Visual SLAM and Moving-object Detection for a Small-size Humanoid Robot

Yin-Tien Wang[1], Ming-Chun Lin[2] and Rung-Chi Ju[1]
[1]Department of Mechanical and Electro-Mechanical Engineering, Tamkang University, Taipei Hsien, Taiwan.
[2]Department of Electrical Engineering, Technology and Science Institute of Northern Taiwan, Taipei, Taiwan.
Corresponding author E-mail: ytwang@mail.tku.edu.tw

Abstract: *In the paper, a novel moving object detection (MOD) algorithm is developed and integrated with robot visual Simultaneous Localization and Mapping (vSLAM). The moving object is assumed to be a rigid body and its coordinate system in space is represented by a position vector and a rotation matrix. The MOD algorithm is composed of detection of image features, initialization of image features, and calculation of object coordinates. Experimentation is implemented on a small-size humanoid robot and the results show that the performance of the proposed algorithm is efficient for robot visual SLAM and moving object detection.*
Keywords: *Simultaneous Localization and Mapping (SLAM); Humanoid Robot; Moving Object Detection.*

1. Introduction

In recent years, SLAM systems have been successfully implemented and validated by many researchers (Dissanayake, M.W.M.G. et al., 2001; Davison, A.J. et al., 2007; Montemerlo, M. & Thrun, S., 2007; Wang, C.C., 2007). Especially, the *MonoSLAM* developed by Davison, A.J. et al. (2007) provides a real-time algorithm which can locates the 3D trajectory of a monocular camera and maps the beacons in the environments, simultaneously. Nevertheless, the research in this paper aims at providing a novel moving object detection (MOD) algorithm for a class of visual SLAM systems, as well as implementing the visual SLAM with MOD on a small-size humanoid robot system.
MonoSLAM (Davison, A.J. et al., 2007) utilizes an extended Kalman filter (EKF) to update the estimation of the robot state as well as the map of beacons in the environments, recursively. We extend the method of state estimation to detection of a moving object. The basic concept is to collect the nearby image features together to form a prospective object in the environments. In the paper, the object is assumed to be a rigid body and its coordinate system in space is represented by a position vector and a rotation matrix. We present a procedure to form the body-fixed coordinate by choosing three arbitrary and non-collinear image features on the object, and then determine the rotational angles of the object in terms of the roll-yaw-pitch coordinate system.
The developed visual SLAM with MOD is verified through experiments on a small-size humanoid robot. The experimental results show that the performance of the proposed algorithm is efficient for supporting the small-size humanoid robot simultaneously navigating and detecting moving objects in the environments.
The contributions in this paper are two-fold. First, we develop a novel moving object detection algorithm and integrate with *MonoSLAM*. Second, the SLAM with MOD is implemented on a small-size humanoid robot system.

2. EKF-based SLAM

Based on the EKF estimation method, Davison, A.J. et al. (2007) proposed the *MonoSLAM* which can proceed to estimate the state of the free-moving camera with a monocular vision and construct the environmental map, simultaneously. The free-moving camera is presumed to be at constant velocity, and the acceleration is caused by an impulse noise from the external force. Therefore, the velocity noise of the camera is defined as:

$$w_k = \begin{bmatrix} w_{vk} \\ w_{\omega k} \end{bmatrix} = \begin{bmatrix} a_k \Delta t \\ \alpha_k \Delta t \end{bmatrix}$$

where k is the time step; a and α are linear and angular acceleration of the camera, respectively; Δt is the sampling time interval; w_v and w_ω are linear and angular velocity noise caused by acceleration, respectively. Using the EKF method to estimate the state of the system, and the vector of the state is chosen as

$$x = [x_R^T \; Y_1^T \; Y_2^T \cdots Y_n^T]^T$$

x_R is a 12×1 state vector of the camera including the position r^W, rotational angle ϕ^C, velocity v^W, and angular velocity ω^C; Y_i is the three-dimensional coordinates in world frame of ith image feature; n is the number of the image features. The state transition equation of the camera is:

$$x_{Rk} = f(x_{Rk-1}, u_{k-1}, w_{k-1}) = \begin{bmatrix} r_k^W \\ \phi_k^C \\ v_k^W \\ \omega_k^C \end{bmatrix} = \begin{bmatrix} r_{k-1}^W + (v_{k-1}^W + w_{vk-1})\Delta t \\ \phi_{k-1}^C + (\omega_{k-1}^C + w_{\omega k-1})\Delta t \\ v_{k-1}^W + w_{vk-1} \\ \omega_{k-1}^C + w_{\omega k-1} \end{bmatrix}$$

The Jacobian matrices A_k and W_k for calculating the covariance matrix are given as

$$A_k = \frac{\partial f}{\partial x_R}\left(x_{Rk-1|k-1}, u_{k-1}, 0\right) = \begin{bmatrix} I & 0 & \Delta t & 0 \\ 0 & I & 0 & \Delta t \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

$$W_k = \frac{\partial f}{\partial w}\left(x_{Rk-1|k-1}, u_{k-1}, 0\right) = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2.1. Sensor Model

MonoSLAM uses a monocular vision as the only sensing device, and the measurement vector is given as

$$z_k = g(x_k, v_k) = [z_{1k}^T \; z_{2k}^T \cdots z_{mk}^T]^T$$

m is the number of the observed image features in current measurement. For one observed image feature, the measurement is

$$z_{ik} = \begin{bmatrix} I_{ixk} \\ I_{iyk} \end{bmatrix} = \begin{bmatrix} I_{ixk\_true} \\ I_{iyk\_true} \end{bmatrix} + \begin{bmatrix} v_{ixk} \\ v_{iyk} \end{bmatrix} \qquad for \;\; i = 1,\, 2,\, \cdots,\, m$$

I_{ix} and I_{iy} represent the measured pixel coordinates of the i^{th} image feature with Gauss noise v_{ix} and v_{iy}; I_{ix_true} and I_{iy_true} are the real pixel coordinates. The perspective projection method (Hutchinson, S. et al., 1996) is employed in this research to model the transformation from 2D image plane to 3D space coordinates,

$$\begin{bmatrix} I_{ix} \\ I_{iy} \end{bmatrix} = \begin{bmatrix} u_0 + f_c k_u \dfrac{h_{ix}^C}{h_{iz}^C} \\ v_0 + f_c k_v \dfrac{h_{iy}^C}{h_{iz}^C} \end{bmatrix} \tag{1}$$

where f_c is the focal length of the camera denoting the distance from camera center to image plane; (u_0, v_0) is the offset pixel vector from the hardware image plane to pixel image plane; k_u and k_v are the image pixel correctional parameters. Assuming that there is no distortion phenomenon on the image plane and we make k_u and k_v as 1; $h_i^C = [h_{ix}^C \;\; h_{iy}^C \;\; h_{iz}^C]^T$ is the ray vector of the image features in the camera frame. The 3D coordinates of an image feature in world frame, as shown in Figure 1, are given as

$$Y_{ik} = r_k^W + R_C^W h_{ik}^C \tag{2}$$

R_C^W is the rotational matrix from the world frame {W} to the camera frame {C}, represented by using the elementary rotations (Sciavicco L. & Siciliano, B., 1996),

$$R_C^W = \begin{bmatrix} c\phi_y^C c\phi_z^C & s\phi_x^C s\phi_y^C c\phi_z^C - c\phi_x^C s\phi_z^C & c\phi_x^C s\phi_y^C c\phi_z^C + s\phi_x^C s\phi_z^C \\ c\phi_y^C s\phi_z^C & s\phi_x^C s\phi_y^C s\phi_z^C + c\phi_x^C c\phi_z^C & c\phi_x^C s\phi_y^C s\phi_z^C - s\phi_x^C c\phi_z^C \\ -s\phi_y^C & s\phi_x^C c\phi_y^C & c\phi_x^C c\phi_y^C \end{bmatrix}$$

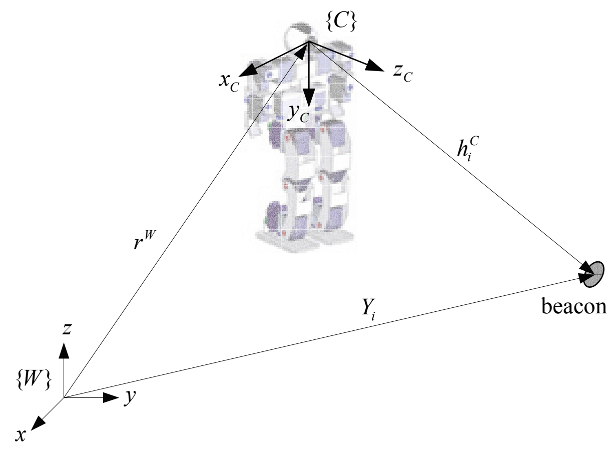

Fig. 1. The camera frame and the world frame.

where $c\phi^C = \cos\phi^C$ and $s\phi^C = \sin\phi^C$. We can utilize Eqn. (2) to calculate the ray vector of an image feature in the camera frame. The vectors h_x^C, h_y^C, and h_z^C are obtained as

$$h_x^C = c\phi_y^C c\phi_z^C (Y_{ix} - r_x^W) + c\phi_y^C s\phi_z^C (Y_{iy} - r_y^W) - s\phi_y^C (Y_{iz} - r_z^W) \tag{3}$$

$$h_y^C = (s\phi_x^C s\phi_y^C c\phi_z^C - c\phi_x^C s\phi_z^C)(Y_{ix} - r_x^W) + s\phi_x^C c\phi_y^C (Y_{iz} - r_z^W) + (s\phi_x^C s\phi_y^C s\phi_z^C + c\phi_x^C c\phi_z^C)(Y_{iy} - r_y^W) \tag{4}$$

$$h_z^C = (c\phi_x^C s\phi_y^C c\phi_z^C + s\phi_x^C s\phi_z^C)(Y_{ix} - r_x^W) + c\phi_x^C c\phi_y^C (Y_{iz} - r_z^W) + (c\phi_x^C s\phi_y^C s\phi_z^C - s\phi_x^C c\phi_z^C)(Y_{iy} - r_y^W) \tag{5}$$

Substituting the ray vectors, Eqns. (3)-(5), into Eqn. (1), we can get the sensor model in terms of coordinates of the image feature in the image plane.

2.2. Image Feature Initialization

During navigating in the environments, the robot locates its global coordinate by consulting the positions of a group of fixed features or beacons. These features are abstracted from the captured image and an initialization process is applied to determine the 3D coordinates of these image features. In the paper, the image feature initialization algorithm proposed by Wang, Y.T. et al. (2009) is employed to initialize the features for SLAM and moving object detection. The procedures of the algorithm include detection of image features, selection of "good features" (Shi, J. & Tomasi, C., 1994), calculation of image depths, and update of feature locations.

In the feature initialization algorithm (Wang, Y.T. et al., 2009), the features are extracted from the image by using the method of Speeded Up Robust Features (SURF) (Bay, H. et al., 2008). SURF is a scale-invariant method for detection of image features. It detects region features from an image and obtains the location and the descriptor vector of each interest point. The basic concept of a scale-invariant method is to detect image features by investigating the determinant of Hessian matrix ***H*** (Lindeberg, T., 1998). Bay, H. et al. (2008) utilize a box

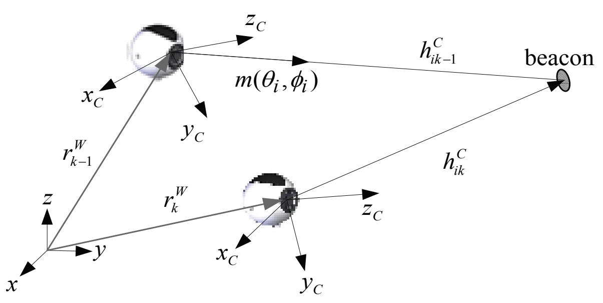

Fig. 2. Image depth obtained by two successive images

filter to process on the image instead of calculating the Hessian matrix, and then the determinant of Hessian matrix is approximated by

$$\det(H)_{\text{approx.}} = D_{xx}D_{yy} - (wD_{xy})^2$$

where D_{ij} are the images filtered by the corresponding box filters; w is a weight constant. The interest points or features are extracted by examining the extreme value of determinant of Hessian matrix. Furthermore, the unique properties of the extracted features are described by using a 64 dimensional descriptor vector.

The extracted features from two successive images can be matched by checking the Euclidean distance d between the corresponding descriptor vectors, and using the nearest neighbor ratio matching strategy (Lowe, D.G., 2004). Therefore, the region features of two images can be tracked efficiently. The matching ratio r is defined as the ratio of the smallest distance d_{1st} to the second smallest distance d_{2nd}. If the ratio r closes to 0.7, we say that these two features are matched.

The depths of image features are calculated using two successive images, as shown in Fig. 2. Assume that there are m image features with 3D position vectors, Y_i, which can be described as Eqn. (2). After determining a good feature i, the xyz coordinate of the feature Y_i is obtained by calculating the image depth of the feature in two successive images (Wang, Y.T. et al., 2009).

3. Moving Object Detection

The moving object can also be detected by investigating the image features on the object, as shown in Fig. 3. For reasons of simplicity, we make three suppositions: first, the image features of a moving object are known; so, it is not necessary to carry on the procedure of object recognition; second, the object is a rigid body and not deformable; therefore, the position vector and rotation matrix of a rigid body can be used to describe the object movement; finally, the object moves slowly or stops sometimes, and then the images taken by the camera may be regarded as at the same position.

Before the process of object detection, the body-fixed coordinate of the object is constructed and the feature database is established by using the image feature detection and tracing method. The description and position vectors of the image features on the object are stored in the established database. We setup the body-fixed coordinate system on the object by using the procedure defined as follows.

Procedure 1: body-fixed coordinate setting

Choose three image features *A*, *B*, and *C* from the initialized feature set. The origin of the coordinate system is set at *A*, as shown in Figure 3. The vector $\overrightarrow{AB}$ from *A* to *B* is *x* axis of body coordinate; the vector cross product $\overrightarrow{AB}\times\overrightarrow{AC}$ is the *y* axis; and $\overrightarrow{AB}\times(\overrightarrow{AB}\times\overrightarrow{AC})$ is the *z* axis.

The algorithm for moving object detection (MOD) is summarized as the following steps:

a. At the current scene, choose three image features that have the largest values of the determinant, called F_j, F_{j+1} and F_{j+2}. And then calculate the description vector of the image feature and its 3D position in world coordinate.

b. Utilize **Procedure 1** and combine with the 3D positions of three image features obtained in Step (a) to set up the coordinate system {a2} on the object. The origin of the coordinates is set at F_j. The rotation matrix $\mathbf{R}_2^0$ is utilized to represent the transformation from the fixed coordinate to the current scene of the moving object.

c. By using the description vectors of image features F_j, F_{j+1} and F_{j+2}, we can retrieve their correspondences, features M_j, M_{j+1} and M_{j+2} from the feature database. Note that M_j, M_{j+1} and M_{j+2} are the features extracted from the initial scene. Using **Procedure 1** and combine with the 3D positions of those three image features in initial scene we can set up the coordinate system {a1} for the object at the initial scene. Similarly, the rotation matrix $\mathbf{R}_1^0$ represents the transformation from the fixed coordinate to the initial scene of the moving object.

d. Utilize the rotation matrices $\mathbf{R}_1^0$ and $\mathbf{R}_2^0$ to determine the rotation matrix $\mathbf{R}_2^1$ which represents the transformation of the object from the initial scene to the current scene,

$$\mathbf{R}_2^1 = \mathbf{R}_2^0(\mathbf{R}_1^0)^T$$

Note that, the rotation matrix $\mathbf{R}_2^1$ is described by using the roll-pitch-yaw (RPY) angles in fixed coordinate system,

$$\begin{bmatrix} c\theta_z c\theta_y & c\theta_z s\theta_y s\theta_x - s\theta_z c\theta_x & c\theta_z s\theta_y c\theta_x + s\theta_z s\theta_x \\ s\theta_z c\theta_y & s\theta_z s\theta_y s\theta_x + c\theta_z c\theta_x & s\theta_z s\theta_y c\theta_x - c\theta_z s\theta_x \\ -s\theta_y & c\theta_y s\theta_x & c\theta_y c\theta_x \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

The RPY angles of the rotation matrix $\mathbf{R}_2^1$ are determined as

$$\theta_x = A\tan 2\left(r_{32}, r_{33}\right)$$

$$\theta_y = A\tan 2\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right)$$

$$\theta_z = A\tan 2\left(r_{21}, r_{11}\right)$$

e. By investigating the relative position between points M_j, M_{j+1}, M_{j+2} and points *A*, *B*, *C*, we can determine the image correspondence *A*', *B*' and *C*' at the current

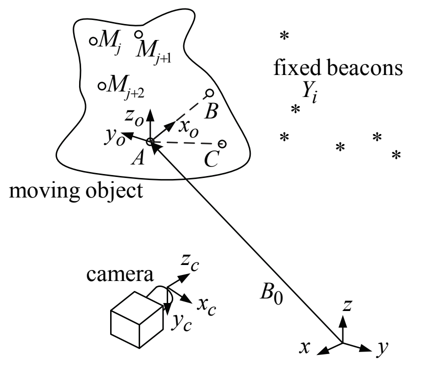

Fig. 3. A group of fixed beacons and a moving object in the initial scene t=0

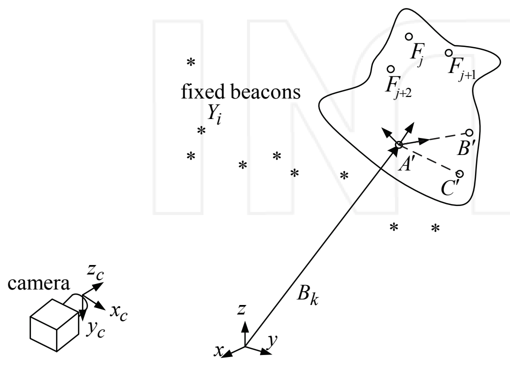

Fig. 4. The same group of fixed beacons and a moving object in the current scene t=k

scene. In the paper, the translational vector $\overrightarrow{AA'}$ as well as the rotational matrix $\boldsymbol{R}$ is utilized to describe the object moving from the initial scene to the current scene, as shown in Fig. 3 and Fig. 4.

4. Experimental Results

4.1. Humanoid Robot

A small-size humanoid robot is designed and fabricated for demonstration, which is equipped with a Microsoft Windows-based industrial PC. The appearance of the designed robot is depicted in Fig. 5 (right). The control system comprises three subsystems, including a vision sensor system, a PC-based control system, and a motor drive system. Each subsystem is able to work independently and also in coordination with each other. The camera system is the only sensor to provide the robot position information in the environments. In this research, we develop the control system by utilizing a Windows-based industrial PC, Wafer-LUKE533, provided by a local vendor. The fabricated robot is 51cm in height and 3kg in weight, including two 12V/2.5AH Li batteries. One battery provides the power for 18 servo motors and the other provides for the controller. The motor drive system is composed of a SSC-32 circuit board and is responsible for driving all the servo motors.

4.2. System Startup

First experimental work is the startup procedure of the system. The EKF-based SLAM must have a startup procedure to determine a reasonable scale of the map. In the startup procedure, several features with known 3D positions and description vectors, as shown in Fig. 5 (left), are stored in the database. The robot stands in front of the image features. By using the SURF method, the image features are detected from the captured image and the 3D positions are retrieved from the pre-stored database. The initial camera state is assumed to be located at an arbitrary position, for example, $[0,0,0]^T$. However, the real camera state locates at $[3,-70,2.7]^T$. Three startup experiments are tested and the results are depicted in Figs. 6(a) and (b). These figures denote the estimated camera states plotted in x-y and x-z plane, respectively. We can see from the plots, the EKF estimations of the camera state begins from the arbitrary initial position $[0,0,0]^T$ and converge to $[0,-55.8,3.7]^T$. The reasons the

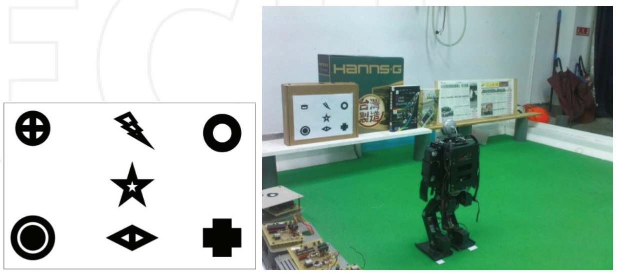

Fig. 5. Pre-stored image features (left); Robot stands in front of the pre-stored image features (right).

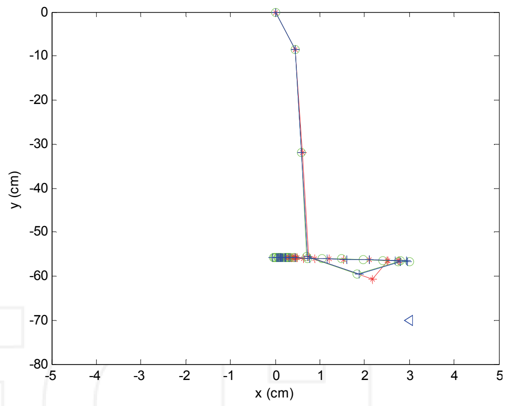

Fig. 6. (a) Experiments of the system startup procedure. The plot of the camera state in x-y plane

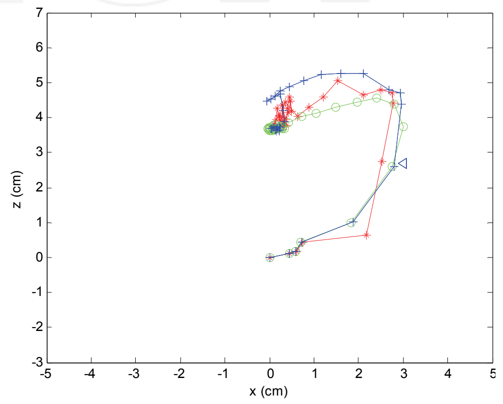

Fig. 6. (b) Experiments of the system startup procedure. The plot of the camera state in x-z plane

estimated states do not converge to the real positions, indicated as a triangular mark in both plots, are the un-calibrated and distortion problems of the low-cost camera.

4.3. Humanoid Robot SLAM

The EKF-based visual SLAM is also implemented on the small-size humanoid robot. We integrate the procedures including the image feature detection and tracking method, feature initialization, system startup procedure, and EKF-based state estimation. And then perform the robot self-localization and mapping procedures simult-aneously. In this experiment, the robot moves from the left- to right-side of the field, as shown in Fig. 7 and the estimate state and image features are depicted as a 3D map shown in Fig. 8. In the plot, the dots indicate the landmarks obtained from the initialized image features and the asterisks represent the state of the camera equipped on the robot.

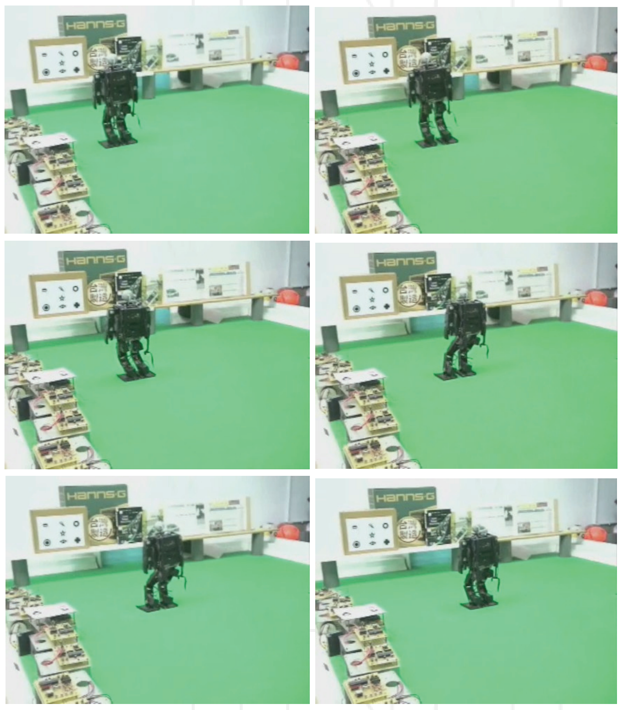

Fig. 7. Humanoid robot moves from left- to right-side of the field. Meanwhile, the EKF-based visual SLAM is performed

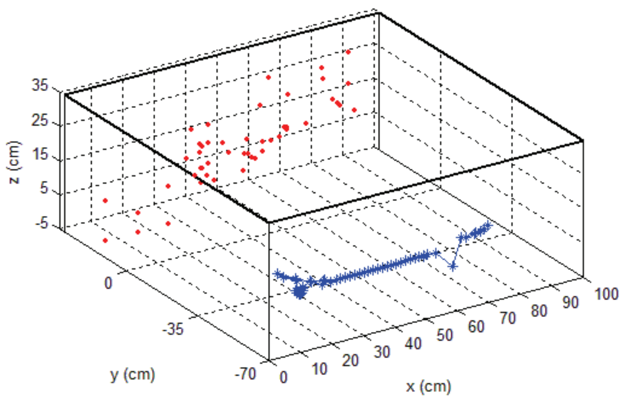

Fig. 8. Three-dimensional plot of the estimate state and image features.

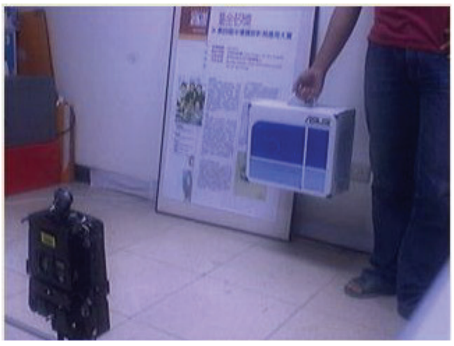

Fig. 9. Experimental set-up

Location	Object Displacement (*cm*)			Object Orientation (*degree*)		
	x	*y*	*z*	*yaw*	*pitch*	*roll*
B_0	-18.5	90.0	23.0	0°	0°	0°
B_1	-10.0	102.5	6.7	5.8°	-15.1°	-35.0°
B_2	-12.1	77.5	21.4	-8.6°	15.3°	52.6°
B_3	-22.1	101.7	26.5	4.7°	38.9°	-21.3°

Table 1. Displacement and orientation of the moving object

4.4. Moving Object Detection

The experimental set-up of moving object detection is depicted in Fig. 9. The humanoid robot implements robot SLAM and moving object detection in a corner of the laboratory. Four different locations of the moving object are tested and shown in Fig. 10. By using the proposed initialization algorithm, fixed features (yellow circles) in the environments are selected as beacons for robot SLAM, while the moving features (black circles) are classified as the image features on the moving object. The displacement and orientation of the detected object are listed in Table 1. In the table, the object displacement *xyz* indicates the translational displacement from the origin of the world coordinate (set on the camera) to the origin of the coordinate on the box, which is located on top-right corner of the box. The object orientation represents the *yaw-pitch-roll* angles of the box with respect to world coordinate. The results show that the proposed method determines the position and orientation of a moving rigid object effectively.

5. Conclusions

In this research, we developed an algorithm of moving object detection (MOD) and integrated it into the *MonoSLAM*. The moving object is detected by extracting the image features on the rigid body. Three arbitrary features are chosen to form a coordinate systm representing the moving object at current scene. The coordinate system of the initial scene is obtained by matching the extracted image feature with that in the database. The displacement and orientation of the moving object is determined by solving the inverse problem of the rotation matrix from initial scene to current scene.

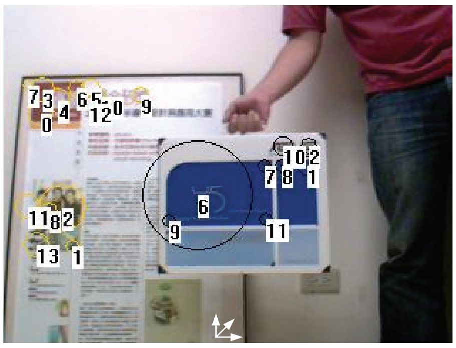

Fig. 10. (a) Moving object detection: Location B_0

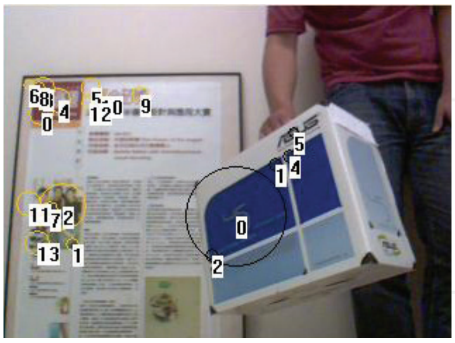

Fig. 10. (b) Moving object detection: Location B_1

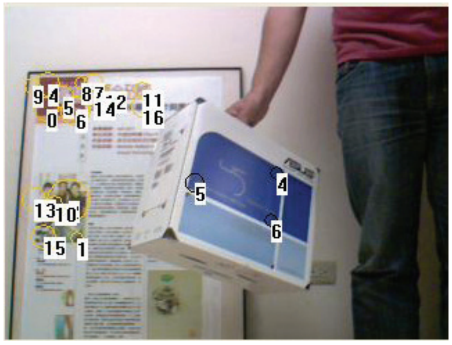

Fig. 10. (c) Moving object detection: Location B_2

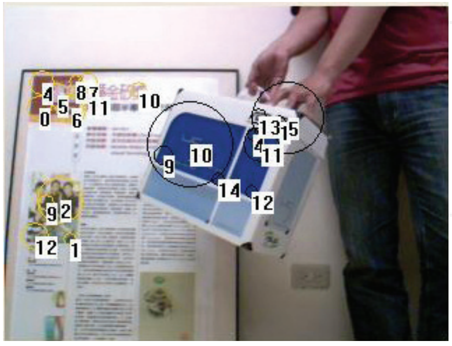

Fig. 10. (d) Moving object detection: Location B_3

The tasks of SLAM with MOD are implemented on a PC-based controller for a small-size humanoid robot. The contribution of this research is in two aspects: firstly, a novel algorithm for moving object detection is developed for the EKF-based SLAM by using SURF, a scale-invariant image feature extraction method. Secondly, the SLAM with MOD is implemented on a small-size humanoid robot system. Experimental works are performed in this paper and the results show that the SLAM with the proposed moving object detection algorithm has the capability to support the humanoid robot simultaneously navigating and detecting moving objects in the environments.

Up to now, the developed detector can determine rigid moving objects. In the future, we will extend the method to detecting multiple objects with different moving velocities, or non-rigid moving objects like human body with articulated arms and legs.

6. Acknowledgment

This work was supported in part by the National Science Council in Taiwan under grant no. NSC98-2221-E-032-012 to Y.T. Wang.

7. References

Bay, H.; Ess, A., Tuytelaars, T. & Van Gool, L. (2008). SURF: speeded up robust features, *Computer Vision and Image Understanding*, vol.110, pp.346-359.

Davison, A.J.; Reid, I.D., Molton, N.D. & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.29, no.6, pp.1052-1067.

Dissanayake, M.W.M.G.; Newman, P., Clark, S., Durrant-Whyte, H. & Csorba, M. (2001). A solution to the Simultaneous Localization and Map Building (SLAM) Problem, *IEEE Transactions on Robotics and Automation*, vol.17, no.3, pp.229-241.

Hutchinson, S.; Hager, G.D. & Corke, P.I. (1996). A tutorial on visual servo control, *IEEE Transactions on Robotics and Automation*, vol.12, no.5, pp.651-670.

Lindeberg, T. (1998). Feature detection with automatic scale selection, *International Journal of Computer Vision*, vol.30, no.2, pp79-116.

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision*, vol.60, no.2, pp. 91-110.

Montemerlo, M. & Thrun, S. (2007). *FastSLAM*, Springer-Verlag.

Sciavicco L. & Siciliano, B. (1996). *Modelling and Control of Robot Manipulators*, McGraw-Hill, New York, NY, 1996.

Shi, J. & Tomasi, C. (1994). Good features to track, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.593-600.

Wang, C.C.; Thorpe, C., Thrun, S., Hebert, M. & Durrant-Whyte, H. (2007). Simultaneous localization, mapping and moving object tracking, *International Journal of Robotics Research*, vol.26, no.9, pp.889-916.

Wang, Y.T.; Lin, M.-C., Ju, R.-C. & Huang, Y.-W. (2009). Image Feature Initialization for SLAM and Moving Object Detection, *Innovative Computing, Information and Control - Express Letters*, vol.3, no.3(A), pp.477-482.