TRACKING MOVING OBJECTS BY A MOBILE CAMERA

MS-CIS-88-97

Sang Wook Lee and K. Wohn

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

1987

TRACKING MOVING OBJECTS BY A MOBILE CAMERA

Sang Wook Lee
and K. Wohn

MS-CIS-88-97
GRASP LAB 168

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

November 1988

Acknowledgements: This research was supported in part by DARPA grants NOOO14-85-K-0018, N00014-88-K-0632, U.S. Postal Service Contract 104230-87-H0001/M-0195, NSF grants MCS-8219196-CER, IRI84-10413-AO2 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

Abstract

A system for video tracking of a moving object by the robot-held camera is presented, and efficient tracking methods are proposed. We describe our initial implemention of a system which is capable of tracking a single moving object against highly textured background. A pyramid-based image processor, PVM-1 is employed to support some fast algorithms in locating the moving object from the difference image. Object tracking is accomplished in the static look-and-move mode by the translational motion of a CCD camera mounted on the robot arm. Discussion is given on the implementation of tracking filters and on the effective utilization of multi-resolution processing for the object searching. Finally a method for dynamic look-and-move tracking is proposed for the future improvement of tracking performance.

Key Words

tracking, real-time image processing, image motion analysis

1. Introduction

One of the goals of image motion analysis is to recover the 3-D geometric shape of a scene as well as its relative motion with repect to the camera. Although there have been many algorithms developed towards this goal, few of them have shown their robustness in unstructured environments. This is partly due to the fact that most of the algorithms proposed so far rely on small number of frames (usually two or three) and consequently they are extremely sensitive to the noise in the input (either the image flow or the disparity vector). It is only recently that researchers started investigating the temporal behavior of a moving scene for an extended period of time, in an attempt to improve both the 2-D motion and the 3-D motion (in terms of the translation and rotation of moving bodies) over time by utilizing the explicit model for the 3-D motion [Broida1986a, Weng1987a, Broida1986b, Iu1989a, Kumar1989a]. While the motion analysis for a relatively long period of time has many expected advantages, it requires the ability of real-time tracking to keep the moving object of interest always in sight.

Another advantage of tracking ability was addressed by Bandyopahathy et. al. [Bandyopadhay1985a] who suggested that the procedure of recovering 3-D motion and structure can be simplified if one traces a fixed point of moving object continuously so that the viewing axis of camera coordinates is always coincident with the moving point. Another insightful remark is due to Bajcsy who claimed that visual perception is highly exploratory; we move our heads or change our position to get a better view of the scene [Bajcsy1988a]. One necessary condition to realize the exploratory perception is the ability to solve the correspondence problem; which region of an image physically corresponds to which region of another image. Although exploratory perception may be based on the *static-first* approach such as stereo vision and multi-view analysis, image motion as an independent perceptual module must be incorporated to the *static-first* modules to deal with more complex, dynamic environment. For example, if one wants to handle a moving object on the conveyer belt, it is necessary for the system to keep track the moving part while it conducts exploratory perception. Moreover, in robot hand-eye coordination which is one of the primary target domains of our current investigation, demanded is a vision system which is able to respond to a moving environment and generate the proper feedback in real time.

Perhaps the simplest senario on target tracking would be the one in which an object moves againt the uniform background. The moving object is isolated easily by the thresholding technique. In noisy environments, one can determine the thresholding level adaptively based on the stochastic analysis to distinguish the object of interest from the background. Radar target tracking falls into this category.

The above technique cannot be applied to natural scenes of highly textured background. There are several techniques, however, for detecting moving objects against the textured, but stationary background. Jain claimed that some useful information can be extracted by analyzing the difference image between two successive frames in time [Jain1981a]. Anderson et. al. implemented a real-time tracking system based on the motion energy computed from the difference between two consecutive frames[Anderson1985a]. Note that these methods will not work wherever the relative motion between the background and the camera exists.

The most general and versatile method would be the one based on the image flow analysis [Burt1989a]. Image flows are obtained either by the local analysis which establishes the correspondance of features such as gray levels, edge segments and feature points [Horn1981a, Hildreth1984a, Waxman1985a], or by the global anaysis which identifies global features such as closed contours or regions, and computes the image motion directly from their changes (e.g., [Kanatani1984a, Brown1987a]). By segmenting image motion field, one can discriminate moving objects from the moving background [Adiv1985a]. Although there have been some concerns on the architectural issues for the motion estimation [Burt1982a, Anandan1987a, Waxman1988a], the task of recovering and segmenting image flows is computationally too costly to be of any practical use in real time. More effective and reliable methods as well as the hardware that can carry out the required computation handling enormous amount of data are yet to come.

In this paper we describe our initial development of a system which is capable of tracking a single moving object against highly textured background. Our method of detecting and locating a moving object is based on the image differencing method and utilizes the pyramid processing algorithms developed by Anderson et. al. [Anderson1985a]. A CCD camera is mounted on a robot arm (PUMA 560) which moves perpendicular to the camera axis so that the moving object appears at the center of the image plane (or at least near the fovea) all the time. See Figure 1 for the experimental setup. The system employs an existing pyramid-based image processor (PVM-1) which supports some fast algorithms in locating a moving object. We have accomplished a moderate success so far; the speed of object motion is limited to 10 degrees of viewing angle per second (It roughly corresponds to the motion of an object which is located 2 feet away from the camera and moves 6 inches per second).

As indicated before, the image differencing method imposes a requirement that the camera should be in a quienscent state at the time of image acquisition, otherwise there is no way of discriminating moving part of an image from its moving background. The limitation of tracking speed is mainly attributed from the momentary suspension of tracking motion. To alleviate the large discontinuity in monitoring an object, we propose a tracking method based on the spatio-temporal analysis of image flow. The method of compensating the background drift due to the camera motion is described in Section 5, and now under implementation. In section 2 the configuration and the operation of the system is described, and the detailed presentation on object searching algorithms is given in Section 3. Section 4 discusses the estimation and prediction of a moving object for the compensation for the delay between the visual observation and the actual tracking by the manipulator.

2. The System Configuration

The video trackig system has been implemented by combining the pyramid image processing unit with the PUMA 560 system supported by its own controller. As shown in Figure 2, the image processing unit consists of the PVM-1, a prototype pyramid processor developed at the Sarnoff Research Center, the image controller with video buffers, and the dedicated microcomputer-level processor (IBM-RT). This unit performs the image acquisition from the robot-held camera and processes the sequence of images at the video rate, for the calculation of the position variables of the moving object. The master control processor (VAX 11/750) receives the position variables from the image processing unit, and performs the temporal filtering for the estimation and prediction. We use our existing software for the communication between the master processor and the robot controller.

The system operation is in the static look-and-move (SLAM) mode, that is, the robot is not moving at the time of visual observation. Upon obtaining a measured position of an object, the robot-held camera follows the object within a designated period of time. The camera motion is translational in 2-D x-y coordinates, and the perceived 2-D position of a moving object is uniquely mapped into the camera position by simple scaling and calibration. Although this SLAM system imposes a restriction that the speed of the object should be substantially lower than that of the tracking robot, the object is assumed to be highly maneuvering.

2.1. Pyramid Processing and Motion Detection

An image pyramid is a set of multi-resolution images in which both the bandwidth and the image size are varied in regular steps [Burt1983a]. A Gaussian pyramid is constructed by successive application of Gaussian low-pass filtering and decimation by a half, as shown in Figure 3. In the figure, G_0 denotes the original image of 256 by 256 pixels, and G_1, G_2,.... are low-passed and decimated copies of G_0 with the images sizes 128 by 128, 64 by 64,..., repectively. A Laplacian pyramid is a set of band-pass images L_k's which are obtained by subtracting the undecimated images of G_{k+1}'s from G_k's. As a result, the band-passed images L_k's are the ones that can be obtained by applying Laplacian operators on the images G_k's.

The pyramid processor is a special-purpose hardware for constructing pyramids in real time [Wal1985a]. It is capable of producing complete Gaussian or Laplacian pyramid from a 256 by 256 image in one video image frame (1/30 of a second). This pyramid processor supports fast algorithms for several image analysis tasks, among others, the coarse-fine target detection is our primary concern.

It has been demonstrated by Anderson et. al. [Anderson1985a] that the pyramid processor can effectively be utilized for the real-time detection and tracking of image changes. In their approach, the motion detection is accomplished by the image differencing in time. (See Figure 4.) The difference of images is obtained by subtracting an image from another adjacent in time. This pixel-by-pixel subtraction works as a temporal high-pass filter; a fast moving object would leave more energy in the difference image. The difference image is then decomposed into a set of spatial frequency bands through the construction of a Laplacian pyramid, each level of which shows the different temporal and spatial information on the moving object. Slow motion of an object is mostly apparant in the top level L_0. Therefore the image at L_0 is selected and the absolute value of pixels is taken as the measure for the motion energy. From the absolute-valued image of L_0, is constructed the Gaussian pyramid which represents the locally integrated measures for the motion energy in various resolutions.

With the final Gaussian pyramid, motion of an object can be located in a highly efficient manner. The coarse-fine search begins by examining the 16 by 16 image at G_4. If there is any noticiable change in energy at G_4, a 16 by 16 subimage at G_3 are examined around the center of energy change detected at G_4, resulting in a more precise information on the position of object. This refinement process continues towards G_0 until the number of pixels with non-zero motion energy exceeds a predefined threshold. The threshold value is calibrated before the tracking begins so that the window of examination does not get smaller than the size of object. By successively confining the region of search in this way, the size of image data for the searching operation can be greatly reduced. For instance, searching of an object in five 16 by 16 images instead of the entire 256 by 256 image results in a data reduction of about 1/50, facilitating the real-time operation.

2.2. Timing Scheme for Synchronization

Figure 5 shows the timing diagram for video tracking. Most of the image acquisition and processing are synchronized with the vertical sync pulse of the video display signal and thus the period of processing time can be denoted in the unit of *VFT*, video frame time (1 *VFT* = 1/30 *seconds*).

The image acquisition time t_a is 2 *VFT*'s with the sampled images for the differencing apart in time by 1 *VFT*. After t_a, $t_p (= 2\ VFT)$ is required for the pyramid processor to construct the Laplacian and Gaussian pyramids. The next delay of t_c is required for the calculation of position variables by the dedicated microcomputer which is capable of examinig two 16 by 16 images in 1 *VFT*. t_c could be up to 3 *VFT*'s depending upon the searching algorithms which will be discussed in the following sections. It is after t_c that the measured values of position varibles x and y (denoted by θ_m) are obtained, and sent to the master processor (VAX 750).

Upon receiving the position variables, the master processor performs the estimation and prediction filtering which takes less than 1 *VFT*. This time span is denoted as t_f. The delay of 9 *VFT*'s (denoted as t_m) is reserved for the robot motion as well as for the calculation of the inverse kinematics in the world coordinates x and y. This long delay is necessary for the static look-and-move tracking mode which assumes a complete quiescence of the camera at the time of next visual observation. With our prototype system, the software that drives the robot controller introduces a significant delay of 5 *VFT*'s in sending the position data from the master processor to the robot manipulator. This can be reduced substantially with some future efforts of connecting the image processing unit directly to the robot controller, avoiding the presence of the master processor. The remaining 4 *VFT*'s are required to ensure that the robot has moved to the destination before the next cycle starts. In summary, it takes 15 to 17 *VFT*'s (approximately 0.5 seconds) to complete one operation cycle. Although it is very difficult to further reduce the time required for the image processing, there is enough room to improve the speed by establishing a faster communication to the robot controller.

In any sensor-based tracking and servoing system, inherent are the time delays introduced by the computation of control variables (positions) and by the motion of camera mount (robot). Furthermore there is the intrinsic noise resulting from errors in the visual measurement as well as the positioning errors of the robot. In order to keep the camera foveated at the object as closely as possible, the provision for estimation and prediction is essential for the compensation for the detection error and the processing delay. As shown in Figure 7, the estimated position $\hat{\theta}(k)$ at the discrete cycle reference k, is calculated from the measured value $\theta_m(k)$ and the previously obtained prediction $\hat{\theta}_p(k|k-1)$. $\hat{\theta}(k)$ is then used for the prediction of the next position $\hat{\theta}_p(k+1|k)$ which is sent to the robot manipulator so that the tracking camera can be forwarded to the next position of the object.

3. Searching and Detecting a Moving Object

The first step in tracking with the pyramid processor is to locate the small region of search at each of the five levels of multi-resolution images, over the series of tracking cycles. A few algorithms have been implemented and been tested under various object/background scenarios.

One obvious way of locating the moving object is to successively examine 16 by 16 subimages in each of the five levels until the change energy meets the predefined threshold. As mentioned before, the threshold is set such that the size of the window for examination does not become smaller than the object size. Figure 6a shows a timing diagram of level transition for a level-0 object (the size of which is perceived to be smaller than the window size at the level 0). In each of the 5 levels, the positions of local energy change are integrated to find the object center which is then used for centering a searching window at the next higher level. Note that the level is preset to the bottom level before each cycle of tracking. This searching method requires t_s of 2.5 *VFT* in each cycle due to the limitation of the computing speed (two 16 by 16 image searching in 1/30 msec) of the microcomputer. Figure 6b and Figure 6c depict the level transition and the window converging (for a level-2 object), respectively.

The 5-level searching has the advantage that, since the whole visual area is examined from the bottom level, any single object within the field of view can be located in a complete tracking cycle, that is, this searching is robust for a sudden change of object position. On the other hand, an apparent disadvantage of 5-level searching is long t_s of 2.5 *VFT*'s. Besides, it is susceptable to the background noise due to the wide view of searching. If there is a sudden change of local illumination in the background and the motion energy of background is comparable to that of the moving object, the tracking window tends to follow the background noise, temporarily losing track of the object. The mistracking also occurs when the camera does not reach a static state at the time of visual observation. The searching window then responds to a global drift of the structured backgrounds.

One of our searching algorithms to avoid the instability of 5-level searching is the predictive 2-level searching. With this algorithm, only two 16 by 16 subimages are examined at each of two selected pyramid levels in a tracking cycle, thereby requiring 1 *VFT* for the searching. Let us first consider the tracking of a manuevering level-0 object in a steady state (Figure 7). As shown in Figure 7a, once the object location is identified at the level 0 in a tracking cycle, the searching window for the next cycle can be preset at the one-step lower level centered at the predicted position. This predictive down-leveling is depicted in Figure 7b. While stepping down one level around the predicted center ensures a relatively wide view of searching, the window is still confined to a region of interest, thus insensitive to the background noise or the drift outside the region.

However, for the initial detection of the object position, the 2-level searching algorithm of 1-step-up-down leveling will never guarantee the convergence of the searching window to the higher level of interest as depicted in Figure 8a. Therefore another detection strategy is conceived for a level-0 or level-1 object (for which the pyramid processing is most effective) for the bootstrapping stage, assuming that the object moves without significant changes in size. First, we detect any change of energy at the level 3 or 4; the next searching window is deliberately located at the two level above, i.e., the level 1 or 2, respectively. This abrupt jump between levels guarantees the fast convergence of the searching window until the search is stabilized at the appropriate level. One-step level-down also occurs before the beginning of each tracking cycle, as directed by the algorithm. At the level 2, a predetermined threshold for up-leveling is used to determine the destination, either the level 0 or 1, depending upon the number of active pixels. Figure 8a shows the timing diagram of leveling at the initial detection of a level-0 object. As shown in the figure, only two cycles are needed to reach the steady state. The level-transition diagram for the 2-level predictive searching is shown in Figure 8c. Note that all the levels have unique up- and down-path, except the level 2 at which the up-leveling path is determined by an extra energy threshold.

From the experiments with some highly maneuvering objects of various sizes, it was observed that, inspite of the complexity involved in the initial detection, the 2-level predictive searching is quite insensitive to the background noise or drift, while promptly responding to a moving object upon detection. This algorithm can further be extended to include larger objects at the level 2 or 3 with extra energy thresholds at the levels 3 and 4.

4. Estimation and Prediction of the Target Position

The 2-level searching works best when the target moves with a constant velocity, with the maximum speed confined by the size of searching window. The method breaks down whenever the target travels outside the preditive window. Such situation is depicted in Figure 9, as the target moves too fast for the searching window to keep up with. It is desirable for the searching window to move ahead to the predicted target position so that the searching is carried out from the center of predicted position rather than the current target position. There has been a great deal of work on the high performance tracking estimators and predictors [Fitzgeralda,Houles1978a]. Among others, the famed extended Kalman filter has extensive applications as an optimal filter with proper modeling of target dynamics. At this stage of our work, however, we do not assume any specific application domain for our developmental system. We thus seek a simple estimation scheme for a highly manuevering object in clutter without utilizing a priori model for the object dynamics. For our experiment, therfore, adaptive filters are considered to be suited in that they are not based on the knowledge of object statistics but rely only on the data obtained.

The general form of N-point adaptive estimator can be written as

$$\hat{\theta}(k) = \sum_{n=0}^{N-1} a_n \, \theta_m(k-n) \, , \tag{1}$$

where the adaptive constants a_n's are to be updated to give the minimum mean-squares error (MSE) of $\hat{\theta}(k)$. As well noted, since full adaptation of a_n's requires complex computation involving the calculation of the inverse of the covariance matrix of $\theta_m(k)$, $\theta_m(k-1)$, ..., $\theta_m(k-N+1)$, the computational burden for a small general purpose processor is not trivial for realtime processing with N larger than 3 [Papoulis1965a]. Thus we have used a simpler form of adaptive estimates for our experiment without expecting a significant degradation.

Assuming that we have a predictor which provides the predicted value $\hat{\theta}_p(k \,|k-1)$ before the time k, we wish to form an estimate $\hat{\theta}(k)$ as a linear combination of $\hat{\theta}_p(k \,|k-1)$ and the measured $\theta_m(k)$ obtained by the pyramid processing. Thus we write $\hat{\theta}(k)$ as

$$\hat{\theta}(k) = \alpha_1 \, \hat{\theta}_p(k \,|k-1) + \alpha_2 \, \theta_m(k), \tag{2}$$

where the adaptive weighting factors α_1 and α_2 are to be determined for $\theta(k)$ to have minimum mean-squares error (MSE) under the constraint;

$$\alpha_1 + \alpha_2 = 1 \tag{3}$$

The MSE is then

$$\sigma^2 = E\left[\left[\theta(k) - \hat{\theta}(k)\right]^2\right] \tag{4}$$

$$= E\left[\left[\theta(k) - \alpha_1 \, \hat{\theta}_p(k \,|k-1) - \alpha_2 \, \theta_m(k)\right]^2\right]$$

Assuming that $\hat{\theta}_p$ and θ_m are independent, we can easily get α_1 and α_2 that minimize MSE as

$$\alpha_1 = \frac{\sigma_m^2}{\sigma_p^2 + \sigma_m^2} \ , \tag{5}$$

$$\alpha_2 = \frac{\sigma_p^2}{\sigma_p^2 + \sigma_m^2} \ ,$$

where the variances σ_m^2 and σ_p^2 can be approximated by instantaneous estimation errors;

$$\sigma_p^2 = E\left[\left[\theta(k) - \hat{\theta}_p(k \mid k-1)\right]^2\right] \tag{6}$$

$$\approx \left[\hat{\theta}(k-1) - \hat{\theta}_p(k-1 \mid k-2)\right]^2,$$

$$\sigma_m^2 = E\left[\left[\theta(k) - \theta_m(k-1)\right]^2\right]$$

$$\approx \left[\hat{\theta}(k-1) - \theta_m(k-1)\right]^2$$

Note that the estimated values rely more on the measured values when there is a larger error in the predicted values, and vice versa.

The obtained $\hat{\theta}(k)$ is then used for the calculation of a predictive value $\hat{\theta}_p(k+1 \mid k)$ at the time indexed $k+1$. The predictor can also be a linear combination of simple extrapolation filters rather than a fully adaptive filter. We used a linear combination of a linear 2-point predictor ($\hat{\theta}_l$) and a quadratic 5-point predictor ($\hat{\theta}_q$) as in [Flachs1977a].

$$\hat{\theta}_p(k+1 \mid k) = \beta_1 \, \hat{\theta}_l(k+1 \mid k) + \beta_2 \hat{\theta}_q(k+1 \mid k) \ , \tag{7}$$

where

$$\hat{\theta}_l(k+1 \mid k) = 2\hat{\theta}(k) - \hat{\theta}(k-1), \tag{8}$$

$$\hat{\theta}_q(k+1 \mid k) = \frac{1}{5}\left\{9\hat{\theta}(k) - 4\hat{\theta}(k-2) + 3\left[\hat{\theta}(k-5) - \hat{\theta}(k-4)\right]\right\}.$$

Weighting factors β_1 and β_2 are also to be determined in the same manner as are α_1 and α_2 for $\hat{\theta}$;

$$\beta_1 = \frac{\sigma_q^2}{\sigma_l^2 + \sigma_q^2}, \tag{9}$$

$$\beta_2 = \frac{\sigma_l^2}{\sigma_l^2 + \sigma_q^2},$$

where

$$\sigma_l^2 = \left[\hat{\theta}(k) - \hat{\theta}_l(k \mid k-1)\right]^2, \tag{10}$$

$$\sigma_q^2 = \left[\hat{\theta}(k) - \hat{\theta}_q(k\,|k-1)\right]^2,$$

The 2-point predictor, requiring only two previous values, has an advantage of promptly initiating a prediction upon detecting an object, while the 5-point predictor provides a better curve fitting of the object trajectory. The estimation and prediction loop is shown in Figure 10. The operation is computationally efficient involving less than 20 mutiplications for each position variable. An alternative approach to the same problem is discussed in [Safadi1987a], which utilizes the image velocity estimates in addition to the position data.

Figure 11 illustrates the performance of the 2-level searching with the estimation and prediction. Initially the target moves upward (Figure 11a,b,c), then it reverses the motion abruptly (Figure 11d). It further moves down along a smooth path (Figure 11e,f). During the intitial upward motion the estimated position (depicted as "+") and the predicted position (depicted as "X") almost coincide. Notice the large difference between two positions just after the target changed its motion. Two positions come close each other as the target undergoes the smooth motion thereafter.

5. Dynamic Look-and-Move Tracking

In the static look-and-move tracking scheme described so far, the major impediment to increasing the tracking speed is the unavoidable delay time for searching, filtering and for manipulator motion. To separate the moving object from the stationary background, the camera held by the manipulator has to be completely without motion at the time of visual observation, otherwise we observe serious mistracking as the camera responds to the background. Since the delay time required for the manipulator motion being substantially longer than that for filtering and searching, major efforts in improving the tracking performance may be directed to minimizing the manipulator motion or avoiding the need for the momentary stop of the manipulator. In addition, the minimization of the delay in the communication between the modules is equally important. The improvement of the prediction filter by better modelling of object dynamics is another concern too.

In improving of tracking performance in terms of reducing or avoiding the manipulator delay time, the followings are the major considerations.

1) To reduce the time interval of operation cycle by replacing the robot manipulator with a faster one, which is very costly and even impossible beyond a certain limit.

2) To control the manipulator to move with angular motion only. This has the advantage over the tranlational motion that a small change in camera angle covers a large change in the view of the scene, thereby tracking motion can be completed in a shorter interval even for larger motion of the object. Another advantage of the angular-motion-only scheme is discussed later in this section associated with the method as follows.

3) To develop a new tracking scheme that continuously monitors the target motion without the need of a momentary stop during the image acquisition. This method which we call dynamic look-and-move (DLAM) shall perform the best.

In the rest of this section, we shall describe a simple version of DLAM with the camera under the angular-motion-only. It is interesting to note that the change in image (or image motion) of stationary scene is always predictible whenever the camera undergoes only the rotational motion. It can be seen from the mathematical analysis motivated by the projection relations from a 3-D scene to a 2-D image for an object undergoing a smooth space motion. Adopting a 3-D coordinate system (X, Y, Z) as in previous works on the *structure from motion* [Waxman1985b, Longuet-Higgins1980a], relative rigid body motion can be represented in terms of monocular viewer motion (camera motion): the translational velocity $\boldsymbol{V} = (V_X, V_Y, V_Z)$, and the rotational velocity $\boldsymbol{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)$. (See Figure 12.) The origin of the image coordinate system (x, y) is located at $(X, Y, Z) = (0, 0, 1)$.

As a point P in space (located by position vector $\boldsymbol{R}$) moves with a relative velocity $\boldsymbol{U} = -(\boldsymbol{V} + \boldsymbol{\Omega} \times \boldsymbol{R})$, the corresponding point p in the image plane moves with a velocity $\boldsymbol{v}(x, y)$ given by

$$v_x = \left\{ x \frac{V_Z}{Z} - \frac{V_X}{Z} \right\} + \left\{ xy\, \Omega_X - (1 + x^2)\, \Omega_Y + y\, \Omega_Z \right\}, \tag{11a}$$

$$v_y = \left\{ y \frac{V_Z}{Z} - \frac{V_Y}{Z} \right\} + \left\{ (1 + y^2)\, \Omega_X - xy\, \Omega_Y - x\, \Omega_Z \right\}. \tag{11b}$$

These equations define an instantaneous image flow field, assigning a unique 2-D image velocity $\boldsymbol{v}$ to each direction (x, y) in the observer's field of view. One can notice that, although the image flow is determined by many 3-D parameters - translation, rotation and the distance to the 3-D point, the contribution by the rotational motion to the image flow is independent of other parameters. This is no more true for the translational motion as they are coupled with Z, the distance parameter, that is, the change in camera location by translation will cause the image to change inversely proportional to the distance between camera and scene, due to the perspective effect. Having known the rotational parameter of camera motion, one can compensate for the effect of rotational motion from the image flow. Any non-zero image flow after the compensation is due to the target motion.

A simple DLAM tracking with the angular-motion-only scheme can be implemented using the following methods of motion compensation.

1) Compensation of image velocity; The image flow v_x and v_y generated due to the camera motion $\boldsymbol{\Omega}$ is calculated and subtracted from the image flow field obtained from the two frames of images, as illustrated in Figure 13a. Since this method is based on the general image flow analysis, the computaional load for the extraction of the image flow field is not trivial.

2) Compensation of image itself; Prior to the image differencing, one can compensate for the image displacement due to the camera motion during the short period of sampling interval (1 *VFT*) as shown in Figure 13b. Although pixel by pixel operation is required due to the non-uniform displacement of the image, the computation is less involved than the above method.

The key to the successful compensation for the background shift is to provide an apparatus for the camera to move with respect to the origin (0, 0, 0). Also crucial is the precise knowledge of the angular velocity as well as the reference position of the manipulator. As we are dealing with the whole image plane in DLAM tracking, it is necessay to compensate for the nonlinearity in the visual observation with camera.

As discussed before, in SLAM mode, the manipulator control and the data acquisition/processing are completely separated in time, while in DLAM mode the image acquisition can be done anytime since the camera does not have to wait for a quiescence of the manipulator. Therefore image acquisition and the manipulator motion can proceed simultaneously though perhaps at different sampling rate.

In addition to the DLAM tracking, there are many topics demanding further developments. Among others, tracking of multiple-moving objects is an obvious extension of the single object tracking. It requires separate management of mutiple windows for searching in one image, that is, upon detecting multiple-moving objects in a pyramid level, several independent windows should be set up at the next higher level centered at each center of motion energy. For the realization of this scheme, more computation is required but it can be highly parallelized. In tracking multiple objects, a strategy should be provided in maintaining the field of view as the objects are getting apart. For instance, one can increase the field of view by changing either the camera zoom or the camera position along the line of sight, or the combination of both. An alternative way is to follow only one object of the most interest. In this case a decision of *which object to follow* should be made on some criteria, for example, the motion energy, the size and the shape of object. Another concern is the identification of moving object, that is, identifying the object shape within the searching window using some pattern recognition techniques. The pyramid structure can also be exploited in the shape recognition [Burt1988a]. The object feature in turn is potentially another source of visual information for tracking. This will lead to a feature-based tracking combined with the position-based scheme.

6. Concluding Remark

A video tracking system in SLAM mode has been realized for a moving object in highly textured background. We described the initial implementation of the tracking system as a testbed for further development towards various applications in the exploratory vision and the robot hand-eye coordination. The pyramid processing offers some efficient ways of reducing the computation time for the object searching, and the time delay between the visual observation and the manipulator motion is compensated for by a simple, yet efficient prediction/estimation filter. While SLAM tracking has been implemented using communication protocols between existing systems, significant challenge arises in the development of tracking systems of the DLAM scheme. Our future work will be mainly focused on the development of the algorithms as well as on the design of systems for the DLAM tracking as outlined in the the paper.

Acknowledgement

Authors would like to thank to Dr. Peter Burt and his staff at the David Sarnoff Research Center for providing the PVM-1 and its software used in this work. We also thank many members of the the GRASP Laboratory, especially F. Fuma and A. Izaguire for their help with the systems implementation.

References

Broida1986a.
T. J. Broida and R. Chellappa, "Estimation of Object Motion Parameters from Noisy Images," *IEEE PAMI* **8**(1) pp. 90-99 (January 1986).

Weng1987a.
J. Weng, T. S. Huang, and N. Ahuja, "3-D Motion Estimation, Understanding, and Prediction from Noisy Image Sequences," *IEEE PAMI* **9**(3) pp. 370-389 (May 1987).

Broida1986b.
T. J. Broida and R. Chellappa, "Kinematics and Structure of a Rigid Object from a Sequence of Noisy Images," *Proc. IEEE Workshop on Motion,* pp. 95-100 (1986).

Iu1989a.
S-L. Iu and K. Wohn, "Estimation of 3-D Motion and Structure Based on a Temporally Oriented Approach with the Method of Regression," *IEEE Workshop on Visual Motion,* pp. 273-281 (March 1989).

Kumar1989a.
R. V. Raja Kumar, A. Tirumalai, and R. Jain, "A Non-Linear Optimization Algorithm for the Estimation of Structure and Motion Parameters," *IEEE Conf. Computer Vision Pattern Recognition,* pp. 136-134 (June 1989).

Bandyopadhay1985a.
A. Bandyopadhay and J. Aloimonos, "Perception of Rigid Motion from Spatio-Temporal Derivatives of Optical Flow," TR-157, Computer Science Department, Univ. Rochester (March 1985).

Bajcsy1988a.
R. Bajcsy, "Active Perception," *IEEE Proceedings* **76**(8)(August 1988).

Jain1981a.
R. Jain, "Dynamic Scene Analysis Using Pixel-based Processes," *Computer* **14** pp. 12-18 (1981).

Anderson1985a.
C. H. Anderson, P. J. Burt, and G. S. van der Wal, "Change detection and Tracking Using Pyramid Transform Techniques," *Proc. SPIE* **579** (September, 1985).

Burt1989a.
P. J. Burt, J. R. Burgen, R. Hingorani, R. Kolczynski, W. A. Lee, A. Leung, J. Lubin, H. Shvaytser, and Object Tracking with a Moving Camera, *IEEE Workshop on Visual Motion,* pp. 2-12 (March 1989).

Horn1981a.
B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence* **17** pp. 185-204 (1981).

Hildreth1984a.
E. C. Hildreth, ''Computations Underlying the Measurement of Visual Motion,'' *Artificial Intelligence* **23** pp. 309 - 354 (1984).

Waxman1985a.
A. M. Waxman and K. Wohn, ''Contour Evolution, Neighborhood Deformation and Global Image Flow: Planar Surfaces in Motion,'' *Intl. Journal of Robotics Research* **4** pp. 95-108 (1985).

Kanatani1984a.
K. Kanatani, ''Detection of Surface Orientation and Motion From Texture by a Stereological Technique,'' *Artificial Intelligence* **23** pp. 213-237 (1984).

Brown1987a.
C. Brown, J. Aloimonos, M. Swain, P. Chou, and A. Basu, ''Texture, Contour, Shape and Motion,'' *Pattern Recognition Letters* **5** pp. 151-168 (Feb. 1987).

Adiv1985a.
G. Adiv, ''Determining 3-D Motion and Structure from Optical Flow Generated by Several Moving Objects,'' *IEEE PAMI* **7**(4) pp. 384-401 (July 1985).

Burt1982a.
P. J. Burt, C. Yen, and X. Xu, ''Local Correlation Measures for Motion Analysis,'' *IEEE Conf. Computer Vision Pattern Recognition,* pp. 269-274 (June 1982).

Anandan1987a.
P. Anandan, ''A Unified Perspective on Computational Techniques for the Measurement of Visual Motion,'' *Proc. 1'st Intl. Conf. Computer Vision,* pp. 219-230 (June 1987).

Waxman1988a.
A. M. Waxman and J. Wu, ''Convected Activation Profiles and Image Flow Extraction,'' *IEEE Conf. Computer Vision Pattern Recognition,* (June 1988).

Burt1983a.
P. J. Burt and E. H. Adelson, ''The Laplacian Pyramid as a Compact Image Code,'' *IEEE Trans. Communication* **31** p. 532 (1983).

Wal1985a.
G. S. van der Wal and J. O. Sinniger, ''Real-Time Pyramid Tranform Architecture,'' *Proc. SPIE* **579**(September, 1985).

Fitzgeralda.
R. Fitzgerald, ''Simple Tracking Filters: Closed Form Solutions,'' *IEEE Trans. Aerospace and Electronic Systems* **17**(6) pp. 781-785 (1986).

Houles1978a.
A. Houles and Y. Bar-Shalom, ''Mutisensor Tracking of a Manuevering Target in Clutter,'' *Proc. IEEE Natl. Aerospace and Electronics Conf.*, (May 1978).

Papoulis1965a.
A. Papoulis, *Probability Random Variables Stochastic Processes,* McGraw-Hill, New York (1965).

Flachs1977a.

G. M. Flachs, W. E. Thomson, R. J. Black, J. M. Taylor, W. Cannon, R. Rogers, and Y. Hsun U, ''An Automatic Video Tracking System ,'' *Proc. IEEE Natl. Aerospace and Electronics Conf.*, (May 1977).

Safadi1987a.

R. Safadi, ''An Adaptive Tracking Algorithm for Robotics and Computer Vision Applications,'' M.S. Thesis, Dept. of Electrical Engineering, University of Pennsylvania (December 1987).

Waxman1985b.

A. M. Waxman and S. Ullman, ''Surface Structure and 3-D Motion From Image Flow: A Kinematic Analysis,'' *Intl. Journal of Robotics Research* **4** pp. 72-94 (1985).

Longuet-Higgins1980a.

H. Longuet-Higgins and K. Prazdny, ''The Interpretation of a Moving Retinal Image,'' *Proc. Royal Soc. London* **B 208** pp. 385-397 (1980).

Burt1988a.

P. J. Burt, ''Smart Sensing within a Pyramid Vision,'' *IEEE Proceedings* **76**(8) pp. 1006-1025 (August 1988).

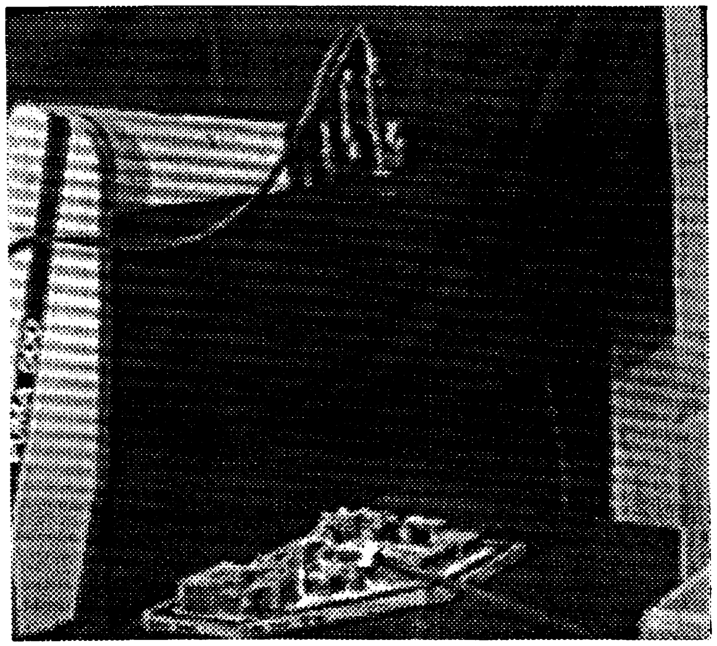


Figure 1.
The experimental setup. The robot manipulator tracks an object which moves against the campus model.

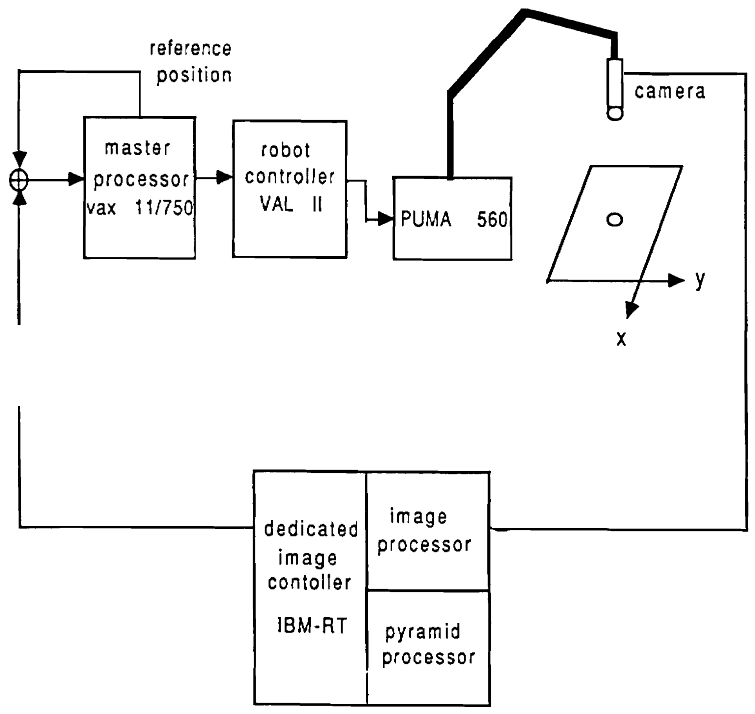


Figure 2.
The schematic diagram of the tracking system.

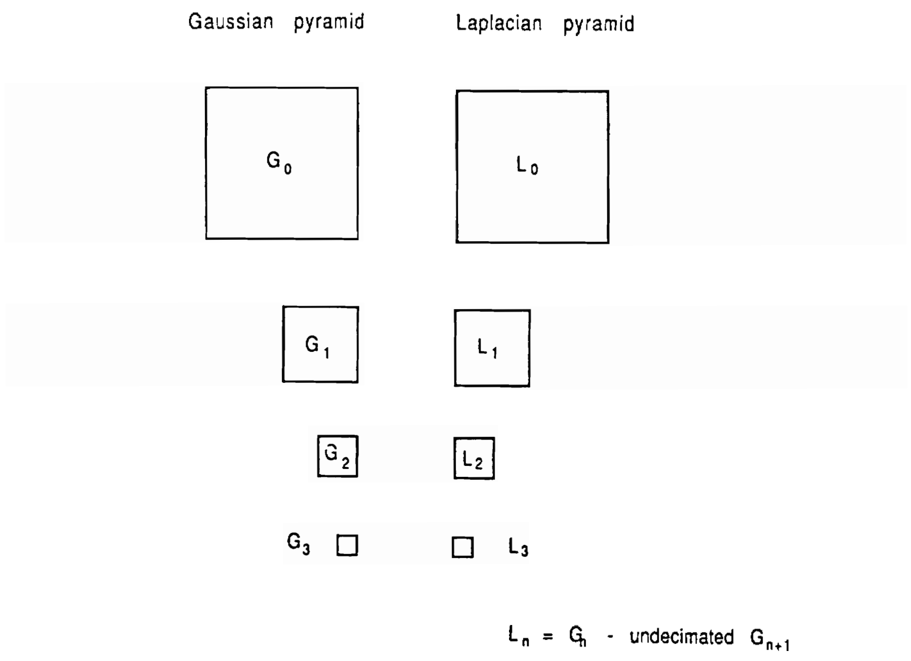


Figure 3.
Gaussian and Laplacian pyramids constructed by the PVM-1.

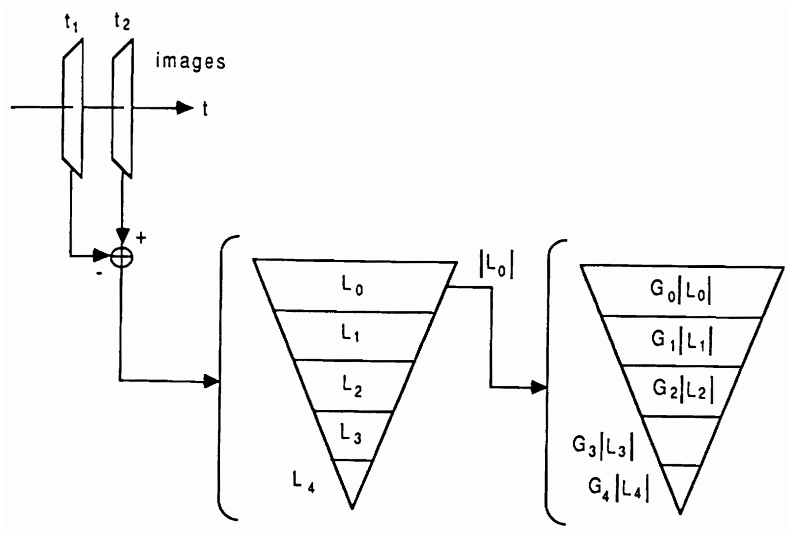


Figure 4.
Target detection is carried out by differencing two consecutive frames. The Gaussian and Laplacian pyramids are constructed to locate the target in real-time.

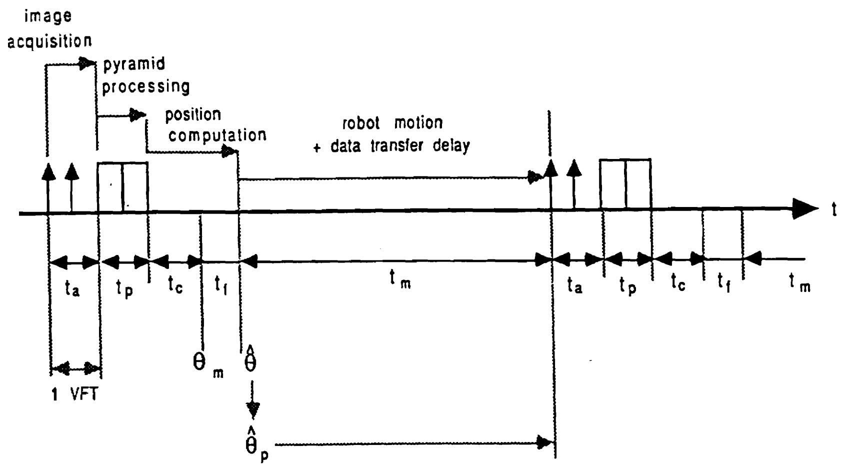


Figure 5.
The basic timing diagram of the tracking system.

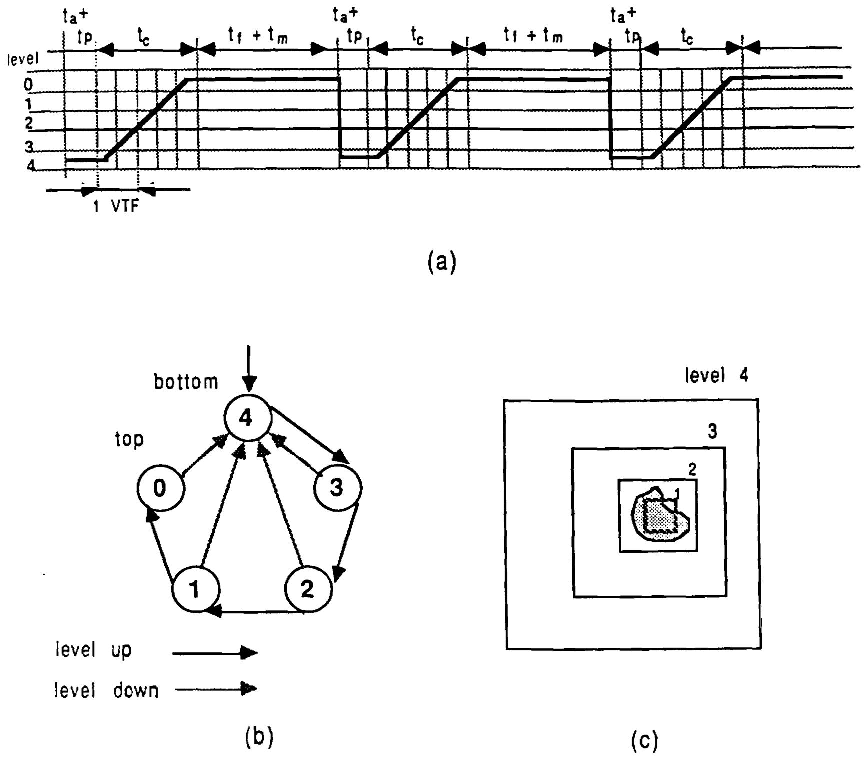
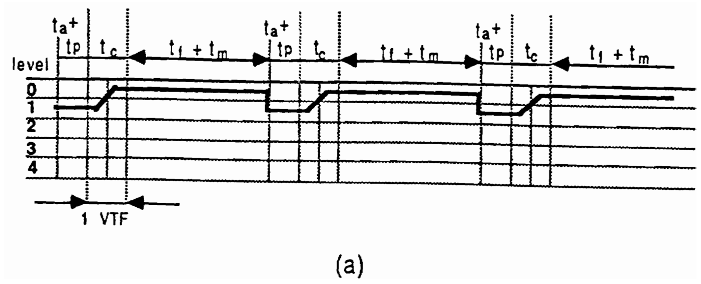


Figure 6.
The 5-level searching. (a) The timing diagram of level transition for a level-0 object. (b) The level transition. (c) The window converging to the target.



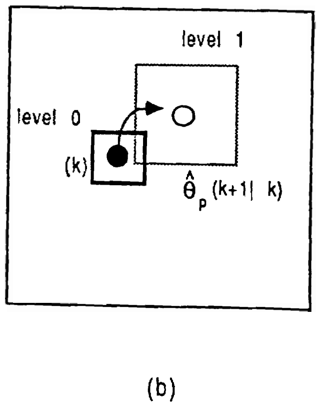


Figure 7.
The 2-level searching in the normal operational mode. (a) The timing diagram of level transition for a level-0 object. (b) The window predicting the target position.

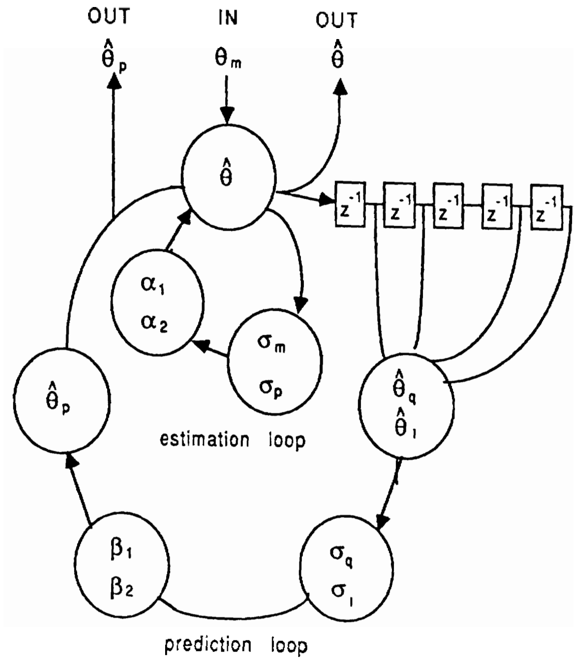
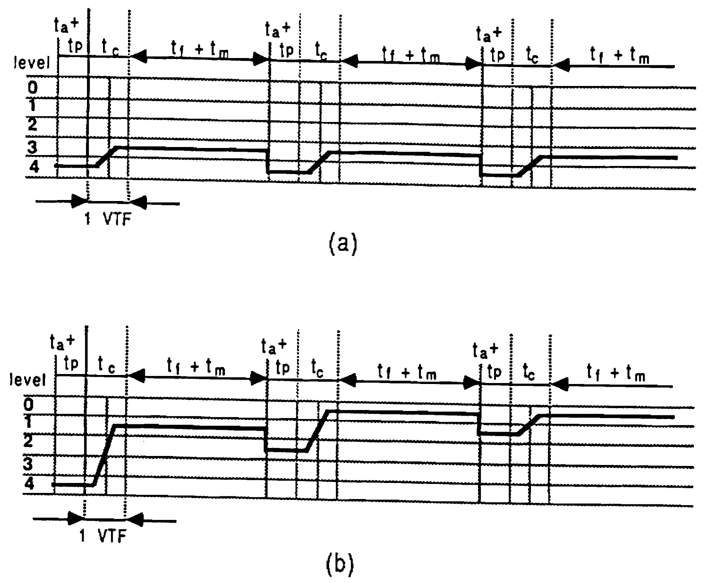


Figure 10.
The procedure for the position estimation and prediction.



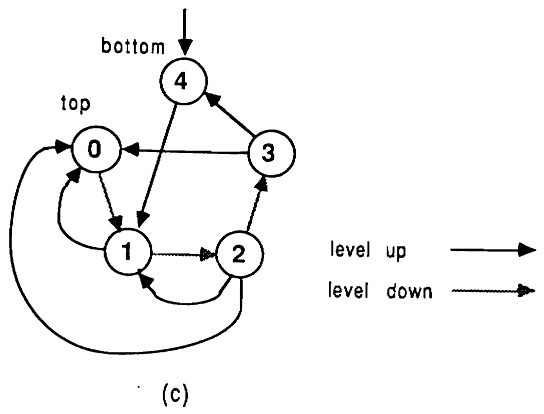


Figure 8.

The 2-level searching with the bootstrapping. (a) The timing diagram without the bootstrapping provision. (b) The timing diagram with the bootstrapping. (c) The level transition.

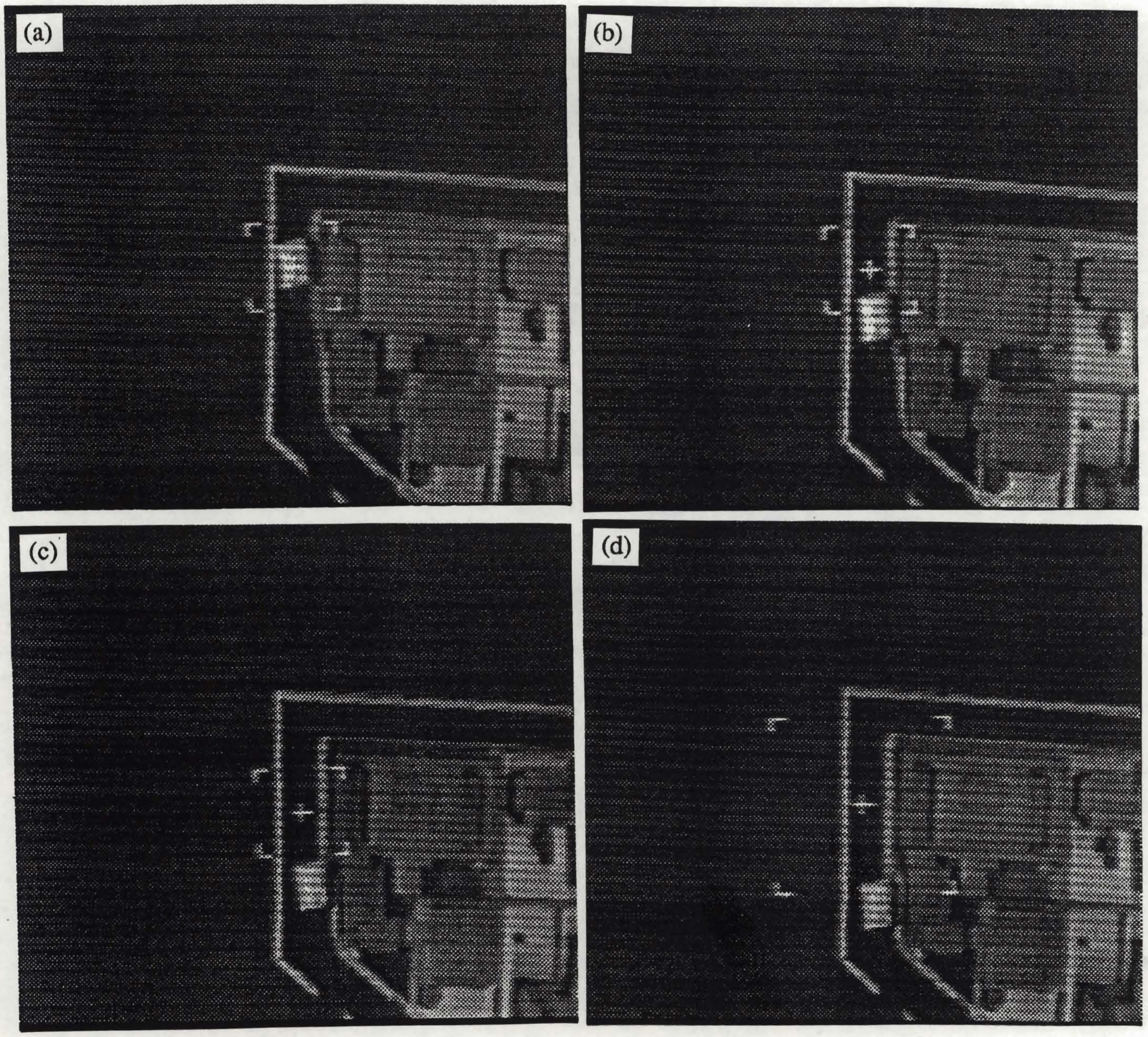


Figure 9.
The performance of 2-level searching when the target moves too fast. The searching window keeps increasing until it looses the target completely. Four frames are shown.

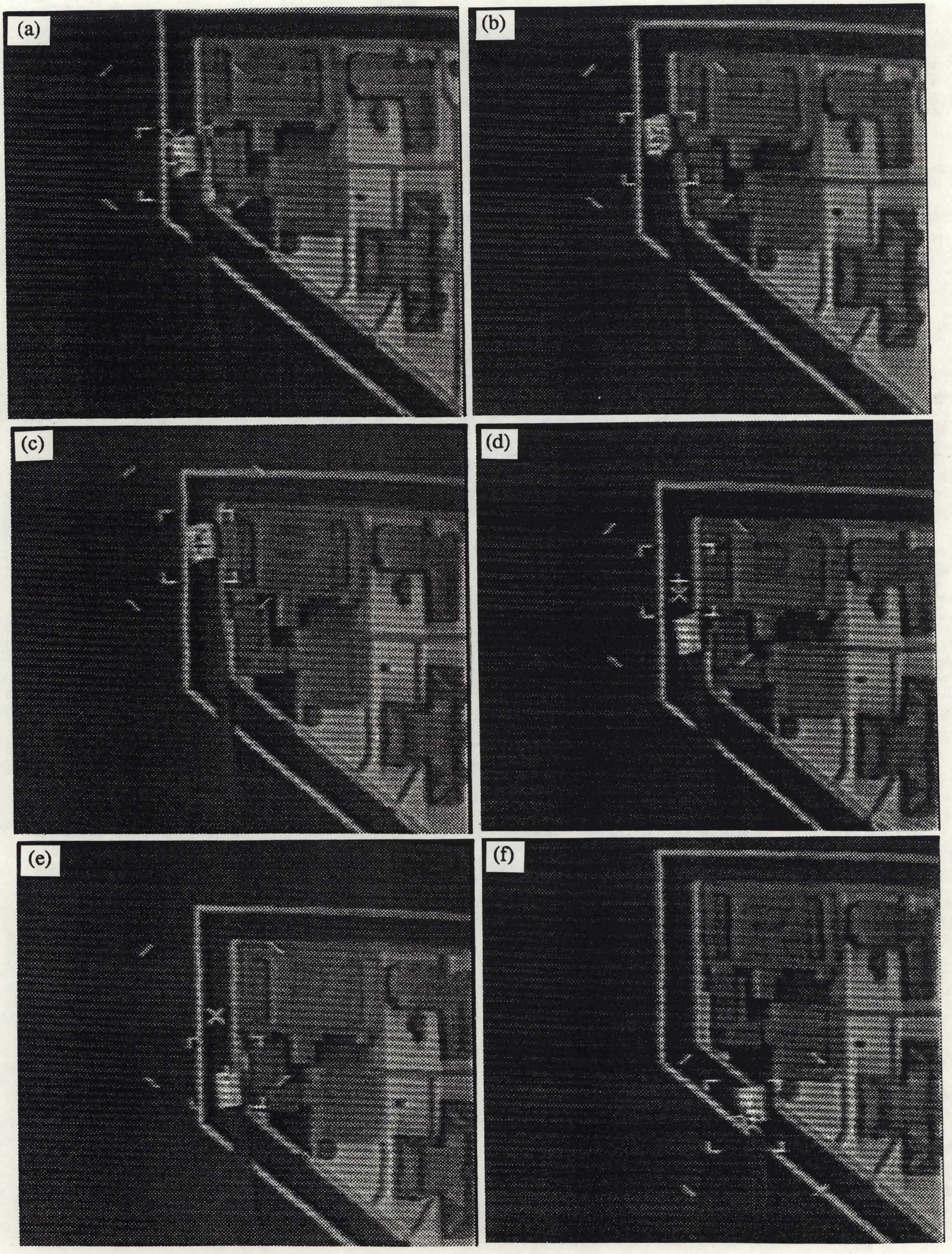


Figure 11.

The performance of 2-level searching with the estimation and prediction. Six

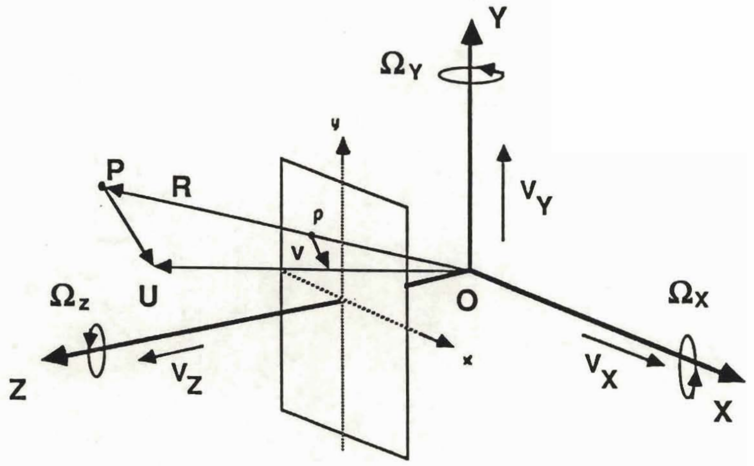


Figure 12.
The camera coordinate system under the 3-D motion.

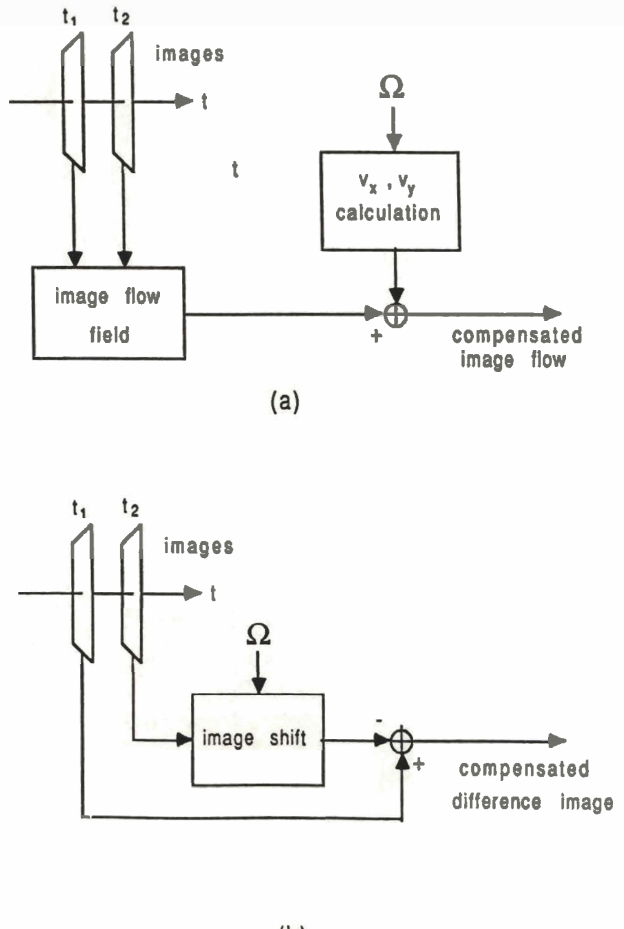


Figure 13.
Compensation for the camera motion. (a) The motion compensation prior to the flow estimation. (b) The motion compensation posterior to the image differencing.