time intervals (Fig. 4). The first interval is when the remote controller is almost not moving and pointing to the robot. The second interval is when the remote controller is moving to point to the destination. The third interval is when the remote controller stops and is pointing to the destination. We call the first and third intervals zero velocity intervals since the remote controller is not moving during the intervals. Thus the remote control process begins with detecting a zero velocity interval.

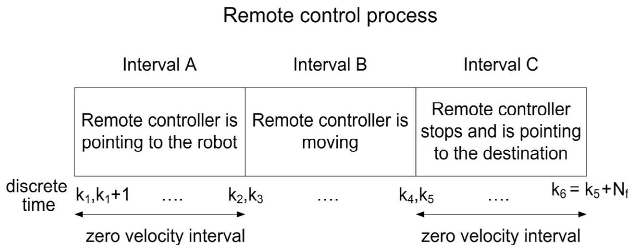
Remote control process
Interval A
Interval B
Interval C
Remote controller is pointing to the robot
Remote controller is moving
Remote controller stops and is pointing to the destination
discrete time
k_1, k_1+1 …. k_2, k_3 …. k_4, k_5 …. $k_6 = k_5+N_f$
zero velocity interval
zero velocity interval

Fig. 4. Remote control process is divided into three intervals

Since it is difficult to directly measure velocity of the remote controller, we assume that the remote controller is not moving if gyroscope values and changes of accelerometers are small for a specified time. Specifically, the discrete time k is assumed to belong to a zero velocity interval if the following are satisfied:

$$\begin{cases} \|y_g[i]\| \le \delta_g & k-N_g \le i \le k+N_g \\ \|y_a[i+1]-y_a[i]\| \le \delta_a & k-N_a \le i \le k+N_a \end{cases}$$

where y_g and y_a are gyroscope output and accelerometer output, respectively. δ_g, δ_a, N_g and N_a are parameters for the zero velocity interval detection.

3.2 Position and orientation computation algorithm

When the remote controller is pointing to the robot (Interval A in Fig. 4), relative position and orientation of the remote controller with respect to the robot are computed.

Four coordinate frames are used in this paper (Fig. 5): body coordinate frame, robot coordinate frame, world coordinate frame and navigation coordinate frame. Recall that the camera and the inertial sensor unit have the same coordinate system. Three axes of the body coordinate coincide with those of the camera (or the inertial sensor unit) in the remote controller. The origin of the body coordinate frame is the same as that of the camera coordinate frame. The robot coordinate frame is fixed on the robot, where the *x*-*y* plane is parallel to the plate of the robot. Its origin is at the center of the plate.

The world coordinate frame is defined as the robot coordinate frame at the start of the remote control process (that is, at the start of Interval A in Fig. 4). The defined world coordinate frame is only valid during the same remote control process. For different remote control processes, different world coordinate frames are chosen.

The navigation coordinate frame is a local level frame with its *z* axis pointing up. In this paper, we assume that the navigation coordinate frame and the world coordinate frame are the same. That is, the plate of the robot is assumed to be completely level. This assumption simplifies derivation of the destination computation.

Let $r \in \Re^3$ be a position of the remote controller in the world coordinate frame at the time *k*. Let $C_b^w \in \Re^{3\times 3}$ be a rotation matrix between the world coordinate frame and the body coordinate frame. Suppose p_w is a position of the landmark on the robot (center of the circle) and its image coordinates of the camera is $[u,v]'$. Then, in the perspective projection, the following equation is satisfied [14]:

$$p_w = C_b^w s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} + r \qquad (1)$$

where *s* is the scaling factor.

It is known [10, 11] that if there are at least four landmarks (that is, there are four (1) equations), we can compute C_b^w and *r*. We used the method in [10] to compute C_b^w and *r* since it is fast and robust to noises. Recalling that the navigation coordinate frame and the world coordinate frame are assumed to be the same, we can also compute pitch and roll information of C_b^w from accelerometers using the following relationship:

$$y_a = \left(C_b^w\right)' \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \qquad (2)$$

where *g* is the gravitation acceleration.

Ideally pitch and roll values computed using (1) and (2) should be the same. They are, however, not the same in practice due to sensor noises and the fact that the plate of the robot is not completely level. In the paper, pitch and roll are computed from (2) and the remaining yaw is computed using (1). In Fig. 4, note that k_2 is the final time of interval A. Thus in this section, $r[k_2]$ and $C_b^w[k_2]$ are computed.

3.3 Remote controller movement computation algorithm

Once we compute the initial position ($r[k_2]$) and orientation ($C_b^w[k_2]$) of the remote controller, the next step is to compute the final position ($r[k_6]$) and orientation ($C_b^w[k_6]$) using inertial navigation algorithms [12]. If the remote controller stops moving for more than N_f discrete times (that is, if the length of interval C in Fig. 4 is larger than N_f), $k_6 = k_5 + N_f$ (see Fig. 4) becomes the final time. Among various inertial navigation algorithms, we used the initial navigation algorithm in [15], where a 15 states complementary Kalman filter is used. In the Kalman filter, we used the fact that the remote controller is not moving during interval C (see Fig. 4) to reduce errors in the final position and orientation.

3.4 Camera pointing destination computation algorithm

After the final position ($r[k_6]$) and orientation

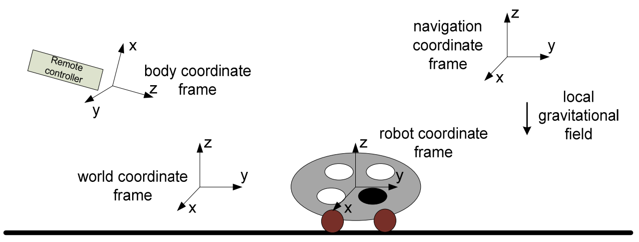

Fig. 5. Four Coordinates used in this paper.

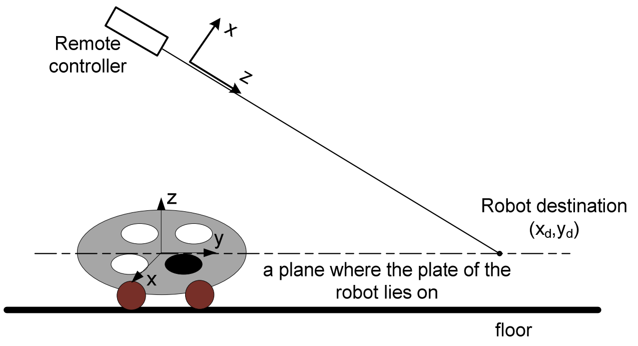

Fig. 6. Robot destination computation.

($C_b^w[k_6]$) are computed, the robot destination (x_d, y_d) is computed. The robot destination is the intersection (see Fig. 6) of the *z* axis of the body coordinate (pointing line of the camera center) and the *x*-*y* plane of the world coordinate (a plane where the plate of the robot lies on). The destination can be easily computed from the relationship (1):

$$\begin{bmatrix} x_d \\ y_d \\ 0 \end{bmatrix} = C_b^w[k_6] s \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + r[k_6]. \tag{3}$$

From the third row of (3), *s* is given by

$$s = -\frac{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} r[k_6]}{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} C_b^w[k_6] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}. \tag{4}$$

Inserting (4) into (3), we can obtain (x_d, y_d).

4. Mobile Robot Control Algorithm

The control objective is to move the mobile robot to the destination (x_d, y_d). There are a number of control algorithms (for example, see [16, 17]) for mobile robot control. A simple algorithm is used in this paper. The robot rotates toward the destination and moves straight toward the destination. During the movement, the robot position (x, y) is estimated using wheel encoders [18]. Let $(\hat{x}, \hat{y})$ be the estimated robot position. The robot moves until $(\hat{x}, \hat{y}) = (x_d, y_d)$. Due to the dead-reckoning error, the real position and the estimated position may not be the same, that is $(x, y) \neq (\hat{x}, \hat{y})$. Note that the camera is pointing to (x_d, y_d). If (x, y) (the position of the robot) is near (x_d, y_d), the robot is inside the viewpoint of the camera. Thus $(x - x_d, y - y_d)$ can be measured using four landmarks and the camera. If $\sqrt{(x-x_d)^2 + (y-y_d)^2} < \delta_d$, the robot stops. Otherwise, the robot is controlled until $\sqrt{(x-x_d)^2 + (y-y_d)^2} < \delta_d$ is satisfied. Once $\sqrt{(x-x_d)^2 + (y-y_d)^2} < \delta_d$ is satisfied, one remote control process is finished. If we move the remote controller pointing to another places, a new remote control process begins.

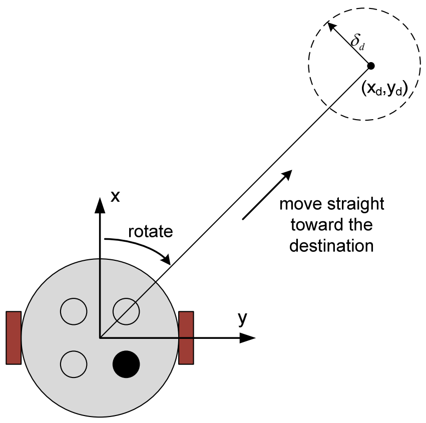

Fig. 7. Mobile robot control.

5. Error Analysis of the Computed Destination

In this section, destination computation errors are analyzed. The accuracy of dead reckoning is also important [18]. However, the error analysis of dead reckoning is not a main issue of this paper and thus is not considered.

Accuracy of computed destination depends on accuracy of C_b^w and r, which represents attitude and position of the remote controller, respectively. The initial values of C_b^w and r are computed using vision data (see (1)). Based on these values, the final values are computed using the inertial navigation algorithm. Thus errors in C_b^w and r are divided into an error in the vision algorithm part and an error in the inertial navigation algorithm.

First the error in the vision algorithm part is analyzed. While a camera is not moving, images of a landmark are taken under 3 different lighting conditions (by turning on and off lamps in the room). Each condition lasts about 10 seconds. In Fig. 8, *x* image coordinate of a circle landmark (obtained from the image processing) is plotted. We can see that the *x* values are affected by the lighting conditions. After C_b^w and r values are computed from (1), r_3 and yaw angle of C_b^w are plotted in Fig. 9 to see how much C_b^w and r fluctuate.

From the experiment, the error covariances of estimation of C_b^w (in terms of Euler angle estimation errors) and r are determined to be $\mathbf{0.5 \times 10^{-5}} I_3$, where $I_3 \in R^{3\times3}$ is an identity matrix.

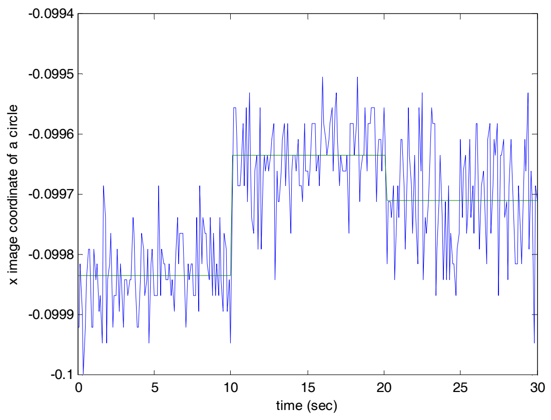

Fig. 8. *x* image coordinate variation under different lighting conditions

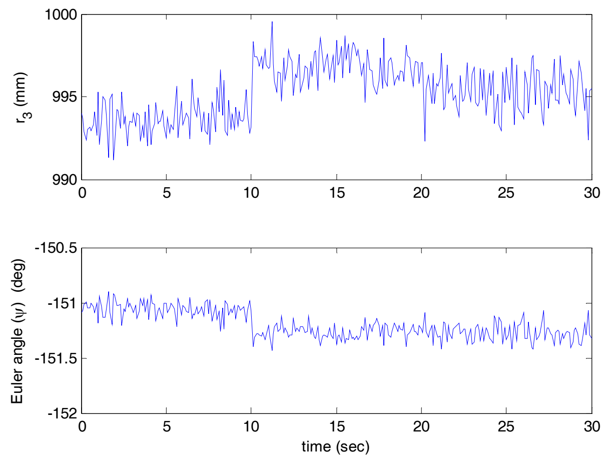

Fig. 9. Computed C_b^w and r values (r_3 and yaw Euler angle is plotted as an example)

We also found that the error covariance increases if a smaller landmark circle is used. The center of a circle is estimated by the least square estimation of the perimeter of a circle and a larger circle means more perimeter points.

The estimation error of an inertial navigation algorithm mainly depends on the accuracy of sensors (accelerometers and gyroscopes). The gyroscope output is modeled as follows:

$$y_g = \omega + v_g$$

where ω is the angular velocity and v_g is the white Gaussian sensor noise. The bias variation is ignored since each session of the remote control action takes at most a few seconds. The covariance of v_g is determined from Allan variance plot and is given by 0.00007. Similarly, the covariance of the accelerometer sensor noise is given by 0.00015.

To test the accuracy of the inertial navigation algorithm, the remote controller is placed on the floor without moving. Except for the beginning and the end of the data, the zero velocity updating is not used intentionally to test the accuracy of the inertial navigation algorithm. The position is computed with different durations of interval B (see Fig. 4): 0.6 ~ 2.5 seconds. Since the remote controller is not moving, the true position movement is zero. The computed position movement by the inertial navigation algorithm is given in Fig. 10 (x position error). It can be seen that the position error increases as time goes by. The line in the graph is the square root of Kalman filter position error covariance, which indicates a theoretical error prediction.

From the error covariance of Kalman filter, the error covariances of inertial navigation estimation of C_b^w (in terms of Euler angle estimation errors) and r are determined to be 0.0001 and 0.00019 if the duration of interval B is 1.5 seconds. Note that the error covariances depend on the duration of interval B.

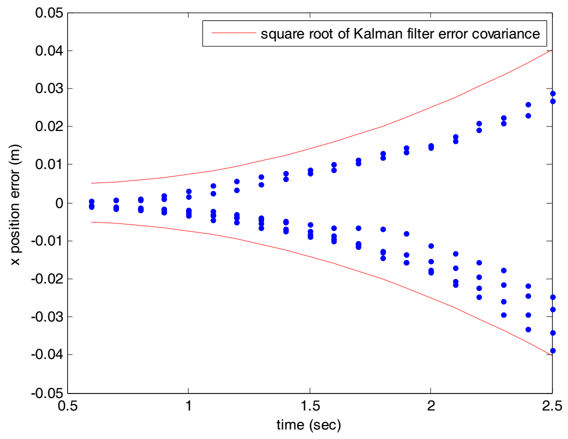

Fig. 10. Position estimation error of an inertial navigation algorithm (dots: actual position error, solid line: square root of Kalman filter position error covariance)

Finally a simulation is done to predict the destination estimation error. It is assumed that the remote controller is hold 1m behind the mobile robot at 1m height. The remote controller is moved 0.1m while rotating 45 degree. The duration of interval B is assumed to be 1.5 seconds. The destination estimation errors for 100 simulations are given in Fig. 11.

The average distance error is 0.026m and the worst case error is 0.074m. Another 100 simulations are done assuming that there is no estimation error from the vision algorithm. The average distance error is 0.023m. Then 100 simulations are done assuming there is no inertial navigation estimation error. The average distance error is 0.008m. Thus the major source of the destination error seems to be from the inertial navigation algorithm.

In [13], only rotation of a remote controller is considered while the position movement of a remote controller is ignored. The average distance error of the algorithm in [13] for the same simulation setting is given by 0.10 m. This large error is due to the fact the position movement of a

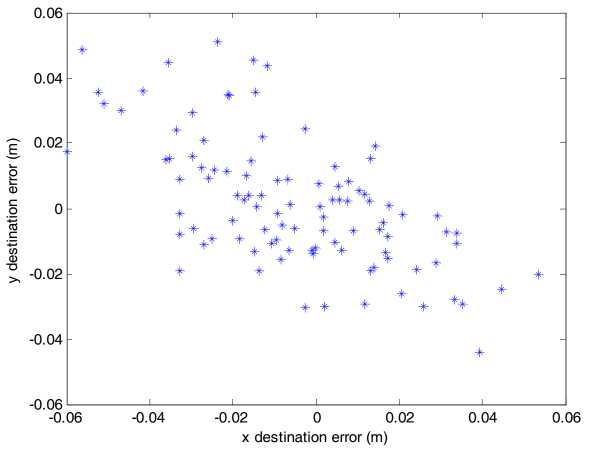

Fig. 11. Simulation result of destination computation errors

remove controller (0.1m) is completely ignored in the algorithm of [13]. On the other hand, this movement is completely compensated in the proposed method.

6. Experiments

The most important part of the proposed control algorithm is the accuracy of the destination computed from (3). To test the accuracy, four circle landmarks are placed on a grid floor (Fig. 12). After initially pointing to the landmarks, we move the remote controller to point to other places. The real destination (where the camera is pointing to) is manually marked using the laser pointer (see Fig. 2). In Fig. 13, real destinations (o) and computed destinations (*) are plotted for five different destinations. It can be seen that the real and computed destinations are not far away: the maximum distance error is $6.02cm$ while the average error is $3.82cm$. The experiment error is similar to predicted errors in the simulation.

Another experiment is done for longer ranges. The true destination and estimated destination are given in Table 1. As the destination becomes further away, the distance errors become larger. Considering it is difficult to point to the exact location in the long distance, we believe 20cm error in 5m range is acceptable.

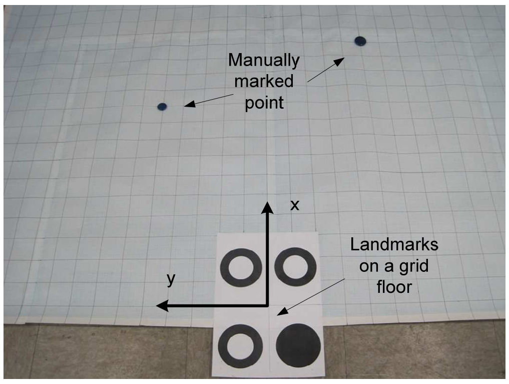

Fig. 12. Experiment setting.

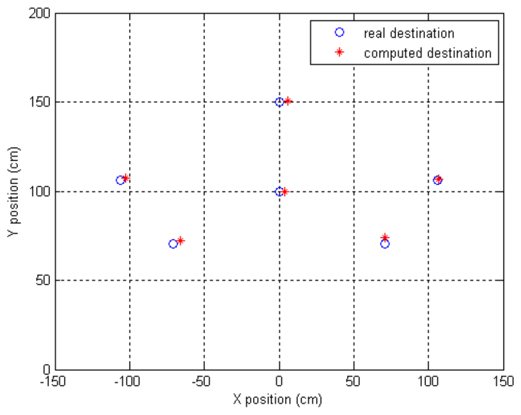

Fig. 13. Real and computed positions.

Table 1. Real and computed positions for longer range (cm)

Real destination		Computed destination		Distance error
x	y	x	y	
298	1	293.7	1.8	4.4
367	12	362.9	15.1	5.1
425	13	420.1	0.0	13.9
530	13	510.1	14.9	20.0

The next experiment is to test how the destination location and the duration of the remote controller movement (duration of interval B in Fig. 4) affect the destination accuracy.

In Fig. 14, the remote controller pointed to different destinations: $y_d = 0$ and different $x_d = (30, 60, 90, 120, 150, 180cm)$. We can see that the destination computation errors increase as the distance increases.

In Fig. 15, the remote controller pointed to the same destination $(x_d, y_d) = (90cm, 0cm)$ with different moving times. We can see that the destination computation errors increase as the moving time increases. The results in Fig. 14 and Fig. 15 are not surprising since the error of an inertial navigation algorithm increases as the time goes by due to sensor noises, sensor bias and numerical errors.

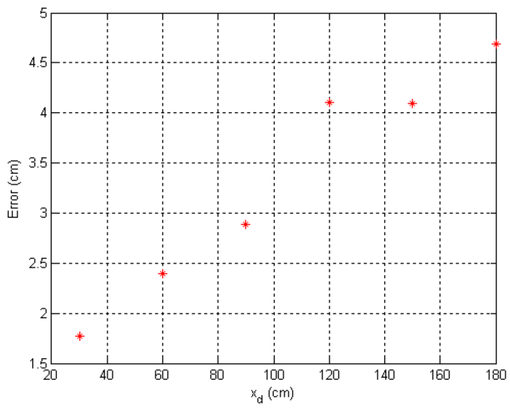

Fig. 14. The distance computation error with different destinations ($y_a = 0$ and different x_d).

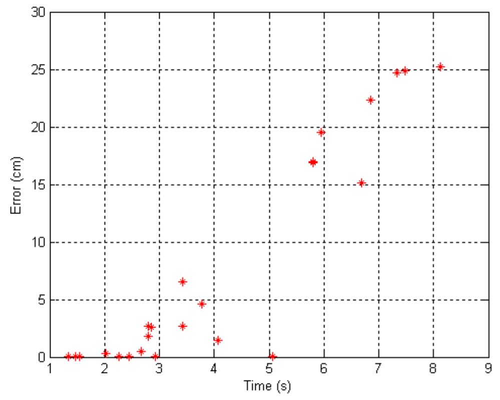

Fig. 15. The destination computation error with different movement time

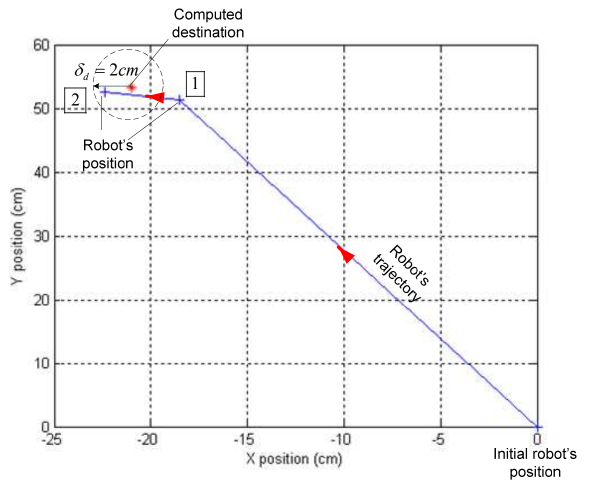

Fig. 16. Robot's trajectory.

In Fig. 16, the trajectory of the robot is given. '*' point is the computed destination (x_d, y_d). The robot first rotated toward the destination and moved to the point $\boxed{1}$. At this point, the camera can see the robot and computes $(x - x_d, y - y_d)$, which is the difference between the real position and the destination of the robot. Since the distance is larger than $\delta_d = 2cm$, the robot moves toward (x_d, y_d). We can see that the robot stops at $\boxed{2}$ inside a radius δ_d circle with the center (x_d, y_d).

7. Conclusion

A remote control algorithm of a mobile robot using inertial sensors and vision is proposed. The relative position between a mobile robot and a remote controller is computed. Fast movement of the remote controller is estimated using an inertial navigation algorithm. Thus we are able to control the robot even if it is temporarily outside the viewpoint of the camera.

Destination estimation errors are analyzed through experiments and simulations. In the experiment, we showed that the estimated robot destination computed by an inertial navigation algorithm has a few centimeter errors with a few seconds movement. The destination error increases if the remote controller movement time increases or the destination distance becomes larger. With a simple robot control algorithm, it is shown that the robot moves to the destination successfully.

Acknowledgement

This work was supported by National Research Foundation of Korea Grant funded by the Korean Government (No. 2012-0008400).

References

[1] S. K. Agrawal, X. Chen and J. C. Galloway, "Training special needs infants to drive mobile robots using force-feedback joystick," IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 4797-4802.

[2] G. Bourhis and M. Sahnoun, "Assisted control mode for a smart wheelchair," IEEE 10th International Conference on Rehabilitation Robotics (ICORR), 2007, pp. 158-163.

[3] G. Novak, "Roby-Go, a prototype for several MiroSOT soccer playing robot," Second IEEE International Conference on Computational Cyber-netics (ICCC), 2004, pp. 207-212.

[4] G. Novak and S. Mahlknecht, "TINYPHOON a tiny autonomous mobile robot," Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE), 2005, pp. 1533-1538.

[5] M. Achtelik, T. Zhang, K. Kuhnlenz and M. Buss, "Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors," International conference on Mechatronics and Automation (ICMA), 2009, pp. 2863-2869.

[6] B. Hartmann, N. Link and Gert F. Trommer, "Indoor 3D position estimation using low-cost inertial sensors and marker-based video tracking," IEEE/ION Position Location and Navigation Symposium (PLANS), 2010, pp. 319-326.

[7] H. Q. P. Nguyen, H. J. Kang and Y. S. Suh, "A visual-inertial servoing method for tracking object with two landmarks and an inertial measurement unit," International Journal of Control, Automation and Systems, Vol. 9, pp. 317-327, 2011.

[8] J. Chen and A. Pinz, "Structure and motion by fusion of inertial sensors and vision-based tracking," Proceedings of the 28th OAGM/AAPR Conference, 2004, pp. 55- 62.

[9] L. L. Ong, M. Ridley, J. H. Kim, E. Nettleton and S. Sukkarieh, "Six DoF decentralised SLAM," Australasian Conference on Robotics and Automation, 2003, pp. 10-16.

[10] C. P. Lu, G. D. Hager and E. Mjolsness, "Fast and globally convergent pose estimation from video images," IEEE transactions on pattern analysis and machine intelligence, Vol. 22, pp. 610-622, June 2000.

[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Communication of the ACM, Vol. 24, pp. 381-395, 1981.

[12] D. H. Titterton and J. L. Weston, Strapdown inertial navigation technology, Peter Peregrinus Ltd., IEE, 1997.

[13] Y. S. Suh, S. K. Park, D. N. Kim and K. H. Jo, "Remote control of a moving robot using the virtual

link," IEEE International Conference on Robotics and Automation, 2007, pp. 2343-2348.

[14] D. A. Forsyth and J. Ponce, Computer vision: a modern approach, Prentice Hall, January, 2003.

[15] Y. S. Suh and S. K. Park, "Pedestrian inertial navigation with gait phase detection assisted zero velocity updating," Proceedings of the 4th International Conference on Automation robots and agents (ICARA), 2009, pp. 336-341.

[16] S. Lee, Y. Youm and W. Chung, "Control of car-like mobile robots for posture stabilization," Proceeding of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99), Vol. 3, 1999, pp. 1745-1750.

[17] B. S. Park, J. B. Park and Y. H. Choi, "Adaptive observer-based trajectory tracking control of nonholonomic mobile robots," International Journal of Control, Automation and Systems, Vol. 9, pp. 534-541, 2011.

[18] J. Borenstein and L. Feng, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots," IEEE Journal of Robotics and Automation, Vol. 12, pp. 869-880, December, 1996.

Dang Quoc Khanh He received his B.S. degree of Automation from Department of Electrical Engineering at Hanoi University of Science and Technology, Vietnam in 2009. In 2012, he received his M.S. degree from School of Electrical Engineering at University of Ulsan, Korea. He is currently pursuing the Ph.D degree at University of Ulsan. His research interests include mobile robot control, motion tracking and personal navigation.

Young-Soo Suh He received his B.S. and M.S. degrees from the Department of Control and Instrumentation Engineering at Seoul National University, Korea in 1990 and 1992, and his Ph.D. degree from the Department of Mathematical at Engineering and Information Physics at the University of Tokyo, Japan in 1997. He is currently a professor in the Department of Electrical Engineering, University of Ulsan, Korea. His research interests include networked control systems and attitude estimation and control, and personal navigation systems.