In *the 8th Conference on Intelligent Autonomous Systems*
pp. 980--987, Amsterdam, The Netherlands, March 10-13, 2004.

Detecting Moving Objects using a Single Camera on a Mobile Robot in an Outdoor Environment

Boyoon Jung and Gaurav S. Sukhatme
$boyoon|gaurav@robotics.usc.edu$

Robotic Embedded Systems Laboratory
Center for Robotics and Embedded Systems
Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0781

Abstract. Robust detection of moving objects from a mobile robot is required for safe outdoor navigation, but is not easily achievable since there are two motions involved: the motions of moving objects and the motion of the sensors used to detect the objects. We have experimented with a probabilistic approach for moving object detection from a mobile robot using a single camera in outdoor environments. The ego-motion of the camera is compensated using corresponding feature sets and outlier detection, and the positions of moving objects are estimated using an adaptive particle filter and EM algorithm. The algorithms are implemented and tested on three different robot platforms (robotic helicopter, Segway RMP, and Pioneer2 AT) in an outdoor environment, and the detection results are analyzed.

1 Introduction

Outdoor mobile robots have many potential applications; examples include street cleaning, traffic policing etc. The populated outdoor environment is fairly challenging to contemporary mobile robots. There are many reasons for this: *eg.* rough terrain, unstructured objects, and motion of objects in the environment. This latter category may include motions as diverse as those due to pedestrians, bicycles, automobiles, etc. Since some objects move faster than the robot, motion detection and estimation for potential collision avoidance are the most fundamental skills that a robot needs to function effectively outdoors.

Detecting motion of external objects from a moving robot is the subject of active research. There are two independent motions involved: the motion of the robot (and hence the sensors it carries) and the motions of moving objects in the environment. Unfortunately, those two motions are blended together when measured through a sensor such as a camera. In order for a robot to detect moving objects robustly, it should be able to decompose these two independent motions from sensor readings.

The computer vision community has proposed various methods to stabilize camera motions by tracking features [1, 2] and computing optical flow [3, 4]. 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 leads to poor estimation. Other approaches that extend these methods for motion tracking using a pan/tilt camera include those in [5, 6, 7]. 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. [8] tracks a single object in forward-looking infrared (FLIR) imagery taken from an airborne, moving platform, and [9, 10] tracks cars in front using a camera mounted on a vehicle driven on a paved road.

Once motion has been identified, objects in the scene need to be tracked. Work focusing on robust multiple target tracking using probabilistic filters includes [11] which uses a particle filter to track people indoors (corridors) using a laser rangefinder, and [12] which also uses a particle filter to track multiple objects using a stationary camera. A Kalman filter was used in [13] to detect and track human activity with the combination of a static camera and a moving camera.

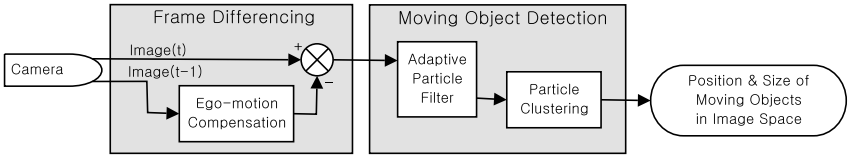


Figure 1: Processing sequence for moving object detection from a mobile robot

In this paper, we propose an approach to detect moving objects from a mobile robot using a *single camera* in an outdoor environment. The motion detection process is performed in two steps: the ego-motion compensation of camera images, and the position estimation of moving objects in the image space. For robust detection and tracking, the position estimation process is performed using a Bayes filter, and an adaptive particle filter is utilized for iterative estimation. We envisage our algorithm being used within a control loop, thus real-time requirements and robustness are the main design issues. Our algorithm was implemented and tested on three different robotic platforms (robotic helicopter, *Segway RMP*, and *Pioneer2 AT*), which have unique characteristics in terms of motions. The robots were able to detect moving objects robustly in various situations.

2 Two Independent Motions

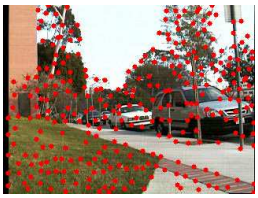
Frame differencing, which compares two consecutive image frames and finds moving objects based on the difference, is perhaps the most intuitive and fast algorithm for moving object detection, especially when the viewing camera is static. However, when the camera moves (*eg.* when it is mounted on a mobile robot), straightforward differencing is not applicable because a big difference is generated by simply moving the camera even if nothing moves in the environment. There are two *independent motions* involved in the moving camera scenario: motions of moving objects and the ego-motion of the camera. Since these two motions are blended into a single image, the ego-motion of the camera should be eliminated so that the remaining motions, which are due to moving objects, can be detected. Figure 1 shows the processing sequence of our moving object detection algorithm. Frame differencing is utilized, but the the ego-motion of the camera in the previous image ($Image(t-1)$) is compensated before comparing it with the current image ($Image(t)$).

Real outdoor images are contaminated by various noise sources, *eg.* poor lighting conditions, camera distortion, unstructured and changing shape of objects, etc. Thus perfect ego-motion compensation is rarely achievable. Even assuming that the ego-motion compensation is perfect, the difference image would still contain structured noise on the boundaries of objects because of the lack of depth information from a monocular image. Some of these noise terms are transient and some of them are constant over time. We use a probabilistic model to filter them out and to perform robust detection and tracking. The probability distribution of moving objects in image space is estimated using an adaptive particle filter [14]. As shown in Figure 1, the final particles are clustered using a mixture of Gaussians [15] for the position estimation.

It may be noted that, taken individually, each of these techniques are well-known. Our contribution in this paper is the experimental evaluation of these techniques on real monocular imagery from a robot platform taken in populated unstructured outdoor environments.

3 Ego-motion Compensation

The ego-motion of the camera can be estimated by tracking features between images [1, 2, 7]. When the camera moves, two consecutive images, I^t (the image at time t) and I^{t-1} (the image at time $t-1$), are in different coordinate systems. Ego-motion compensation is a transformation from the image coordinates of I^{t-1} to that of I^t so that the two images can be compared directly. The transformation can be estimated using two corresponding feature sets: a set of features in I^t and a set of corresponding features in I^{t-1}. However, since there are independently moving objects in the images, a transform model and outlier detection algorithm needs to be designed so that the result of ego-motion compensation is not sensitive to object motions.


(a) Outdoor features


(b) Selected features at time t

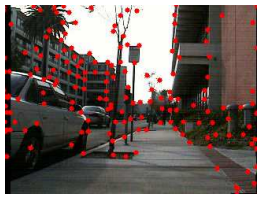

(c) Tracked features at time $t + 30$

Figure 2: Feature selection and tracking

We adopt the feature selection algorithm introduced in [16] for corresponding feature set selection. Figure 2 (a) shows selected features in the outdoor image; corners of bricks and cars, and leaves and grass that have complex textures were selected as features. The feature selection algorithm runs on image (I^{t-1}), and generates features (f^{t-1}). The *Lucas-Kanade* method [17] is applied to track those features in the subsequent image (I^t) to find the corresponding set of features (f^t). For efficiency, the search range was limited to a small constant distance (assuming a bounded robot speed). Figure 2 shows the robustness of the tracking method. Figure 2 (b) shows the features selected from the image I^t, and Figure 2 (c) shows the same features tracked over 30 frames on the image I^{t+30}. The erroneous features on image boundaries are eliminated for subsequent processing.

Once the correspondence $< f^{t-1}, f^t >$ 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, bilinear model, and pseudo-perspective model. When the interval between consecutive images is very small, most ego-motion 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 (most sensors on our robot including the camera produce data at 10 Hz), 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, not a zoom. Therefore, a nonlinear transformation model is required for our case. On the other hand, an over-fitting 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. We used a bilinear model for the experiments reported in this paper:

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

Given a transformation model (T_t), the cost function for least square optimization is defined as

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

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:
$$\begin{cases} f_i \in F_{in} & \text{if } |f_i^t - T_{0\,t-1}^t(f_i^{t-1})| < \epsilon \\ f_i \in F_{out} & \text{otherwise} \end{cases} \tag{3}$$
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 small compared to the background; the features which do not agree with the main motion are considered as outliers. This assumption will be violated 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.

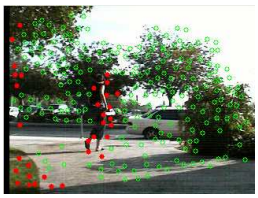


Figure 3: Inliers (empty circles) and outliers (filled circles)

(a) Image at time $t-1$

(b) Image at time t

(c) Compensated image of (a)

Figure 4: Image Transformation: (c) is the transformed image of (a) into the coordinates of (b)

For frame differencing, 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):

$$I_{comp}(x,y) = I_{t-1}\left({T_{t-1}^{t}}^{-1}(x,y)\right) \tag{4}$$

Figure 4 (c) shows the compensated image of Figure 4 (a); the translational and forward motions of the camera were clearly eliminated. The difference image between two consecutive images is computed using the compensated image:

$$I_{diff}(x,y) = |\,(I_{comp}(x,y) - I_t(x,y))\,| \tag{5}$$

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)).

4 Moving Object Detection

The *Frame Differencing* step in Figure 1 generates the difference images, $I_{diff}^0, I_{diff}^1, \cdots, I_{diff}^t$, whose normalized pixel values represent the probability of moving objects. Based on the sequence of these difference images, the position and size of the moving objects are estimated. This estimation process can be written using a Bayesian formulation. Let $\mathbf{x}^t$ represent the position of a moving object and $P_m(\mathbf{x}^t)$ be the posterior probability distribution of the object:

$$\begin{aligned}
P_m(\mathbf{x}^t) &= P(\mathbf{x}^t | I_{diff}^0, I_{diff}^1, \cdots, I_{diff}^t) \\
&= \alpha^t\ P(I_{diff}^t | \mathbf{x}^t, I_{diff}^0 \cdots, I_{diff}^{t-1})\ P(\mathbf{x}^t | I_{diff}^0 \cdots, I_{diff}^{t-1}) \\
&= \alpha^t\ P(I_{diff}^t | \mathbf{x}^t)\ P(\mathbf{x}^t | I_{diff}^0 \cdots, I_{diff}^{t-1}) \\
&= \alpha^t\ P(I_{diff}^t | \mathbf{x}^t)\ \int P(\mathbf{x}^t | I_{diff}^0 \cdots, I_{diff}^{t-1}, \mathbf{x}^{t-1}) P(\mathbf{x}^{t-1} | I_{diff}^0 \cdots, I_{diff}^{t-2})\ d\mathbf{x}^{t-1} \\
&= \alpha^t\ P(I_{diff}^t | \mathbf{x}^t)\ \int P(\mathbf{x}^t | \mathbf{x}^{t-1}) P(\mathbf{x}^{t-1} | I_{diff}^0 \cdots, I_{diff}^{t-2})\ d\mathbf{x}^{t-1} \\
&= \alpha^t\ P(I_{diff}^t | \mathbf{x}^t)\ \int P(\mathbf{x}^t | \mathbf{x}^{t-1}) P_m(\mathbf{x}^{t-1})\ d\mathbf{x}^{t-1}
\end{aligned} \tag{6}$$

The *Particle filter* [18] 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

(a) difference without compensation

(b) difference with compensation

Figure 5: Results of frame differencing

(a) Input image (b) Particles filter output (c) Gaussian mixture function

Figure 6: Moving object detection procedure

perform multi-modal tracking is attractive for multiple object detection and tracking. An efficient variant, called the *Adaptive Particle Filter*, was introduced in [14]. 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.

Particle filters require two models for the estimation process: an action model and a sensor model. A constant-velocity action model was assumed for moving object detection. Where an i_{th} particle is defined as $s^t = [x \;\; y]^T$ and Δt is a time interval,

$$s_i^{t+1} = s_i^t + \Delta t \times \dot{s}_i^t + Normal(\frac{\gamma}{\omega_i^t}) \tag{7}$$

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 converging direction. However, a dynamic mixture of divergence and convergence is required to detect newly introduced moving objects. [18] introduced a mixture model to solve this problem, but in the image space the probability $P(\mathbf{x}^t|I_{diff}^t)$ is uniform and the dual MCL becomes random. Therefore, we used a simpler, but effective method by adding inverse-proportional noise. For the sensor model, the normalized difference image (I_{diff}) is directly used as sensor input. The particle filter uses a $m \times m$ fixed-size mask (usually 5×5) to evaluate each particle. By using the mask, salt-and-pepper noise can be eliminated.

$$\omega_i^t = \frac{1}{m^2} \sum_{j=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} I_{diff}\left(s_i^t(x) - j, s_i^t(y) - k\right) \tag{8}$$

Figure 6 (b) shows the output of the particle filter. Red 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.

The particle filter generates a set of weighted particles that estimate the posterior probability distribution of moving objects, but the particles are not easy to process in the following step. More intuitive and meaningful data can be extracted by clustering the particles. Given the estimated posterior distribution using particles, a mixture of Gaussians is inferred corresponding to the posterior distribution using the *Expectation-Maximization* (EM) algorithm [15]. The Gaussian mixture function represents the original posterior distribution and the regions of moving objects can be extracted by thresholding the Gaussian mixture function. Figure 6 (c) shows the Gaussian mixture function, and the blue rectangle indicates the extracted region of the pedestrian in the input image. For real-time response, the maximum iterations of the EM algorithm is fixed to a constant.

5 Experimental Results and Discussion

The 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. The *Robotic Helicopter* [19] in Figure 7 (a) is an autonomous flying vehicle carrying a monocular camera facing downward. Once it takes off and hovers, planar movements become 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 for a change of direction still generate complicated video sequences. Also, high-frequency vibration of the engine adds motion-blur to camera images. The *Segway RMP* in Figure 7 (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

(a) Robotic Helicopter (b) Segway RMP (c) Pioneer2 AT

Figure 7: Robot platforms for experiments

moves. Especially when the robot accelerates/decelerates, the pitch angle increases seriously. 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 7 (c) is a typical four-wheeled, statically stable robot. Since the Pioneer2 robot is the most static platform among these robot platforms, we drove the robot on the most severe test environment. Figure 8 (c) and Figure 9 (c) show the rocky terrain where the robot was driven. In addition, the moving objects were occluded occasionally because of the trees in the environment.

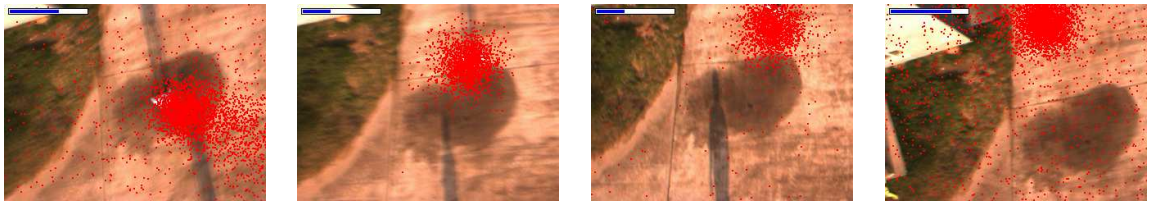
The computation was performed on embedded computers (Pentium III 1GHz) on the robots. Low resolution (320x240 pixels) input images were chosen for real-time response, and the tracking algorithm was able to process five frames per second. Since the algorithm is supposed to run in parallel with other processes (eg. navigation and communication), less than 70 percent of the CPU time was dedicated for tracking. The snapshots of the particle filter tracking moving objects are shown in Figure 8. The maximum number of particles was set to 5,000, and the minimum number of particles was set to 500. The figures show that the particle filter reduces the number of particles for efficient estimation when it converges.

The performance of the tracking algorithm was evaluated by comparing to 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 9 shows this evaluation process. The left windows show the input images, and the right windows show the posterior distribution (Gaussian mixture) functions. The thick rectangles indicate the position of manually-tracked objects, the thin rectangles indicate the output of the tracking algorithm, and the thin lines show the distance between the center of the rectangles. The final evaluation result is shown in Table 1. *Motions* is the number of moving objects over the total number of frames. *Detected* is the total number of detected objects, and *True +* and *False +* are the number of correct detections and the number of false-positives. *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 six 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 camera field-of-view, the tracking algorithm detected 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 was the worst; however, as explained in the previous section, the terrain for the experiment was more challenging and the input images were more blurred and unstable.

6 Conclusion and Future Work

A robust real-time algorithm for moving object detection was introduced for an outdoor robot carrying a single camera. The ego-motion of the camera was estimated using corresponding feature sets, and the positions of moving objects were estimated using an adaptive particle filter. The algorithms were

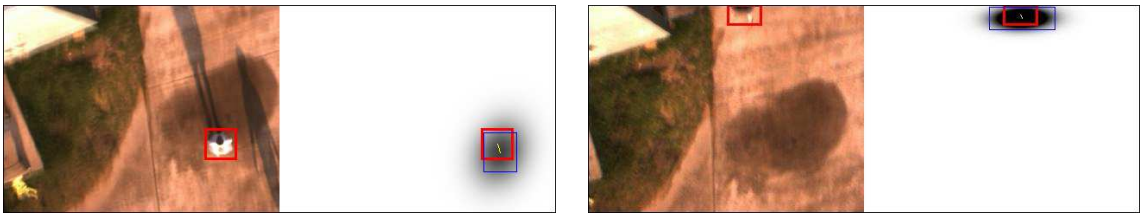


(a) Robotic helicopter

(b) Segway RMP

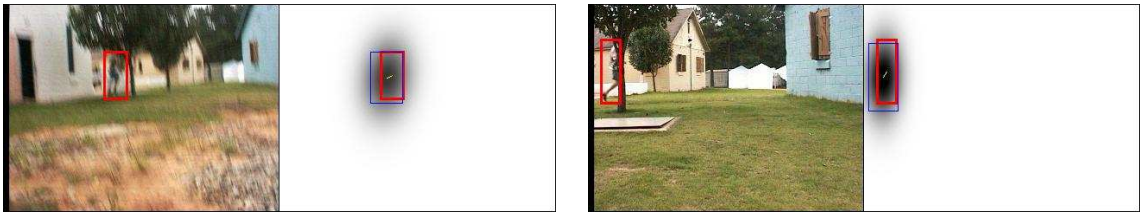
(c) Pioneer2 AT

Figure 8: Particle filter output



(a) Robotic helicopter

(b) Segway RMP



(c) Pioneer2 AT

Figure 9: Performance Evaluation

Platform	Motions	Detected	True +	False +	Detection Rate	Avg. Error
Robotic helicopter	35 / 43	29	29	0	82.86 %	13.26
Segway RMP	220 / 230	208	206	2	93.63 %	20.29
Pioneer2 AT	172 / 195	126	114	12	66.28 %	13.54

Table 1: Performance of moving object detection algorithm

implemented and tested on three different real robots in various outdoor environments, and the performance statistics shows the current system can detect moving objects robustly in the camera image space, and shows a basic tracking capability.

The current system can also estimate the direction and speed of the moving objects since the state vector of the particle filter includes velocity information. A limitation of the current system is that it provides the position and velocity information of moving objects in the image space, which is 2-dimensional. Since a single camera has limit on retrieving depth information, the information from a camera alone is not rich enough to construct full 3-dimensional models of moving objects. However, our robots are equipped with a laser rangefinder, which provides the depth information of a singe plane. Given the optical properties of a camera and the geometry between the camera and the laser rangefinder, the distance information from the laser rangefinder can be projected onto the image coordinates. As a result, the image pixels at the same height as the laser rangefinder will have depth information. 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.

Acknowledgments

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.

References

[1] 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.

[2] 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.

[3] 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*, 1997.

[4] 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.

[5] 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.

[6] 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.

[7] Gian Luca Foresti and C. Micheloni. A robust feature tracker for active surveillance of outdoor scenes. *Electronic Letters on Computer Vision and Image Analysis*, 1(1):21–34, 2003.

[8] 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*, Kauai, Hawaii, December 2001.

[9] 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.

[10] Marinus B. van Leeuwen and Frans C.A. Groen. Motion Interpretation for In-Car Vision Systems. In *the Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Lausanne, Switzerland, October 2002.

[11] Dirk Schultz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, pages 1165–1170, 2001.

[12] 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.

[13] 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.

[14] Dieter Fox. Kld-sampling: Adaptive particle filter. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[16] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Pittsburgh, PA, April 1991.

[17] 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.

[18] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141, 2001.

[19] 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.