To appear in *International Journal of Social Robotics*
Special Issue on People Detection and Tracking, 2010

Real-time Motion Tracking from a Mobile Robot

Boyoon Jung · Gaurav S. Sukhatme

the date of receipt and acceptance should be inserted later

Abstract A mobile robot needs to perceive the motions of external objects to perform tasks successfully in a dynamic environment. We propose a set of algorithms for multiple motion tracking from a mobile robot equipped with a monocular camera and a laser rangefinder. The key challenges are 1. to compensate the ego-motion of the robot for external motion detection, and 2. to cope with transient and structural noise for robust motion tracking. In our algorithms, the robot ego-motion is directly estimated using corresponding feature sets in two consecutive images, and the position and velocity of a moving object is estimated in image space using multiple particle filters. The estimates are fused with the depth information from the laser rangefinder to estimate the partial 3D position. The proposed algorithms have been tested with various configurations in outdoor environments. The algorithms were deployed on three different platforms; it was shown that various type of ego-motion were successfully eliminated and the particle filters were able to track motions robustly. The real-time capability of the tracking algorithm was demonstrated by integrating it into a robot control loop.

Keywords mobile robot, motion tracking, particle filter.

Boyoon Jung
NavCom Technology, Inc.,
A John Deere Company,
Torrance, USA.
E-mail: bjung@navcomtech.com

Gaurav S. Sukhatme
Robotic Embedded Systems Laboratory,
Center for Robotics and Embedded Systems,
University of Southern California,
Los Angeles, USA.
E-mail: gaurav@robotics.usc.edu

1 Introduction

Motion tracking is a fundamental capability that a mobile robot must have in order to operate in a dynamic environment. Moving objects (*eg.*, people) are often subjects for a robot to interact with, and in other contexts (*eg.*, traffic) they could be potentially more dangerous for safe navigation compared to stationary objects. Further, capabilities like localization and mapping critically depend on separating moving objects from static ones. Finally motion is the most critical feature to track for many surveillance or security applications. Robust motion detection and tracking are key enablers for many mobile robot applications.

The motion tracking problem from a mobile robot is illustrated in Figure 1. There are multiple moving objects in the vicinity of a mobile robot, and their positions are measured using the on-board sensors; some measurement errors are added during the process. An estimation process is required to compute the positions and velocities of the moving objects in the robot's local coordinate system from the measurement contaminated with noise. In the variant of the problem studied here, we require real-time estimates without prior knowledge about the number of moving objects, the motion model of objects, or the structure of the environment.

There are two main challenges in the motion tracking problem. First, there are two *independent motions* involved - the ego-motion of the mobile robot and the external motions of moving objects. Since these two motions appear blended in the sensor data, the ego-motion of the robot needs to be eliminated so that the remaining motions, which are due to moving objects, can be detected. Second, there are *various types of noise* added at various stages. For example, real outdoor images are contaminated by various noise sources

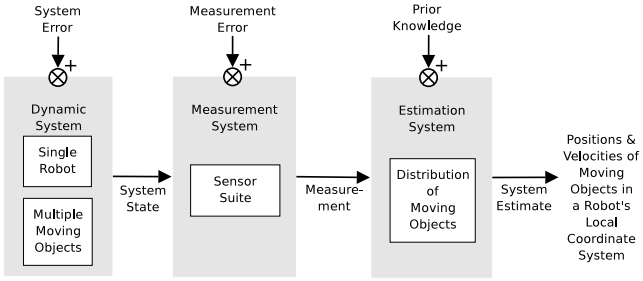


Fig. 1 Motion tracking from a mobile robot.

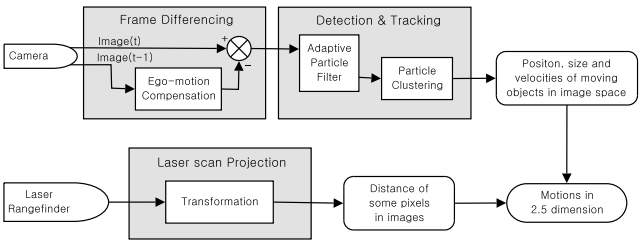


Fig. 2 Processing sequence of motion tracking system.

including poor lighting conditions, camera distortion, unstructured and changing shape of objects, etc. Perfect ego-motion compensation is rarely achievable, thus it adds another type of uncertainty to the system. Some of these noise terms are transient and some of them are constant over time.

Our approach to the problem is to design a simple and fast ego-motion compensation algorithm in the pre-processing stage for real-time performance, and to develop a probabilistic filter in the post-processing stage for uncertainty and noise handling. Since the sequence of camera images contains rich information of object motion, a monocular camera is utilized for motion detection and tracking. A laser rangefinder provides depth information of image pixels for partial 3D position estimation. Figure 2 shows the processing sequence of our motion tracking system.

The rest of the paper is organized as follows. Section 2 summarizes the related work on this topic. The detailed ego-motion compensation algorithm is given in Section 3, and the design of the probabilistic filter is explained in Section 4. Section 5 describes how to fuse the estimation result in image space with laser rangefinder data, and Section 6 reports the experimental results and analyzes the performance of the proposed algorithms. The current status and possible improvements are discussed in Section 7.

2 Related Work

The target tracking problem in general has been studied by various research group from different points of view, and a huge body of literature has been produced. In this section, we confine the scope of literature coverage to the research on *motion* detection and tracking using *visual sensor(s)* from a *mobile platform*.

The most fundamental requirement to achieve stable motion detection from a mobile platform is to compensate the motion of a sensor. The computer vision community has proposed various methods to stabilize camera motions by tracking features [48,6,58,15,27] and computing optical flow [31,44,18]. These approaches focus on how to estimate the transformation (*homography*) between two image coordinate systems. However, the motions of moving objects are typically not considered, which often leads to poor estimation. Other approaches that extend these methods for motion tracking using a pan/tilt camera include those in [34,33,11]. However, in these cases the camera motion was limited to translation or rotation. When a camera is mounted on a mobile robot, the main motion of the camera is a forward/backward movement, which makes the problem different from that of a pan/tilt camera. There is other research on tracking from a mobile platform with similar motions. [57,7] tracks objects in imagery taken from an airborne, moving platform, and [3,50] track cars in front using a camera mounted on a vehicle driven on a paved road.

Extracting motions from a sequence of images is another challenging issue. The most popular approach is to construct a background model and subtract the model from input images [55,45,8], which is applicable to the case of stationary sensors restrictively. There have been research on maintaining the integrity of the background model even when a sensor moves [37,17, 22], but those solutions still require the assumption of small, pan/tilt camera-type motions. Another well-known technique is to use optical flow [31,4,52]. Motion can be segmented using the characteristics of optical flow vectors in an image sequence, but this method is sensitive to noise and computationally more expensive than the background subtraction method. Other techniques include [53,56]. More general methods for visual people detection are summarized in [40]. The performance of visual detection can be improved by utilizing multiple cameras [1,10], by combining with other types of sensors (e.g. laser range scanner) [43], or by exploiting domain-specific, priori knowledge [30].

Once motions have been identified, objects in the scene need to be tracked over time. The Kalman filter has been extensively used for single motion tracking because of its mathematical soundness and computational efficiency [36,22], but it performs poor when the state variable does not follow Gaussian distribution. The limitation has been overcome by utilizing par-

ticle filter technique as shown in [16,32]. In order to track multiple motions, data association issue needs to be addressed. Joint Probability Data Association Filtering (JPDAF) [35] and Multiple Hypothesis Tracking (MHT) [9] are popular statistical data association techniques. As dedicated mothods for spatial tracking, [46] introduced Bayesian inference method using track graph, and [26] presented a group tracking method when maintaining the state of individual people would be intractable. Recently [29] introduced a learning method to build the dynamic models of moving objects.

3 Ego-motion Compensation

The ego-motion compensation is a coordinate conversion procedure. Assume that a sensory data acquisition process is as follows: (1) one set of data D is acquired at time t when a robot is located at (x, y, α)[1], (2) the robot (and the sensors) moves to $(x+\Delta x, y+\Delta y, \alpha+\Delta\alpha)$ for Δt, and (3) another set of data D' is acquired at time $t+\Delta t$. In this case, the data D and D' cannot be compared directly because they are captured in different coordinate systems. Therefore, the data D should be compensated for the ego-motion $(\Delta x, \Delta y, \Delta\alpha)$, which means that the data D should be transformed as if it were acquired when a robot was located at $(x+\Delta x, y+\Delta y, \alpha+\Delta\alpha)$. The goal of the ego-motion compensation step is to compute this transformation T. The transformation can be estimated *directly* or *indirectly*.

The *indirect* method is to estimate a robot pose each time using various sensors (*eg.* gyroscope, accelerometer, odometer and/or GPS), and compute the ego-motion $(\Delta x, \Delta y, \Delta\alpha)$ first. Once the ego-motion is computed, the transformation T can be computed based on the geometric properties of sensors. The pose estimation technique has been well studied [25,39], and the indirect method may be effective when the transformation error is constrained linearly in the ego-motion estimation error. However, when a projection operation is involved in the transformation, as in our case, the indirect method is not appropriate since a tiny angular error in the motion estimation step would induce a huge position error after being projected into the data space. Therefore, we choose the direct method.

The *direct* method is to infer the transformation T by corresponding salient features in the data set D to those in the data set D'. The feature selection and matching techniques for various types of sensors has been studied by [48,31,12,28,14]. Since the transformation is estimated using the corresponding feature set directly, the quality of the transformation relies on the quality of selected features from the data sets. Unfortunately, in our case, the quality of the features is poor due to independently moving objects in image data. Therefore, a transformation model and outlier detection algorithm needs to be designed so that the estimated transformation is not sensitive to those object motions.

[1] (x, y) represents the 2D position in a global coordinate system, and α represents the heading of a robot.

3.1 Feature Selection and Tracking

Image feature selection and matching technique has been studied for various vision-based applications including object recognition, shape analysis, motion analysis, and stereo matching. Two most successful visual features are corner and SIFT features.

A *corner feature* is characterized by the high intensity changes in both horizontal and vertical directions, and well-known corner detection algorithms are KLT (Kanade-Lucas-Tomasi) corner detector [48,31] and Harris corner detector [15]. Those corner detectors are often used jointly with optical flow techniques to track the corners in a video sequence assuming their displacements are small.

SIFT (Scale Invariant Scale Transform) feature [27, 24] is more advanced visual feature, which is known to be relatively invariant to image translation, scaling and rotation and partially invariant to changes in illumination and local image deformations. The feature is represented as a multi-dimensional vector called a key-point descriptor, and the matching between two features can be performed simply by computing their distance in the multi-dimensional space. However, searching a key-point and creating its descriptor is computationally intensive, and it is not suitable for real-time applications, unless the algorithm is implemented in hardware as in [2,41].

We adopt the KLT feature tracking algorithm [42,5] to select and correspond features between two images mainly due to its computational efficiency. Given two consecutive images (the previous image I^{t-1} and the current I^t), a set of features (F^{t-1}) is selected from the image I^{t-1}, and a corresponding feature set (F^t) is constructed by tracking the same features on the image I^t.

3.2 Transformation Estimation

Once the correspondence $F =< F^{t-1}, F^t >$ is known, the ego-motion of the camera can be estimated using a transformation model and an optimization method. We have studied three different models.

Affine Model :

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\ f_x^{t-1} + a_1\ f_y^{t-1} + a_2 \\ a_3\ f_x^{t-1} + a_4\ f_y^{t-1} + a_5 \end{bmatrix}$$

Bilinear Model :

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\ f_x^{t-1} + a_1\ f_y^{t-1} + a_2 + a_3\ f_x^{t-1}\ f_y^{t-1} \\ a_4\ f_x^{t-1} + a_5\ f_y^{t-1} + a_6 + a_7\ f_x^{t-1}\ f_y^{t-1} \end{bmatrix}$$

Pseudo-perspective Model :

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\ f_x^{t-1} + a_1\ f_y^{t-1} + a_2 + \ a_3\ {f_x^{t-1}}^2 + a_4\ f_x^{t-1}\ f_y^{t-1} \\ a_5\ f_x^{t-1} + a_6\ f_y^{t-1} + a_7 + \ a_4\ f_x^{t-1}\ f_y^{t-1} + a_3\ {f_y^{t-1}}^2 \end{bmatrix}$$

When the interval between consecutive images is very small, most ego-motions of the camera can be estimated using an affine model, which can cover translation, rotation, shearing, and scaling motions. However, when the interval is long, the camera motion in the interval cannot be captured by a simple linear model. For example, when the robot moves forward, the features in the image center move slower that those near the image boundary, which is a projection operation, not a simple scaling. Therefore, a nonlinear transformation model is required for those cases. On the other hand, an overfitting problem may be caused when a model is highly nonlinear, especially when some of the selected features are associated with moving objects (*outliers*). There is clearly a trade-off between a simple, linear model and a highly nonlinear model, and it needs more empirical research for the best selection. We used a bilinear model for the experiments reported in this paper.

When the transformation from the image I^{t-1} to the image I^t is defined as T_{t-1}^t, the cost function for least square optimization is defined as:

$$J = \frac{1}{2} \sum_{i=1}^{N} \left\{ f_i^t - T_{t-1}^t \left(f_i^{t-1} \right) \right\}^2 \tag{1}$$

where N is the number of features. The model parameters for ego-motion compensation are estimated by minimizing the cost. However, as mentioned before, some of the features are associated with moving objects, which lead to the inference of an inaccurate transformation. Those features (outliers) should be eliminated from the feature set before the final transformation is computed. The model parameter estimation is thus performed using the following two-step procedure:

1. compute the initial estimate T_0 using the full feature set F.
2. partition the feature set F into two subsets F_{in} and F_{out} as:

$$\begin{cases} f_i \in F_{in} & \text{if } |f_i^t - T_{0\,t-1}^t(f_i^{t-1})| < \epsilon \\ f_i \in F_{out} & \text{otherwise} \end{cases} \tag{2}$$

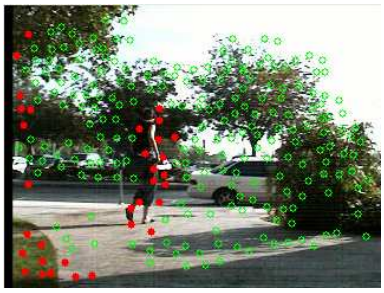


Fig. 3 Outlier feature detection ($\epsilon = 3$).

3. re-compute the final estimate T using the subset F_{in} only.

Figure 3 shows the partitioned feature sets: F_{in} is marked with empty circles, and F_{out} is marked with filled circles. Note that all features associated with the pedestrian are detected as outliers. It is assumed for outlier detection that the portion of moving objects in the images is relatively smaller compared to the background; the features which do not agree with the main motion are considered as outliers. This assumption will break when the moving objects are very close to the camera. However, most of the time, these objects pass by the camera in a short period (leading to transient errors), and a high-level probabilistic filter is able to deal with the errors without total failure.

3.3 Frame Differencing

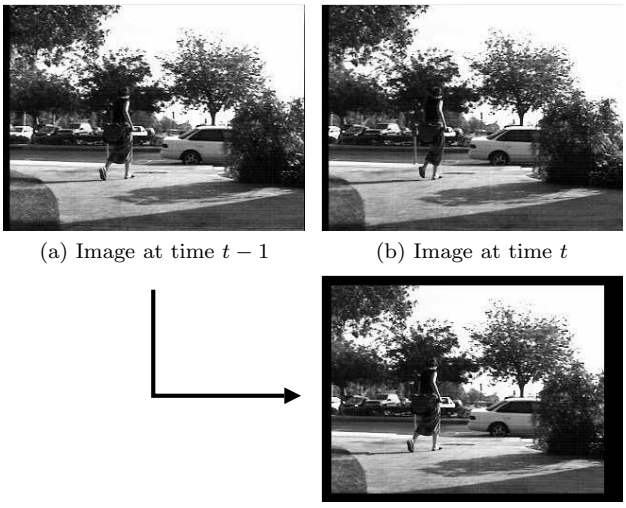
The image I^{t-1} is converted using the transformation model before being compared to the image I^t in order to eliminate the effect of the camera ego-motion. For each pixel (x, y):

$$I_c(x, y) = I^{t-1} \left({T_{t-1}^t}^{-1}(x, y) \right) \tag{3}$$

Figure 4 (c) shows the compensated image of Figure 4 (a); the translational and forward motions of the camera were clearly eliminated. The valid region $\Re$ of the transformed image is smaller than that of the original image because some pixel values on the border are not available in the original image I^{t-1}. The invalid region in Figure 4 (c) is filled black. The difference image between two consecutive images is computed using the compensated image:

$$I_d(x, y) = \begin{cases} |\left(I_c(x, y) - I^t(x, y) \right)| & \text{if } (x, y) \in \Re \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

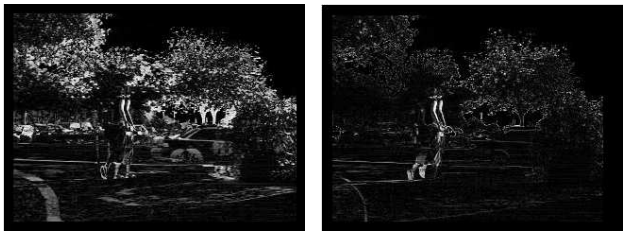
Figure 5 compares the results of two cases: frame differencing without ego-motion compensation (Figure 5 (a)) and with ego-motion compensation (Figure 5 (b)). The results show that the ego-motion of the camera is decomposed and eliminated from image sequences.



(a) Image at time $t-1$ (b) Image at time t

(c) Compensated image of (a)

Fig. 4 Image Transformation.



(a) without compensation (b) with compensation

Fig. 5 Results of frame differencing ($\epsilon = 3, \Delta t = 0.2$).

4 Motion Detection in 2D Image Space [2]

The *Frame Differencing* step in Figure 2 generates the sequence of difference images, $I_d^0, I_d^1, \cdots, I_d^t$, whose pixel values represent the amount of motion occurred in the position. However, as described earlier, the difference images contain two different types of errors. There are transient errors caused by imperfect ego-motion compensation, and this type of error should be filtered out using their temporal properties. There are also persistent errors caused during data acquisition. Since the camera positions are different when two consecutive images are captured, it is inevitable that some information is newly introduced to the current image I^t or some information of the previous image I^{t-1} is occluded in the image I^t.

To deal with those errors, a probabilistic approach is adopted. The normalized pixel values in the difference images can be interpreted as the probability of the existence of moving objects in that position, and the position and size of the moving objects are estimated over time. This estimation process can be modeled using a Bayesian formulation. Let $\mathbf{x}^t$ represent the state (*eg.* the position and velocity) of a moving object (a motion blob in images).

$$\mathbf{x} = [\ x\ \ y\ \ \dot{x}\ \ \dot{y}\ ]^T \tag{5}$$

The posterior probability distribution $P_m(\mathbf{x}^t)$ of the state is derived as follows.[3]

$$\begin{aligned} P_m(\mathbf{x}^t) &= P(\mathbf{x}^t | I_d^0, I_d^1, \cdots, I_d^t) \\ &= \eta^t\ P(I_d^t|\mathbf{x}^t) \int P(\mathbf{x}^t|\mathbf{x}^{t-1}) P_m(\mathbf{x}^{t-1})\ d\mathbf{x}^{t-1} \end{aligned} \tag{6}$$

Now the posterior probability distribution can be updated recursively by applying a perception model $P(I_d^t|\mathbf{x}^t)$ and a motion model $P(\mathbf{x}^t|\mathbf{x}^{t-1})$ over time.

4.1 Bayesian Filter Design

Equation 6 shows how the sequence of difference images and the motion model of a moving object are integrated into the state estimation process. However, there are still three questions to answer: (1) how to define a perception model, and (2) how to define a motion model, and finally (3) how to represent the posterior probability distribution.

The perception model $P(I_d^t|\mathbf{x}^t)$ captures the idea that if there is a motion at position $\mathbf{x}^t$, then the difference values of the pixel in that position and its neighbors should be big. For example, let us assume a small moving object that occupies a single pixel $\mathbf{p}^{t-1}$ on a camera image. When the object moves to its neighbor pixel $\mathbf{p}^t$, then the difference values of both pixels $\mathbf{p}^{t-1}$ and $\mathbf{p}^t$ would be big. This *neighborhood* can be modeled using a multi-variate Gaussian, and the perception model can be defined as

$$P(I_d^t\,|\,\mathbf{x}^t) = \int I_d^t(\mathbf{x}) \times \frac{1}{\sqrt{(2\pi)^d |\Sigma_s|}}\, e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x^t})^T \Sigma_s^{-1} (\mathbf{x}-\mathbf{x^t})}\, d\mathbf{x} \tag{7}$$

where d is the dimension of the state $\mathbf{x}$, and the covariance matrix Σ_s controls the range of effective neighborhood, which is determined empirically.

The motion model $P(\mathbf{x}^t|\mathbf{x}^{t-1})$ captures the best guess about the motion of a moving object. Since no prior knowledge of an object motion is assumed, a constant velocity model is a natural choice. The uncertainty of an object motion is modeled by the covariance matrix Σ_m of a multi-variate Gaussian.

$$\mu = \begin{bmatrix} x^{t-1} + \Delta t \times \dot{x}^{t-1} \\ y^{t-1} + \Delta t \times \dot{y}^{t-1} \\ \dot{x}^{t-1} \\ \dot{y}^{t-1} \end{bmatrix} \tag{8}$$

[2] More detailed description including all algorithms and configurations can be found in [20]

[3] η^t in the derivation process is a normalizer.

$$P(\mathbf{x}^t \,|\, \mathbf{x}^{t-1}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_m|}} \, e^{-\frac{1}{2}(\mathbf{x^t}-\mu)^T \Sigma_m^{-1} (\mathbf{x^t}-\mu)} \quad (9)$$

The choice of a representation for the posterior probability distribution is important for real-time system design because the update equation (Equation 6) contains an integral operation and the required computation is intensive. Even when the size of a camera image is small, the state space is sizeable because the state is four-dimensional. The most compact representation is to use a single Gaussian, like the Kalman filter [54,23], but this approach is not appropriate because (1) the initial state of a moving object is not given in priori, and (2) image segmentation is avoided for real-time response, which makes it hard to construct a measurement matrix. In this paper, we adopt the sample-based representation.

4.2 Particle Filter Design

The *Particle filter* [19,47] is a simple but effective algorithm to estimate the posterior probability distribution recursively, which is appropriate for real-time applications. In addition, its ability to perform multi-modal tracking is attractive for unsegmented object detection and tracking from camera images. An efficient variant, called the *Adaptive Particle Filter*, was introduced in [13]. This changes the number of particles dynamically for a more efficient implementation. We implemented the *Adaptive Particle Filter* to estimate the posterior probability distribution in Equation 6. As described in Section 4.1, particle filters also require two models for the estimation process: a perception model and a motion model.

The perception model is used to evaluate a particle and compute its weight (or importance). Equation 7 provides a generic form of the perception model. However, the perception model is simplified for computational efficiency. The ramification of the simplification needs to be studied further quantitatively. A step function is used instead of a multi-variate Gaussian, and the evaluation range is also limited to $m \times m$ fixed area. The $m \times m$ mask should be big enough (usually 5×5) so that salt-and-pepper noise is eliminated. The weight ω_i^t of the i_{th} particle ($s_i^t = [\ x_i^t \ \ y_i^t \ \ \dot{x}_i^t \ \ \dot{y}_i^t\ ]^T$) is computed by

$$\omega_i^t = \frac{1}{m^2} \sum_{j=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} I_d\left(x_i^t - j, y_i^t - k\right) \quad (10)$$

As shown in Equation 10, only the position information is used to evaluate particles.

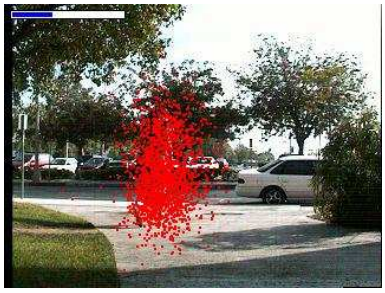


Fig. 6 Particle filter tracking.

The motion model is used to propagate a newly drawn particle according to the estimated motion of a moving object. The motion model in and Equation 9 describes how to compute the probability of the new state $\mathbf{x^t}$ when the previous state $\mathbf{x^{t-1}}$ is given. However, for particle filter update, the motion model should describe how to draw a new particle s_i^t when the previous particle s_i^{t-1} and its weight ω_i^t are given. Therefore, the motion model is defined as

$$s_i^t = \begin{bmatrix} x_i^{t-1} + \Delta t \times \dot{x}_i^{t-1} + Normal(\frac{\gamma_p}{\omega_i^t}) \\ y_i^{t-1} + \Delta t \times \dot{y}_i^{t-1} + Normal(\frac{\gamma_p}{\omega_i^t}) \\ \dot{x}_i^{t-1} + Normal(\frac{\gamma_v}{\omega_i^t}) \\ \dot{y}_i^{t-1} + Normal(\frac{\gamma_v}{\omega_i^t}) \end{bmatrix} \quad (11)$$

where Δt is a time interval, and γ_p and γ_v are noise parameters for position and velocity components respectively. The function $Normal(\sigma)$ generate a Gaussian random variate with zero mean and the standard deviation σ. As shown in Equation 11, the parameterized noise is added to the constant-velocity model in order to overcome an intrinsic limitation of the particle filter, which is that all particles move in a convergence direction. However, a dynamic mixture of divergence and convergence is required to detect newly introduced moving objects. [47] introduced a mixture model to solve this problem, but in the image space the probability $P(\mathbf{x}^t|I_d^t)$ is uniform and the dual MCL becomes random. Therefore, we used a simpler, but effective method by adding inverse-proportional noise.[4]

Figure 6 shows the output of the particle filter. The dots represent the position of particles, and the horizontal bar on the top-left corner of the image shows the number of particles being used at the moment[5].

[4] Empirically it has been proven that the added noise helps detecting a new object faster; however, the effect of the noise term on the convergence properties of the filter needs to be studied further.

[5] The range of particles is set to 1000∼5000 for all experiments.

Fig. 7 Particle clustering.

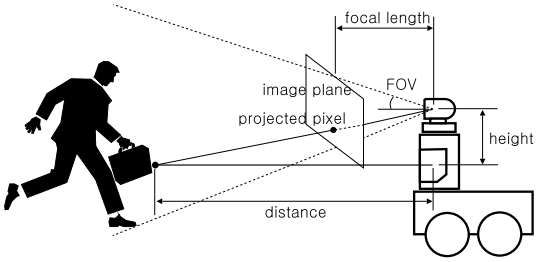


Fig. 8 Projection of laser scans.

4.3 Particle Clustering

The particle filter generates a set of weighted particles that estimate the posterior probability distribution of a moving object, but the set of particles are not easy to process in the following step. More intuitive and meaningful data can be extracted by clustering the particles. A density-based algorithm using a kd-tree is introduced for efficient particle clustering. The main idea is to convert a set of weighted particles into a lower-resolution, uniform-sized grid. The grids can be represented using a kd-tree efficiently, and all clustering operations are performed using the grids instead of particles. Therefore, the required computation is reduced drastically. However, since each grid maintains enough information about the particles in the grid, the statistics of each cluster can be calculated without any accuracy loss.

Figure 7 shows the output of the particle clustering algorithm. The dots represent the position of particles, and the ellipsoid represents the mean and covariance of each cluster.

4.4 Multiple Particle Filters for Multiple Motion Tracking

The particle filter has many advantages as described in [21,47]; one of the advantages is multi-modality. This property is attractive for multi-target tracking because it raises the possibility that a single set of particles can track multiple objects in an image sequence. However, the particle filter is known to be poor at consistently maintaining the multi-modality of the target distributions in some cases [51]. Its multi-modality property is valid only under the following two conditions:

1. *The perception model should be "bad" enough* so that particles converge slowly, and eventually stay on multiple objects.
2. *All objects should be introduced in the beginning of estimation process.* As explained in Section 4.1, particles shows a convergence tendency, and consequently converged particles do not diverge unless the tracked object disappears.

Since those conditions are not valid for the tracking problem, we introduce a tracking system using multiple particle filters. The main idea is to maintain an extra particle filter for a newly introduced or detected object. Since the number of objects is not known in priori, particle filters should be created and destroyed dynamically. Whenever the extra particle filter converges on a newly detected object, a new particle filter is created (as long as the number of particle filters is smaller than the maximum limit N_{max}). Similarly, whenever a particle filter diverges due to the disappearance of a tracked object, it is destroyed. In order to prevent two particle filters from converging on the same object, whenever a particle filter is updated, the difference image is modified for subsequent processing such that difference values covered by the filter are cleared.

5 Position Estimation in 3D Space

A monocular image provides rich information for egomotion compensation and motion tracking in 2D image space. However, a single camera has limits on retrieving depth information, and an additional sensor is required to construct 3D models of moving objects. Our robots are equipped with a laser rangefinder, which provides depth information within a singe plane. Given the optical properties of a camera and the transformation between the camera and the laser rangefinder, distance information from the laser rangefinder can be projected onto the image coordinates (Figure 8).

Given the heading α and the range r of a scan, the projected position (x, y) in the image coordinate system is computed as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{w}{2} \times \left(1 - \frac{\tan(\alpha)}{\tan(f_h)}\right) \\ \frac{h}{2} \times \left(1 + \left(d - \frac{d}{r} \times (r - l)\right) \times \frac{1}{l \times \tan(f_v)}\right) \end{bmatrix} \tag{12}$$

where the focal length of the camera is l, the horizontal and vertical field of view of the camera are f_h and

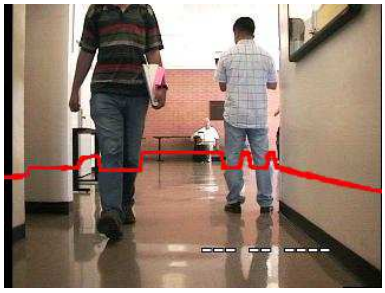


Fig. 9 Projected laser scans.

f_v, the height from the laser rangefinder to the camera is d, and the image size is $w \times h$. This projection model assumes a very simple camera model (a pin-hole camera) for fast computation. As a result of the projection, the image pixels at the same height as the laser rangefinder will have depth information as shown in Figure 9. For ground robots, this partial 3D information can be enough for safe navigation assuming all moving obstacles are on the the same plane as the robot. In terms of moving object tracking, if the region of a moving objects in image space and those pixels are overlapped, then the distance between a robot and the moving object can be estimated.

This naive integration of the 2D motion estimates and range scans from a laser rangefinder is a reasonable practical solution. However, using two separate sensors requires another estimation problem potentially, which is the fusion of multiple asynchronous inputs. A preferred route (not investigated here) would be to use stereo vision for depth information retrieval. If computational power allows one can exploit the facts that stereo (1) provides full depth information of an image space, and (2) a single input source provides synchronous data and better fusion result can be expected.

6 Experimental Results and Discussion

The proposed motion tracking system was tested in various scenarios. First, the robustness of ego-motion compensation and motion tracking algorithms was tested using three different robot platforms. The performance is analyzed in Section 6.1. Second, the multiple-motion tracking system described in Section 4.4 was tested using various scenarios. The results are presented in Section 6.2. Finally, the tracking system was integrated with an actual robot control system. The result is discussed in Section 6.3.[6]

[6] The first and the second experiments were conducted using a monocular camera only. The laser data fusion described in Section 5 is applied to the last experiment only.

6.1 Tracking a Moving Object from Various Platforms

The ego-motion of a mobile robot is diverse according to its actuator design and the way the camera is mounted on the platform. For example, a down-facing camera mounted on an UAV (Unmanned Aerial Vehicle) would show a different ego-motion from a forward-facing camera mounted on a walking robot. In addition, the complexity of the ego-motion increases through the interaction with rough terrain. The tracking performance is also affected by the distribution of occlusive obstacles in an environment. Therefore, the ego-motion compensation and the motion tracking algorithms should be tested on various environments with a wide variety of mobile platforms.

6.1.1 Experimental Setup

The tracking algorithms were implemented and tested in various outdoor environments using three different robot platforms: robotic helicopter, Segway RMP, and Pioneer2 AT. Each platform has unique characteristics in terms of its ego-motion[7].

The *Robotic Helicopter* [38] in Figure 10 (a) is an autonomous flying vehicle carrying a monocular camera facing downward. Once it takes off and hovers, planar movements are the main motion, and moving objects on the ground stay at a roughly constant distance from the camera most of the time; however, pitch and roll motions still generate complicate video sequences. Also, high-frequency vibration of the engine adds motion-blur to camera images.

The *Segway RMP* in Figure 10 (b) is a two-wheeled, dynamically stable robot with self-balancing capability. It works like an inverted pendulum; the wheels are driven in the direction that the upper part of the robot is falling, which means the robot body pitches whenever it moves. Especially when the robot accelerates or decelerates, the pitch angle increases by a significant amount. Since all sensors are directly mounted on the platform, the pitch motions prevent direct image processing. Therefore, the ego-motion compensation step should be able to cope with not only planar movements but also pitch motions.

The *Pioneer2 AT* in Figure 10 (c) is a typical four-wheeled, statically stable robot. Since the Pioneer2 robot is the only statically stable platform among these robot platforms, we drove the robot on the most severe test environment. Figure 13 shows the rocky terrain where the robot was driven. In addition, the moving objects

[7] All input & output videos are available on *http://robotics.usc.edu/∼boyoon/mt.html*.


(a) Robotic Helicopter


(b) Segway RMP


(c) Pioneer2 AT

Fig. 10 Robot platforms for experiments.

were occluded occasionally because of the trees in the environment.

The computation was performed on embedded computers (Pentium III 1.0 GHz) on the robots. Low resolution (320x240 pixels) input images were chosen for real-time response. The maximum number of particles was set to 5,000, and the minimum number of particles was set to 1000. Since the algorithm is supposed to run in parallel with other processes (eg. navigation and communication), less than 70 percent of the CPU time was dedicated for tracking; the tracking algorithm was able to process five frames per second.

6.1.2 Experimental Results

The performance of the tracking algorithm was evaluated by comparing with the positions of manually tracked objects. For each video sequence, the rectangular region of moving objects were marked manually and used as ground truth. Figure 11–13 show this evaluation process. The upper rows show the input image sequence, and the positions of manually-tracked objects are marked with rectangles. The lower rows show the set of particles and the clustering results. The position of each particle is marked with dots, and the horizontal bar on the top-left corner of the image indicates the number of particles being used. The clustering result is represented using an ellipsoid and a line inside. The ellipsoid shows the mean and covariance of the estimated object position, and the line inside of the ellipsoid represents the estimated velocity vector of the object.

The final evaluation result is shown in Table 1. *Frames* is the number of image frames in a video sequence, and *Motions* is the number of moving objects. *Detected* is the total number of detected objects, and *True +* and *False +* are the number of correct detections and the number of false-positives respectively. *Detection Rate* shows the percentage of moving objects correctly detected, and *Avg. Error* is the average Euclidean distance in pixels between the ground truth and the output of tracking algorithm. The average distance error should not be considered as actual error measurement since the tracking algorithm does not perform an explicit object segmentation; it may track a part of an object that generates motion while the ground truth always tracks the whole objects even though only part of the object moves.

The *Robotic helicopter* result shows that the tracking algorithm missed seven objects, but five of them were the cases when a moving object was introduced and showed only partially on the boundary of the image plane. Once the whole object entered into the field of view of the camera, the tracking algorithm tracked it robustly. For the *Segway RMP* result, the detection rate was satisfactory, but the average distance error was larger than the others. The reason was that the walking person was closer to the robot and the tracking algorithm often detected the upper body only, which caused a constant distance error. The *Pioneer2 AT* result shows the higher ratio of false-positives; however, as explained in the previous section, the terrain for the experiment was more challenging (rocky) and the input images were more blurred and unstable. Overall various types of ego-motions were successfully eliminated from input images, and the particle filter was able to track motions robustly from diverse robot platforms.

6.2 Tracking Multiple Moving Objects

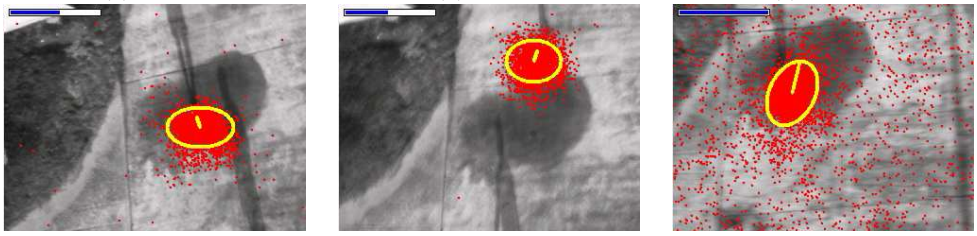
The multiple-motion tracking system using multiple particle filters was introduced in Section 4.4. Since the robustness of an individual filter was analyzed in Section 6.1, we focus on analyzing how multiple filters are created and destroyed effectively when the number of moving objects changes dynamically.

6.2.1 Experimental Setup

The *Segway RMP* in Figure 10 (b) was selected for the experiment because of its complex ego-motion. The *Segway RMP* is a dynamically stable platform, and its

[8] The actual frame rate achieved on the Pentium III (1.0 GHz) computer was roughly 5 Hz.

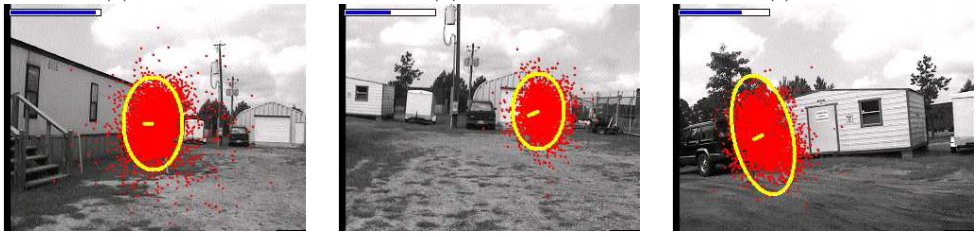
(a) $t = 13$ (b) $t = 21$ (c) $t = 34$



(d) $t = 13$ (e) $t = 21$ (f) $t = 34$

Fig. 11 Moving object tracking from Robotic helicopter.

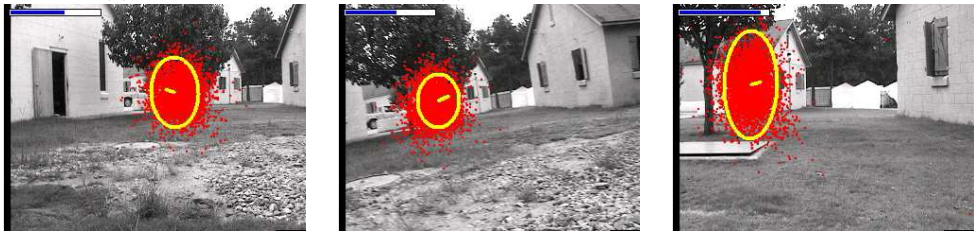
(a) $t = 57$ (b) $t = 119$ (c) $t = 195$



(d) $t = 57$ (e) $t = 119$ (f) $t = 195$

Fig. 12 Moving object tracking from Segway RMP.

(a) $t = 35$ (b) $t = 56$ (c) $t = 191$



(d) $t = 35$ (e) $t = 56$ (f) $t = 191$

Fig. 13 Moving object tracking from Pioneer2 AT.

Table 1 Performance of moving object detection algorithm

Platform	Frames[8]	Motions	Detected	True +	False +	Detection Rate	Avg. Error
Robotic helicopter	43	35	28	28	0	80.00 %	11.90
Segway RMP	230	220	215	211	4	95.90 %	21.31
Pioneer2 AT	195	172	158	146	12	84.88 %	15.87

pitching motions for self-balancing are combined into linear and angular motions. The combination of these motions causes complicated ego-motions even when the robot is driven on a flat terrain or when the robot stops in place. The robot was driven on the USC campus during the daytime when there are diverse activities in the environment including walking people and automobiles.

The tracking performance is analyzed for three different cases. As explained in Section 4.4, if one of the following two conditions is not satisfied, a single particle filter fails to track multiple objects even though it supports multi-modality in theory: (1) all objects should be introduced before a particle filter converges, and (2) the convergence speed of a particle filter should be sacrificed by using a *"bad"* perception model. The first two cases are when one or both conditions can not be satisfied. In the first case, there are three people walking by, but the people are introduced in the input image sequence one by one, which violates the first condition. In the second case, there are two groups of automobiles passing by, and they are introduced sequentially with a big time interval between them. In addition, the automobiles move fast enough so that the convergence speed of a particle filter cannot be sacrificed, which violates the second condition. The results for both cases show how the multiple particle filter approach overcomes the limitation of a single particle filter. The stability of this approach is also clear. In the last case, it is observed how multiple particle filters behave when two people walk in different directions and intersect in the middle.

The computation was performed on a Pentium IV (2.1 GHz) computer, and the image resolution was fixed to 320x240 pixels. The maximum number of particle filters was fixed to five, and for an individual particle filter, the range of the number of particles was set to $(1000 \sim 5000)$. The number of frames processed per second varies based on how many particle filters have been created, but roughly 10 frames were able to be processed.

6.2.2 Experimental Results

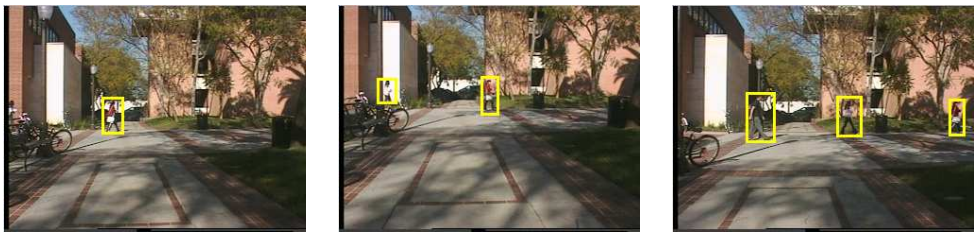
The snapshots of the multiple particle filter tracking multiple moving objects are shown in Figure 14–16. The upper rows of the figures show input image sequences and manually-tracked moving objects in the images. The manually-tracked objects are marked with rectangles. The lower rows show particle filters and the covered area (the minimum rectangular region enclosing each ellipsoid that is generated by the particle clustering algorithm) by each particle filter. Only converged particle filter is visualized on the images. Each particle filter is drawn with different colored dots, and the covered areas are marked with rectangles.

The experimental result of the first case is shown in Figure 14. The estimation process starts with a single particle filter. When the first person enters into the field of view of the camera as in Figure 14 (a), the particle filter converges and starts to track the person as in Figure 14 (d), and a new particle filter is created to explore the remained area. When the second person enters as in Figure 14 (b), the new particle filter converges and starts to track the second person as in Figure 14 (e), and another particle filter is created. This process is repeated whenever a new object is introduced. At the end when three people are in the input image as in Figure 14 (c), the total number of particle filters becomes four; three filters for people and one extra filter to explore.

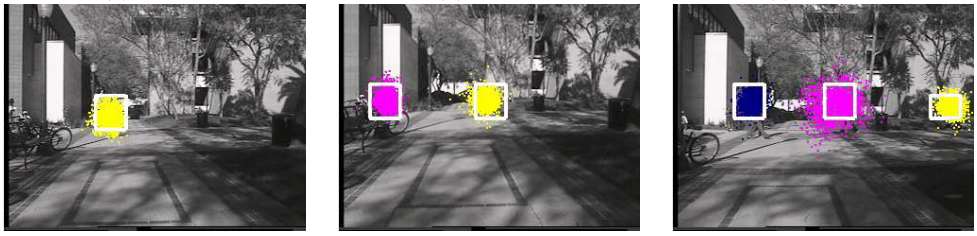
Figure 15 shows the experimental result of the second case. The estimation process is performed in the same way with the first case. Whenever a new automobile is introduced, a new particle filter is created. When the automobile leaves from the field of view of the camera, the particle filter that tracks the automobile diverges and is destroyed eventually.

The experimental result of the third case is shown in Figure 16. There are two people walking in different directions as in Figure 16 (a), and two particle filters are created to track them individually as in Figure 16 (d). The pedestrians intersect in the middle, and keep walking in each direction. Figure 16 (e) and (f) demonstrate that the particle filters track them successfully without being confused by the intersection.

The detailed evaluation result is shown in Table 2. There were untracked motions in common; however, it happens only right after a new object is introduced and before a filter converges on the object. Once a particle filter converges and is associated with the object, it never fails to track the object. For the second case, the detection rate is lower than the other two cases. This is reasonable since the motion of an automobile is much

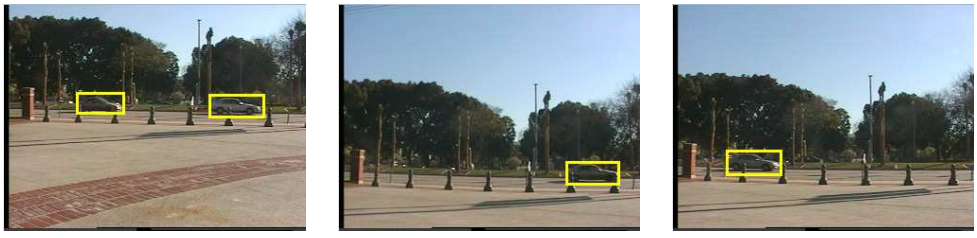


(a) $t = 49$ (b) $t = 95$ (c) $t = 132$

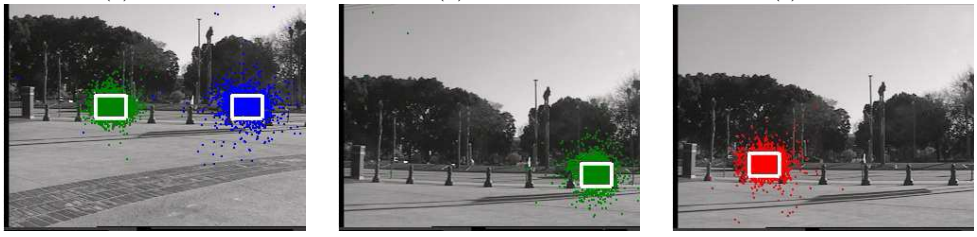


(d) $t = 49$ (e) $t = 95$ (f) $t = 132$

Fig. 14 Tracking people.

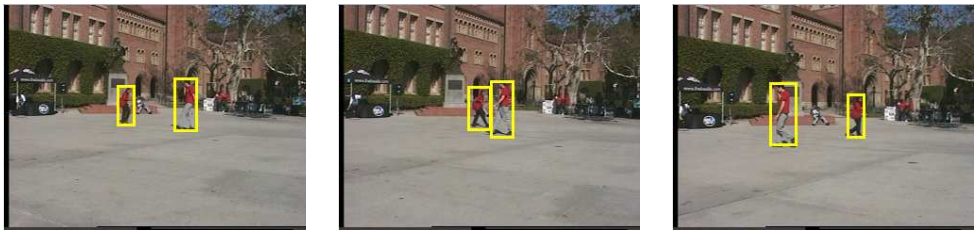


(a) $t = 51$ (b) $t = 66$ (c) $t = 85$

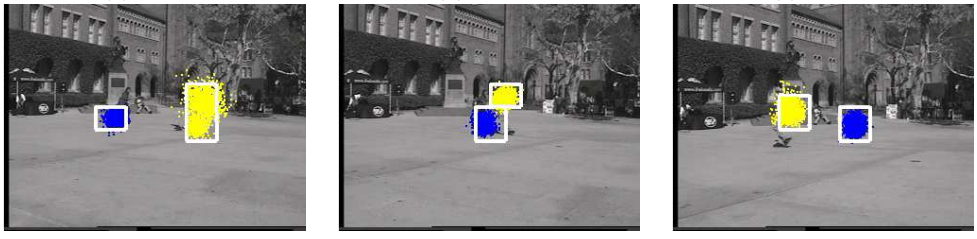


(d) $t = 51$ (e) $t = 66$ (f) $t = 85$

Fig. 15 Tracking automobiles.



(a) $t = 27$ (b) $t = 38$ (c) $t = 67$



(d) $t = 27$ (e) $t = 38$ (f) $t = 67$

Fig. 16 Intersectional particles.

faster than that of a person, and it took longer for a particle filter to converge on it. The false-positives observed in the second case are also related to the faster motion. When an automobile leaves from the camera field of view, it disappears quickly enough so that the particle filter stays converged for one or two frames. In general, the multiple particle filter approach shows stable performance for all cases.

6.3 Close the loop: Following a Moving Object

The proposed tracking system is integrated with a robot control loop, and its robustness and real-time capability are tested. For this experiment, the task of a robot is to wait for a moving object to appear and follow the object.

6.3.1 System Design and Implementation

A robot needs two capabilities to accomplish the task: motion detection and tracking, and local navigation. For motion detection and tracking, the system described in Section 3-5 is utilized. This module takes the sequence of camera images and the laser range scans as inputs, and computes the existence of moving object(s) and the estimation of target positions in the robot's local coordinates. For local navigation, VFH+ (Vector Field Histogram +) [49] algorithm is implemented. Internally, VFH+ algorithm performs two tasks: (1) it retrieves range scans from a laser rangefinder, and build a local occupancy grid map for obstacle avoidance, and (2) when the estimated target position is given, the algorithm generates both translational and rotational motor commands for point-to-point navigation.

The system architecture is presented in Figure 17. The implemented system was deployed on a Segway RMP robot. The computation was performed on an embedded computer (Athlon 1.0 GHz) on the robot, and the image resolution was fixed at 320x240 pixels.

6.3.2 Experimental Result

The snapshots of the robot following a person are shown in Figure 18. When the person entered into the field of view of the camera as in Figure 18 (a), the particle filter converged on him, and the Segway started to follow it. As shown in Figure 18 (b)-(g), there were automobiles, a golf car, and pedestrians passing by in the background. However, the tracking system was not confused by them; the Segway was able to track the person without a single failure. When the person stopped and stood still as in Figure 18 (i), the particle filter diverged, which made the robot stop. However, when the person re-started to walk as in Figure 18 (j), the particle filter was able to detect the motion again, and the robot re-started to follow the person.

[9] The actual frame rate achieved on the Pentium IV (2.1 GHz) computer was roughly 10 Hz.

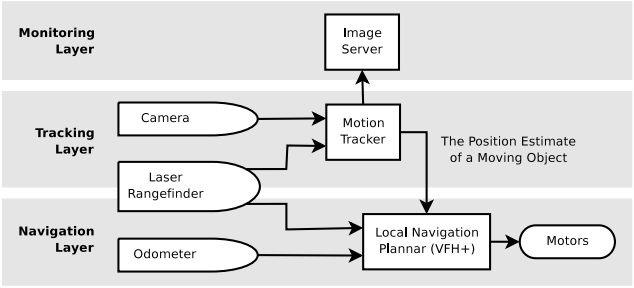


Fig. 17 Control architecture for motion following system.

7 Conclusion

We have presented a set of algorithms for multiple motion tracking from a mobile robot in real-time. There are three challenges: *1) Compensation for the robot ego-motion:* The ego-motion of the robot was directly measured using corresponding feature sets in two consecutive images obtained from a camera rigidly attached to the robot. In order to eliminate the unfavorable effect of a moving object in the image sequence, an outlier detection algorithm has been proposed. *2) Transient and structural noise:* An adaptive particle filter has been designed for robust motion tracking. The position and velocity of a moving object were estimated by combining the perception model and the motion model incrementally. Also, the multiple target tracking system has been designed using multiple particle filters. *3) Sensor fusion:* The depth information from a laser rangefinder was projected into the image space, and the partial 3D position information was constructed in the region of overlap between the range data and the image data.

The proposed algorithms have been tested with various configurations in outdoor environments. First, the algorithms were deployed on three different platforms (*Robotic Helicopter*, *Segway RMP*, and *Pioneer2 AT*), and tested in different environments. The experimental results showed that various type of ego-motions were successfully eliminated from input images, and the particle filter was able to track motions robustly. Second, the multiple target tracking algorithm was tested for different types of motions. The experimental results show that multiple particle filters are created and destroyed dynamically to track multiple targets introduced at different times. Lastly, the tracking algorithm was in-

Table 2 Performance of moving object detection algorithm

Case	Frames[9]	Motions	Detected	True +	False +	Detection Rate	Avg. Error
(1) Pedestrians	141	285	274	274	0	96.14 %	8.68
(2) Automobiles	123	120	95	92	3	76.67 %	20.40
(3) Intersection	81	162	156	156	0	96.30 %	12.34

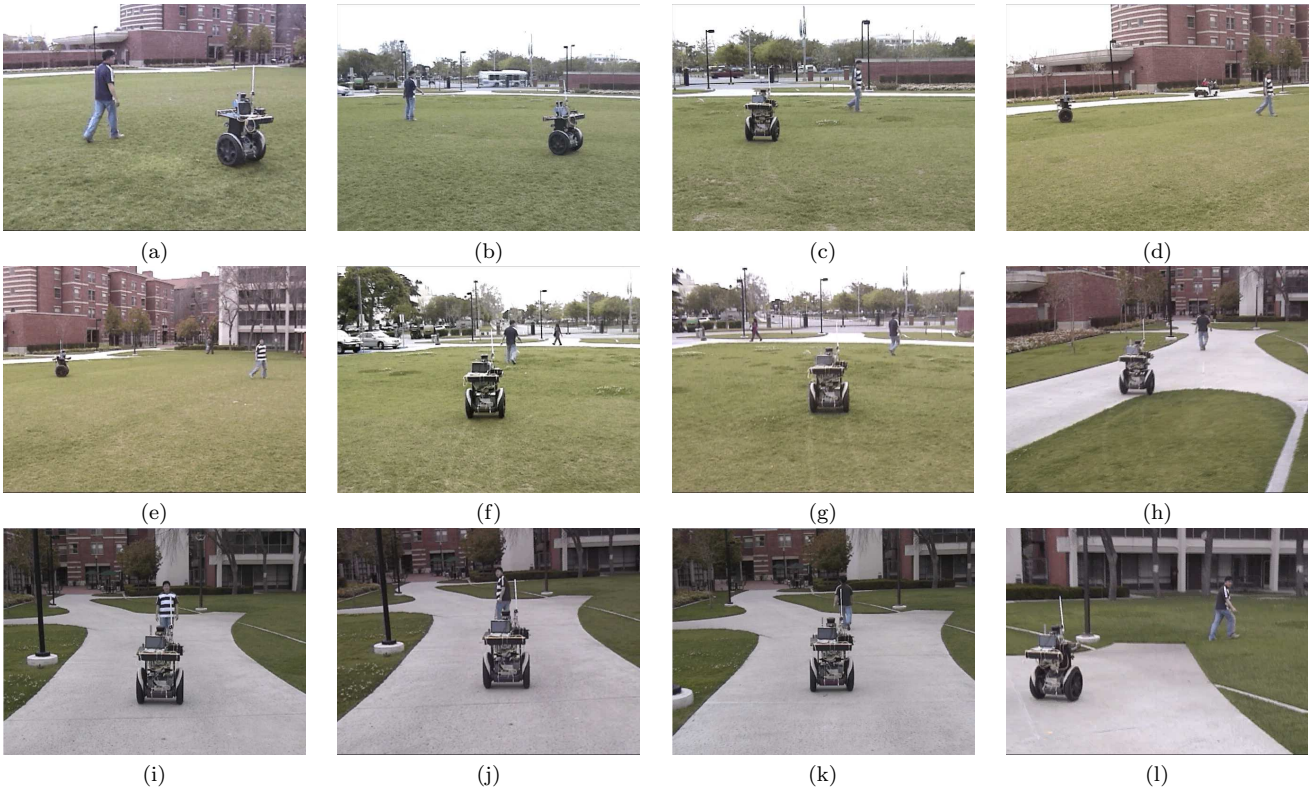


Fig. 18 Snapshots of a Segway RMP robot following a person

tegrated with a robot control loop to test its real-time capability, and the task of following a moving object was successfully accomplished.

The proposed algorithms are expected to be utilized in various application domains as a key enabler. *Localization and mapping* problems have been studied actively by the mobile robotics community, and there are many well-developed techniques widely used. However, most techniques assume a static environment, and their performance is degraded significantly when there are dynamic objects in an environment. Our algorithm provides a robust method to detect dynamic objects in an environment, and it can be exploited in a preprocessing step to filter out data that are associated with the dynamic objects. *Safe navigation* is another fundamental problem in mobile robotics, and most solutions generate motion commands based on local positions of obstacles. However, those solutions become unreliable when obstacles are dynamic, especially when the obstacles move faster than the robot. In this case, a mobile robot needs to predict obstacle position in the near future to avoid collision. The motion velocity (speed and direction) estimation capability of our algorithm could fulfill this requirement. *Human-robot interaction* is active research area in service robot applications, and locating a subject to interact with is a key problem. Our algorithm is applicable in this regard. Needless to say, *surveillance and security* applications could make use of our algorithms to detect and track moving targets.

Acknowledgement

This work is supported in part by DARPA grants DABT63-99-1-0015, and 5-39509-A (via UPenn) under the Mobile Autonomous Robot Software (MARS) program, DOE RIM under grant DE-FG03-01ER45905, and NSF CAREER grant IIS-0133947.

References

1. Max Bajracharya, Baback Moghaddam, Andrew Howard, Shane Brennan, and Larry H. Matthies. Results from a real-time stereo-based pedestrian detection system on a moving vehicle. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
2. Timothy D. Barfoot. Online visual motion estimation using FastSLAM with SIFT features. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 579–585, Alberta, Canada, August 2005.
3. Alireza Behrad, Ali Shahrokni, and Seyed Ahmad Motamedi. A robust vision-based moving target detection and tracking system. In *the Proceeding of Image and Vision Computing Conference*, University of Otago, Dunedin, New Zealand, November 2001.
4. Michael J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.
5. Jean-Yves Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker: Description of the algorithm. Technical report, Intel Research Laboratory, 1999.
6. Alberto Censi, Andrea Fusiello, and Vito Roberto. Image stabilization by features tracking. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, pages 665–667, Venice, Italy, September 1999.
7. Isaac Cohen and Gerard Medioni. Detecting and tracking objects in video surveillance. In *Proceeding of the IEEE Computer Vision and Pattern Recognition 99*, pages 319–325, Fort Collins, June 1999.
8. R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. In *Proceedings of the IEEE*, volume 89, pages 1456–1477, October 2001.
9. Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, February 1996.
10. A. Ess, K. Schindler, B. Leibe, and L. van Gool. Improved multi-person tracking with active occlusion handling. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
11. Gian Luca Foresti and Christian Micheloni. A robust feature tracker for active surveillance of outdoor scenes. *Electronic Letters on Computer Vision and Image Analysis*, 1(1):21–34, 2003.
12. David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.
13. Dieter Fox. KLD-sampling: Adaptive particle filter. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.
14. Jens-Steffen Gutmann and Christian Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, pages 61–67, 1996.
15. Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151, Manchester, 1988.
16. Carine Hue, Jean-Pierre Le Cadre, and Patrick Pérez. A particle filter to track multiple objects. In *IEEE Workshop on Multi-Object Tracking*, pages 61–68, Vancouver, Canada, July 2001.
17. Michal Irani and P. Anandan. Video indexing based on mosaic representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 86(5):905–921, 1998.
18. Michal Irani, Renny Rousso, and Shmuel Peleg. Recovery of ego-motion using image stabilization. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, pages 454–460, March 1994.
19. Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
20. Boyoon Jung. *Cooperative Target Tracking using Mobile Robots*. PhD thesis, University of Southern California, Los Angeles, CA, 2005.
21. Boyoon Jung and Gaurav S. Sukhatme. Detecting moving objects using a single camera on a mobile robot in an outdoor environment. In *International Conference on Intelligent Autonomous Systems*, pages 980–987, The Netherlands, March 2004.
22. Jinman Kang, Isaac Cohen, and Gerard Medioni. Continuous multi-views tracking using tensor voting. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, pages 181–186, Orlando, Florida, December 2002.
23. Jinman Kang, Isaac Cohen, and Gerard Medioni. Continuous tracking within and across camera streams. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 267–272, Madison, Wisconsin, June 2003.
24. Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceeding of the IEEE Computer Vision and Pattern Recognition*, pages 506–513, Washington, DC, June 2004.
25. Alonzo Kelly. A 3D state space formulation of a navigation Kalman filter for autonomous vehicles. Technical Report CMU-RI-TR-94-19, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, May 1994.
26. Boris Lau, Kai O. Arras, and Wolfram Burgard. Multi-modal hypothesis group tracking and group size estimation. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
27. David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
28. Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.
29. Matthias Luber, Kai O. Arras, Christian Plagemann, and Wolfram Burgard. Classifying dynamic objects: An unsupervised learning approach. *Autonomous Robotis*, 26(2–3):141–151, 2009.
30. Matthias Luber, Gian Diego Tipaldi, and Kai O. Arras. Spatially grounded multi-hypothesis tracking of people. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
31. Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–697, 1981.
32. John Maccormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, 2000.
33. Don Murray and Anup Basu. Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):449–459, May 1994.
34. Peter Nordlund and Tomas Uhlin. Closing the loop: Detection and pursuit of a moving object by a moving observer. *Image and Vision Computing*, 14:265–275, May 1996.
35. Christopher Rasmussen and Gregory D. Hager. Probabilistic data association methods for tracking complex visual objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):560–576, 2001.

36. Stan Sclaroff Roḿer Rosales. 3d trajectory recovery for tracking multiple objects and trajectory guided recognition of actions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–123, 1999.
37. Simon Rowe and Andrew Blake. Statistical mosaics for tracking. *J. Image and Vision Computing*, 14:549–564, 1996.
38. Srikanth Saripalli, James F. Montgomery, and Gaurav S. Sukhatme. Visually-guided landing of an unmanned aerial vehicle. *IEEE Transactions on Robotics and Automation*, 19(3):371–381, June 2003.
39. Srikanth Saripalli, Jonathan M. Roberts, Peter I. Corke, Gregg Buskey, and Gaurav S. Sukhatme. A tale of two helicopters. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 805–810, October 2003.
40. Bernt Schiele, Mykhaylo Andriluka, Nikodem Majer, Stefan roth, and Christian Wojek. Visual people detection - different models, comparison and discussion. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
41. Stephen Se, Timothy Barfoot, and Piotr Jasiobedzki. Visual motion estimation and terrain modeling for planetary rovers. In *IEEE International Conference on Robotics and Automation*, pages 2051–2058, South Korea, May 2001.
42. Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Seattle, Washington, June 1994.
43. Luciano Spinello, Rudolph Triebel, and Roland Siegwart. A trained system for multimodal perception in urban environments. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
44. Sridhar Srinivasan and Rama Chellappa. Image stabilization and mosaicking using the overlapped basis optical flow field. In *Proceedings of IEEE International Conference on Image Processing*, pages 356–359, October 1997.
45. Chris Stauffer and Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.
46. J. Sullivan, P. Nillius, and Stefan Carlsson. Multi-target tracking on a large scale: Experiences from football player tracking. In *Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, 2009.
47. Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141, 2001.
48. Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburgh, PA, April 1991.
49. Iwan Ulrich and Johann Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceeding of the IEEE International Conference on Robotics and Automation*, pages 1572–1577, Leuven, Belgium, May 16–21 1998.
50. Marinus B. van Leeuwen and Frans C.A. Groen. Motion interpretation for in-car vision systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 135–140, EPFL, Lausanne, Switzerland, October 2002.
51. Jaco Vermaak, Arnaud Doucet, and Patrick Perez. Maintaining multi-modality through mixture tracking. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, pages 1110–1116, 2003.
52. Rene Vidal. Multi-subspace methods for motion segmentation from affine, perspective and central panoramic cameras. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1216–1221, 2005.
53. John Y.A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 2004.
54. Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report 95-041, Department of Computer Science, University of North Carolina at Chapel Hill.
55. Christopher R. Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.
56. Jiangjian Xiao and Mubarak Shah. Accurate motion layer segmentation and matting. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 698–703, Washington, DC, 2005.
57. Alper Yilmaz, Khurram Shafique, Niels Lobo, Xin Li, Teresa Olson, and Mubarak a. Shah. Target-tracking in FLIR imagery using mean-shift and global motion compensation. In *Workshop on Computer Vision Beyond the Visible Spectrum*, pages 54–58, Kauai, Hawaii, December 2001.
58. I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2D mosaic from a set of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 420–425, 1997.

Boyoon Jung is a Senior Robotics Engineer in Intelligent Mobile Equipment Technology (IMET) division at John Deere. He received his Ph.D. (2005) and M.Sc. (2001) in Computer Science in Computer Science from University of Southern California (USC) with specialization in Robotics and Automation. He also received his M.Sc. (1998) and B.Sc. (1996) in Computer Science from Sogang University, Korea, with specialization in Artificial Intelligence. His research interests include vehicle safeguarding, precise positioning, object classification and tracking, multi-robot cooperation, probabilistic robotics, machine learning, and computer vision.

Gaurav S. Sukhatme is a Professor of Computer Science (joint appointment in Electrical Engineering) at the University of Southern California (USC). He received his undergraduate education at IIT Bombay in Computer Science and Engineering, and M.S. and Ph.D. degrees in Computer Science from USC. He is the co-director of the USC Robotics Research Laboratory and the director of the USC Robotic Embedded Systems Laboratory which he founded in 2000. His research interests are in multi-robot systems and sensor/actuator networks. He has published extensively in these and related areas. Sukhatme has served as PI on numerous NSF, DARPA and NASA grants. He is a Co-PI on the Center for Embedded Networked Sensing (CENS), an NSF Science and Technology Center. He is a senior member of IEEE, and a member of AAAI and the ACM. He is a receipient of the NSF CAREER award and the Okawa foundation research award. He has served on many conference program committees, and is one of the founders of the Robotics: Science and Systems conference. He is one of the program chairs of the 2008 IEEE International Conference on Robotics and Automation. He is the Editor-in-Chief of Autonomous Robots. He has served as Associate Editor of the IEEE Transactions on Robotics and Automation, the IEEE Transactions on Mobile Computing, and on the editorial board of IEEE Pervasive Computing.