Motion Tracking with an Active Camera

Don Murray and Anup Basu, *Member, IEEE*

Abstract—**This work describes a method for real-time motion detection using an active camera mounted on a pan/tilt platform. Image mapping is used to align images of different viewpoints so that static camera motion detection can be applied. In the presence of camera position noise, the image mapping is inexact and compensation techniques fail. The use of morphological filtering of motion images is explored to desensitize the detection algorithm to inaccuracies in background compensation. Two motion detection techniques are examined, and experiments to verify the methods are presented. The system successfully extracts moving edges from dynamic images even when the pan/tilt angles between successive frames are as large as 3°.**

I. Introduction

MOTION detection and tracking are becoming increasingly recognized as important capabilities in any vision system designed to operate in an uncontrolled environment. Animals excel in these areas, and that is because motion is inherently interesting and important. In any scene, motion represents the dynamic aspects. For animals, motion can mean either food or danger, matters of life and death. For a mobile robot platform, motion can imply a chance of collision, dangers to navigation, or alterations in previously mapped regions.

Tracking in computer vision, however, is still in the developmental stages and has had few applications in industry. It is hoped that tracking combined with other technologies can produce effective visual servoing for robotics in a changing work cell. For example, recognizing and tracking parts on a moving conveyor belt in a factory would enable robots to pick up the correct parts in an unconstrained work atmosphere.

In this work, we will consider tracking with an *active camera. Active vision* [4], [5], [2] implies computer vision implemented with a movable camera, which can intelligently alter the viewpoint so as to improve the performance of the system. An active camera tracking system could operate as an automatic cameraman for applications such as home video systems, surveillance and security, video-telephone systems, or other tasks that are repetitive and tiring for a human. Recently, it has been shown that tracking facilitates motion estimation [11].

Manuscript received October 13, 1992; revised May 27, 1993. This work was supported in part by the Canadian Natural Sciences and Engineering Research Council under Grant OGP010539, and by a fellowship from the province of Alberta. Recommended for acceptance by Associate Editor Y. Aloimonos.

D. Murray is with MacDonald Detwiler, 13800 Commerce Parkway, Richmond, BC, V6V 2J3 Canada.

A. Basu is with the Computing Science Department, University of Alberta, Edmonton, AB, Canada T6G 2H1 and with Telecommunications Research Labs, Edmonton, AB, Canada T5K 2P7.

IEEE Log Number 9215856.

Section II contains a brief overview of some of the previous work in topics related to tracking. A description of the tracking system used in this work is given in Section III. The relationship between camera frame positions and pixel locations at different pan/tilt orientations are investigated in Section IV. Section V explains the methods of motion detection that were explored and developed. In Section VI, we show the results of our motion detection algorithms for a sequence of real images. Section VII discusses some of the limitations imposed on the system by synchronization error and noise filtering.

II. Previous Work

In general, there are two approaches to tracking that are fundamentally different. These are recognition-based tracking and motion-based tracking. Recognition-based tracking is really a modification of object recognition. The object is recognized in successive images and its position is extracted. The advantage of this method of tracking is that it can be achieved in three dimensions. Also, the translation *and* rotation of the object can be estimated. The obvious disadvantage is that only a recognizable object can be tracked. Object recognition is a high-level operation that can be costly to perform. Thus, the performance of the tracking system is limited by the efficiency of the recognition method, as well as the types of objects recognizable. Examples of recognition-based systems can be found in the work of Aloimonos and Tsakiris [3], Bray [8], and Gennery [12], Lowe [16], Wilco *et al.* [26], Schalkoff and McVey [22], and others [24], [9], [21].

Motion-based tracking systems are significantly different from recognition-based systems in that they rely entirely on motion detection to detect the moving object. They have the advantage of being able to track any moving object regardless of size or shape, and so are more suited for our systems. Motion-based techniques can be further subdivided into optic flow tracking methods and motion-energy methods, as described in Sections II-A and II-B.

A. Optic Flow Tracking

The field of retinal velocity is known as *optic flow* [15], [20], [25]. The difficulty with optic-flow tracking is the extraction of the velocity field. By assuming that the image intensity can be represented by a continuous function $f(x, y, t)$, we can use a Taylor series expansion to show that

$$0 = \frac{\partial f}{\partial x}u + \frac{\partial f}{\partial y}v + \frac{\partial f}{\partial t}, \tag{1}$$

where $u = dx/dt$ and $v = dy/dt$ are the instantaneous 2-D velocity at (x, y). This is a convenient equation since $\partial f/\partial t, \partial f/\partial x$ and $\partial f/\partial y$ all can be locally approximated.

The difficulty applying (1) is that we have two unknowns and only one equation. Thus, this equation describes a line on which (u, v) must lie, but we cannot solve for (u, v) uniquely without imposing additional constraints, such as smoothness [10], [13], [23]. Some interesting work has been done with partial optic-flow information from an active camera image sequence to detect motion [18]. The significance of this work is that motion detection is achieved from a fully active camera. The algorithm can be evaluated quickly and hence is a computationally efficient method for detecting motion from an unconstained platform. The drawback is that the evaluation is qualitative, and hence it is less discriminatory than a quantitative approach. In addition, in order for the approximation in (1) to be valid, image points should not move more than a few pixels between successive images. This implies that the pan/tilt angles between consecutive frames must be very small. Our system is not constrained by this restriction. Further discussion on this topic can be found in Section VI and VII.

Since determining a complete optic-flow field quantitatively is both expensive and ill-posed, solving the problem for a few discrete points has been a popular alternative for practical systems [6]. This method relies on identifying points of interest (also known as *features*) in a series of images and tracking their motion [7]. The disadvantage with this technique is that the points of interest in each scene must be matched to those of the previous image, which is generally an intractable problem. The difficulties increase in the case of an active camera. Since the scene viewed is dynamic, certain points will pass beyond the field of view while new ones will enter (drop-ins and drop-outs), which increases the difficulty of searching for matching points. The complexity of this problem makes it unsuitable for real-time applications.

B. Motion Energy Tracking

Another method of motion tracking is motion-energy detection. By calculating the temporal derivative of an image and thresholding at a suitable level to filter out noise, we can segment an image into regions of motion and inactivity. Although the temporal derivative can be estimated by a more exact method, usually it is estimated by simple image subtraction:

$$\frac{df(x,y,t)}{dt} \approx \frac{f(x,y,t) - f(x,y,t-\delta t)}{\delta t}.$$

This method of motion detection is subject to noise and yields imprecise values. In general, techniques for improving image subtraction include spatial edge information to allow the extraction of moving edges. Picton [19] utilized edge strength as a multiplier to the temporal derivative prior to thresholding. Allen *et al.* [1] used zero crossings of the second derivative of the Gaussian filter as an edge locator and combined this information with the local temporal and spatial derivatives to estimate optic flow.

For practical, real-time implementations of motion detection, image subtraction combined with spatial information is the most widely used and successful method. In addition to computational simplicity, motion-energy detection is suitable for pipeline architectures, which allow it to be readily implemented on most high-speed vision hardware. One disadvantage of this method is that pixel motion is detected but not quantified. Therefore, one cannot determine additional information, such as the focus of expansion. Another disadvantage is that the techniques discussed are not suitable for application on active camera systems without modification. Since active camera systems can induce *apparent motion* on the scences they view, compensation for this apparent motion must be made before motion-energy detection techniques can be used.

III. Notation and Modeling

A. Notation

Since we often discuss point locations in both two and three dimensions, it is important to differentiate between them. A location in 3-D is written symbolically in capital letters as (X, Y, Z) or is presented as a column vector P, where

$$P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Two-dimensional points are written in lower case, such as (x, y).

Arbitrary homogeneous transformations are formulated as a 4×4 matrix T, where

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & \rho_x \\ r_{21} & r_{22} & r_{23} & \rho_y \\ r_{31} & r_{32} & r_{33} & \rho_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the pan/tilt camera parameters,

- f is the focal length of the camera,
- θ is the tilt angle from the level position,
- α is a small angle of rotation about the pan axis, and
- γ is a small angle of rotation about the tilt axis.

B. Pin-Hole Camera Model

Throughout this work, the pin-hole camera model is used. As shown in Fig. 1, let $OXYZ$ be the camera coordinate system. The image plane is perpendicular to the Z-axis and intersects it at a point $(0, 0, f)$, where f is the focal length. Using this model, the relationships between points in the image plane and points in the camera coordinate system are

$$x = f\frac{X}{Z} \quad y = f\frac{Y}{Z}$$

C. Pan/Tilt Model

The active camera considered in this work is mounted on a pan/tilt device that allows rotation about two axes. Fig. 2 shows the schematics of such a device. The Cohu-MPC system is show in Fig. 3. The reference frame for each camera position is formed by the intersecting axes of rotation (pan = Y -axis, tilt = X-axis). The origin of the camera coordinate system is

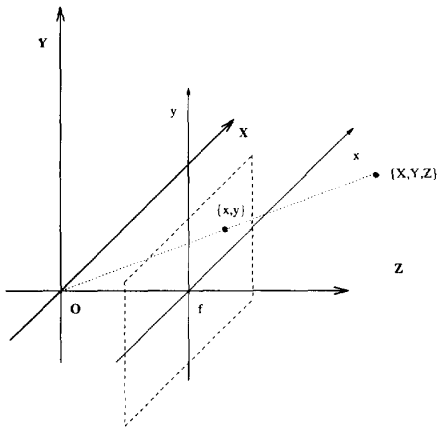


Fig. 1. Pin-hole camera model.

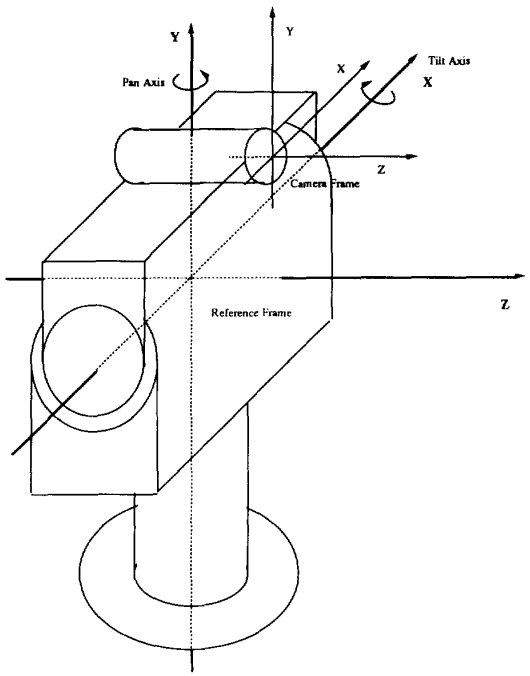


Fig. 2. Schematics of pan/tilt device.

located at the lens centre, which is related to the reference frame by a homogeneous transformation T_c such that

$$P_r = T_c P_c,$$

where P_r and P_c are 3-D points in the reference and camera frames, respectively.

For an arbitrary tile angle θ, the camera transformation T_c is

$$T_c(\theta) = Rot_X(\theta) T_{c|\theta=0} = \begin{bmatrix} 1 & 0 & 0 & \rho_X \\ 0 & c\theta & -s\theta & c\theta\rho_Y - s\theta\rho_Z \\ 0 & s\theta & c\theta & s\theta\rho_Y + c\theta\rho_Z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

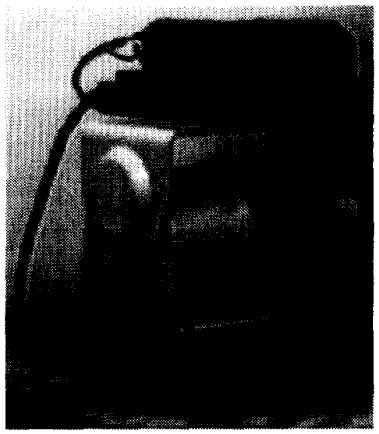


Fig. 3. Cohu-MPC pan/tilt device.

where

$$T_{c|\theta=0} = \begin{bmatrix} 1 & 0 & 0 & \rho_X \\ 0 & 1 & 0 & \rho_Y \\ 0 & 0 & 1 & \rho_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and c and s are used to abbreviate cos and sin, respectively.

IV. Background Compensation

To be able to apply the motion detection techniques to be introduced in Section V, we must compensate for the apparent motion of the background of a scene caused by camera motion. Our camera is mounted on a pan/tilt device and hence is constrained to rotate only. This is ideal for background compensation, since visual information is invariant to camera rotation [14].

Our objective in background compensation is to find a relationship between pixels representing the same 3-D point in images taken at different camera orientations. The projection of a 3-D point on the image plane is formed by a ray originating from the 3-D point and passing through the lens center. The pixel representing this 3-D point is given by the intersection of this ray with the image plane (see Fig. 1). If the camera rotates *about the lens center,* this ray remains the same, since neither endpoint (the 3-D point and the lens center) moves due to this rotation. Consequently, no previously viewable points will be occluded by other static points within the scene. This is important, since it implies that there is no fundamental change in information about a scene at different camera orientations. It should be noted that, for theoretical considerations, the effect of the image boundary is ignored here. Obviously, regions which pass outside the image due to camera motion cannot be recovered. For camera rotation, the only components of the system that move are the camera coordinate system and the image plane. An example of this motion is shown in Fig. 4.

The relationship between every pixel position in two images, taken from different positions of rotation about the lens center, has been derived by Kanatani [14]. In the Appendix, we show that for small displacements of the lens center from the center

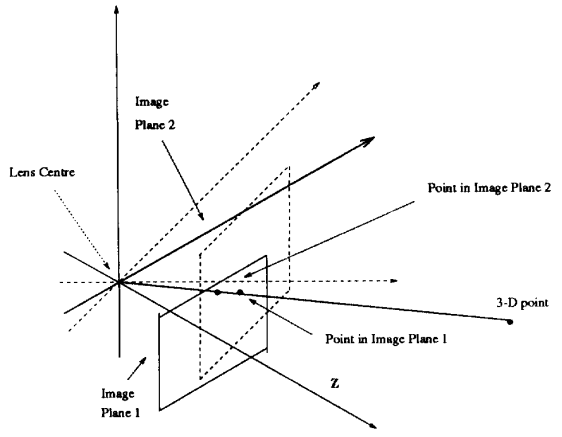


Fig. 4. 3-D point projected on two image planes with the same lens center.

of rotation, Kanatani's relationship remains valid. For an initial inclination (θ) of the camera system and pan and tilt rotations of α and γ, respectively, this relationship is

$$x_{t-1} = f\frac{x_t + \alpha \sin\theta y_t + f\alpha\cos\theta}{-\alpha\cos\theta x_t + \gamma y_t + f}$$

$$y_{t-1} = f\frac{-\alpha\sin\theta x_t + y_t - f\gamma}{-\alpha\cos\theta x_t + \gamma y_t + f},$$

where f is the focal length.

With knowledge of f, θ, γ, and α, for every pixel position (x_t, y_t) in the current image we can calculate the position (x_{t-1}, y_{t-1}) of the corresponding pixel in the previous image.

V. Independent Motion Detection

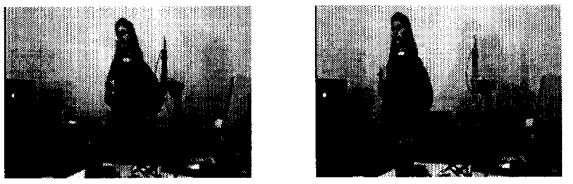
As mentioned in Section II, motion-energy detection has been demonstrated as a successful and practical motion detection approach for real-time tracking. Our implement is therefore based primarily on this technique. Yet, because of the potential error incurred during camera motion compensation, modifications must be made to the motion detection methods. In this section we first discuss, in some detail, motion-energy detection. Then we describe what measures are taken in order to modify these techniques to achieve camera systems.

A. Motion Detection with a Static Camera

In practice, motion-energy detection is implemented through spatio-temporal filtering. The simplest implementation of motion-energy detection is image subtraction. In this method, each image has the previous image in the image sequence subtracted from it, pixel by pixel. This is an approximation of the temporal derivative of the sequence. The absolute value of this approximation is taken and thresholded at a suitable level to segment the image into static and dynamic regions. Fig. 5 shows two frames of an image sequence taken with a static camera. Fig. 6 (left) shows the result of image subtraction.

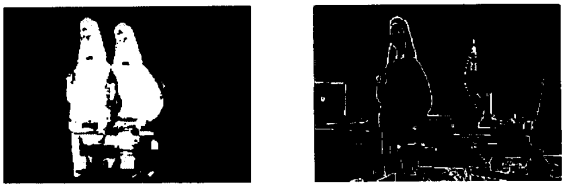
The drawback of this technique is that motion is detected in regions where the moving object was present at times t and $t - \delta t$. This means that the center of the regions of motion is close to the midpoint between the actual positions of the object at t and $t - \delta t$. For systems with a fast sampling rate (small δt) compared to the speed of the moving object, the difference in position of the object between frames will be small, and hence the midpoint between them may be adequate for rough position estimates. For objects with high speeds relative to the sampling rate, we must improve this method. Our aim is to estimate the position of the moving object at time t. This can be achieved by the following steps. First, we obtain a binary edge image of the current frame by applying a threshold to the output of an edge detector. (An example of the resultant edge image is shown in Fig. 6(b).) Then this information is incorporated into the subtracted image by performing a logical AND operation between the two binary images, i.e., the edge image and the subtracted image. This highlights the edges within the moving region to obtain the *moving edges* within the latest frame. Fig. 7(a) shows the result of this operation. As can be seen, edges are also highlighted in the area previously occluded by the moving object. Since these edges have only been viewed for one sample instant, it is unreasonable to expect the system to be able to detect whether or not they are moving, until the next image is taken and processed.

A modified approach was suggested by Picton [19]. He argued that thresholds are empirically tuned parameters and, in order to keep the system as simple and robust as possible, the number of tuned parameters should be minimized. Hence, he proposed reducing thresholding to a single step by multiplying the prethreshold values of the edge strength and image subtraction to obtain a value indicative of both edge strength and temporal change. This product is then thresholded, and



(a) (b)

Fig. 5. Static camera sequence.

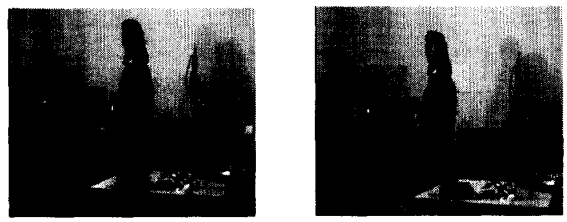


(a) (b)

Fig. 6. Image subtraction and edge images.

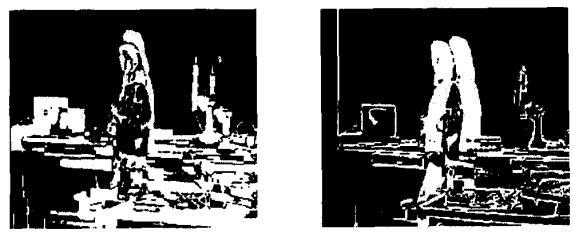
(a) (b)

Fig. 7. Moving edges detected in frame 2 with the two motion detection techniques.



(a) (b)

Fig. 8. Moving camera sequence.



(a) (b)

Fig. 9. Image subtraction with and without compensation.

thus the tuned parameters are reduced to a single threshold value. The result of this multiplication method is shown in Fig. 7(b). As we can see, the boundary of the moving object is the same as with the logical AND method. However, with Picton's method more interior edges are emphasized.

B. Motion Detection by an Active Camera

For a stationary camera, the pixel-by-pixel subtraction described in Section V-A is possible since, with a static scene, a given 3-D point will continuously project to the same position in the image plane. For a moving camera this is not the case. To apply pixel-by-pixel comparison with an active-camera image sequence, we must map pixels that correspond to the same 3-D point to the same image plane position.

Section IV outlined the geometry behind invariance to rotation and derived the mapping function between images. For each pair of images processed in the image sequence, the image at time $t - \delta t$ is mapped so as to correspond pixel by pixel with the image at time t. Regions with no match between the two images are ignored. The image subtraction, edge extraction, and subsequent moving edge detection are performed as in the static case.

Fig. 8 shows two images taken from different camera orientations. Fig. 9(a) shows the results of image subtraction without compensation. Clearly the apparent motion caused by camera rotation has made the static camera methods unsuitable. The object of interest appears to be moving less than the background. Fig. 9(b) shows the results after background compensation. Notice that the background has *not* been entirely eliminated. This is due to inaccuracies in the inputs to the compensation algorithm and approximations made in the algorithm derivation.

The background compensation algorithm was derived with the assumption that rotation occurs about the lens center. In reality this is not the case for our system, and the small amount of camera translation corrupts the compensation method. Also, errors in the pan/tilt position sensors and camera calibration contribute to the compensation inaccuracy. The following section shows how we overcome this compensation noise and improve the robustness of our method.

C. Robust Motion Detection with an Active Camera

If we could achieve exact background compensation, the methods described so far would be sufficient. In the presence of position inaccuracies, however, the results deteriorate rapidly. We use edge information in our techniques to detect moving objects. Ironically, regions with good edge characteristics are the most sensitive to compensation errors during image subtraction. That is, false motion caused by inaccurate compensation will be greatest in strong edge regions, yet these are the very regions that are considered as candidates for moving edge pixels. This problem makes the method discussed unreliable.

Since errors in angle information are inevitably present, it is desirable to develop methods of motion detection that can robustly eliminate the false motion they cause. Errors in pan/tilt angles can be due to sensor error. For a real-time system with a continuously moving camera, there is the additional problem of synchronization. If the instances of grabbing an image and reading position sensors are not perfectly synchronized, the finite difference in time between these events can be considered as error in position sensing. This error is calculated as

$$\theta_e = \omega \times \Delta t,$$

where θ_e is the error in angular position, ω is the angular velocity of rotation, and Δt is the synchronization error. Since few vision systems are designed with this consideration in mind, the problem is a common one in active vision applications.

Fig. 9(a) shows an example of the results of image subtraction after inaccurate background compensation. Notice the region of the moving object contains a broad area where the true motion was detected, whereas false motion is characterized by narrow bands bordering the strong edges representing the background. Our approach to removing the false motion utilizes the expectation of a wide region of true motion being present. Using morphological erosion and dilation (morphological *opening*), we eliminate narrow regions of detected motion while preserving the original size and shape of the wide regions. The method of robust motion detection is shown in a block diagram in Fig. 10.

Morphological Filtering: Morphological filters applied to digital images have been used for several applications, including edge detection, noise suppression, region filling, and skeletonizing [17]. Morphological filtering is essentially an application of set theory to digital signals. It is implemented with a mask M overlaying an image region I. For morphological filtering, the image pixel values, namely

$$\begin{bmatrix} i_{11} & \cdots & i_{1n} \\ \vdots & \ddots & \vdots \\ i_{n1} & \cdots & i_{nn} \end{bmatrix},$$

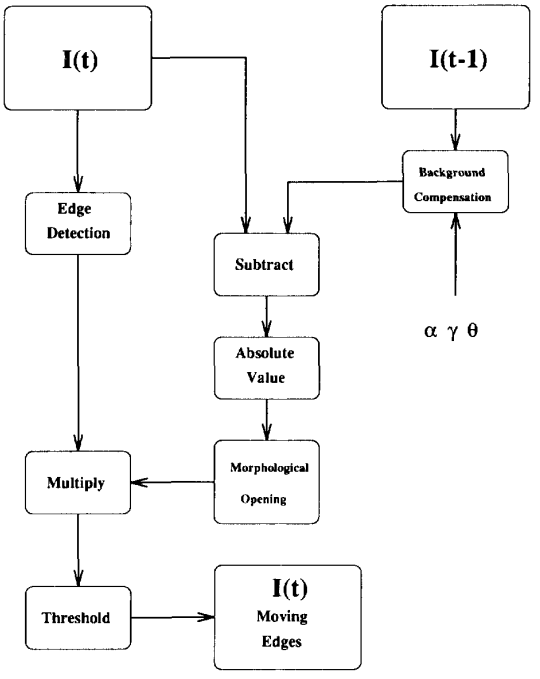


Fig. 10. Block diagram of method for robust motion detection.

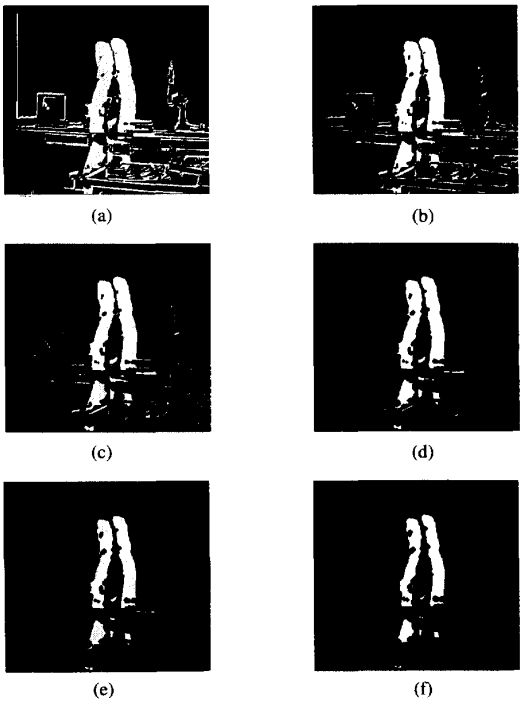


Fig. 11. Subtracted image with various sizes of erosion masks applied.

are selected by the values of M as members of a set for analysis with set theory methods. Usually, values of elements in a morphological filter mask are either 0 or 1. If the value of an element of the mask is 0, the corresponding pixel value is not a member of the set. If the value is 1, the pixel value is included in the set.

We can express the elements of the image region selected by the morphological mask as a set A such that

$$A = \{i_{jk} | 1 \leq j \leq n, 1 \leq k \leq n, m_{jk} = 1\}.$$

The morphological operations we will consider are *erosion* and *dilation*.

Erosion of A is: $E_A = \min(A)$.

Dilation of A is: $D_A = \max(A)$.

By applying erosion to the subtracted image, narrow regions can be eliminated. If the regions to be preserved are wider than the filter mask, they will only be thinned, not completely eliminated. After dilation by a mask of the same size, these regions will be roughly restored to their original shape and size. If the erosion mask is wider than a given region, that region will be eliminated completely and not appear after dilation.

Fig. 11 shows the subtracted image of Fig. 9 (right) eroded by different size masks. For this particular image sequence, we can see that to completely eliminate the noise due to position inaccuracies we must use a mask size of 11×11.

VI. Experimental Results

The experimental results presented here use a sequence of images taken with the pan/tilt device. The image sequence is processed off-line; however, all modules except robust background compensation have already been implemented in real time. Two methods of motion detection were tested: thresholding the temporal and spatial derivatives independently, and multiplication of derivatives prior to thresholding. The image sequences and the processed results are presented in this section.

The camera is mounted on the Cohu-MPC, a pan/tilt device. Instructions for this device are sent from a Sun3 over a serial interface. The Sun3 is mounted on a VMEbus with a real-time frame digitizer board. The camera is a CCD device with standard video output. The Cohu-MPC allows controls of rotation about two axes (pan and tilt) as well as adjustment of zoom and focus settings. The position sensing of the pan/tilt axes is done by a potentiometer coupled to each driving motor shaft.

Fig. 12 shows an image sequence taken with the pan/tilt device. Figs. 13 and 14 show the results of motion detection methods applied to the image sequence with an 11×11 morphological mask. The results shown have been generated using the two motion detection techniques presented in Section V. The two approaches are summarized as follows.

Approach 1: Binary images of the spatial and temporal derivative peaks are formed by thresholding the subtracted and edge strength images. These two binary images are then ANDed together to extract the moving edges in the scene.

Approach 2: The values of the spatial and temporal derivatives are multiplied, and the product is thresholded to extract the moving edges.

Both approaches use two 3×3 sobel edge detection kernels to find the edge strength in the vertical and horizontal directions.

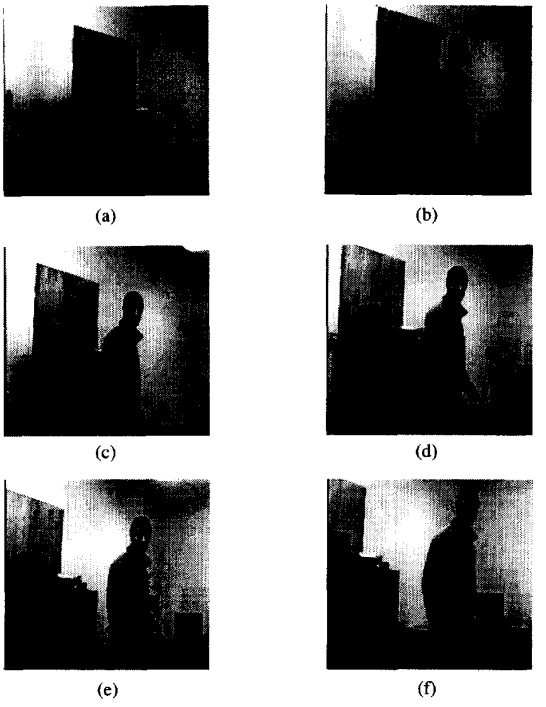


Fig. 12. Pan/tilt image sequence.

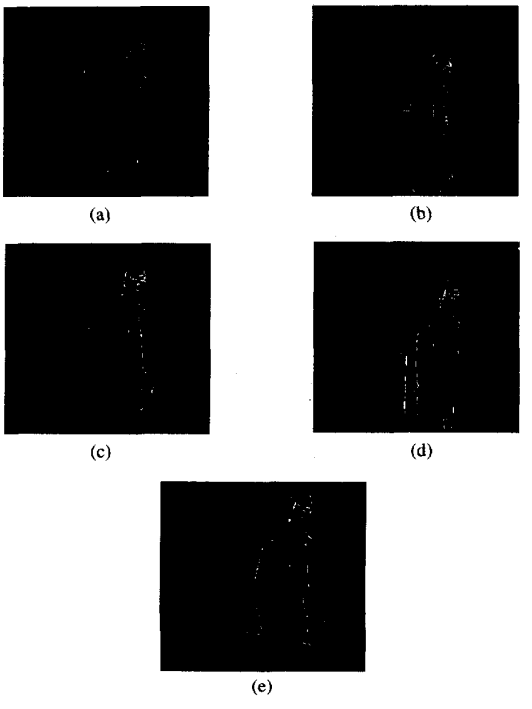


Fig. 13. Moving edges in pan/tilt image sequence (Approach 1).

Although both approaches work reasonably well, they exhibit significantly different characteristics. Approach 1 detects primarily the boundary of the moving object, since the independent thresholding of the edge image tends to eliminate edges within the contour of the object. Approach 2 does not remove the contribution of the fainter edges until after multiplication with the temporal derivative. Thus we find that many interior edges of the moving object are revealed.

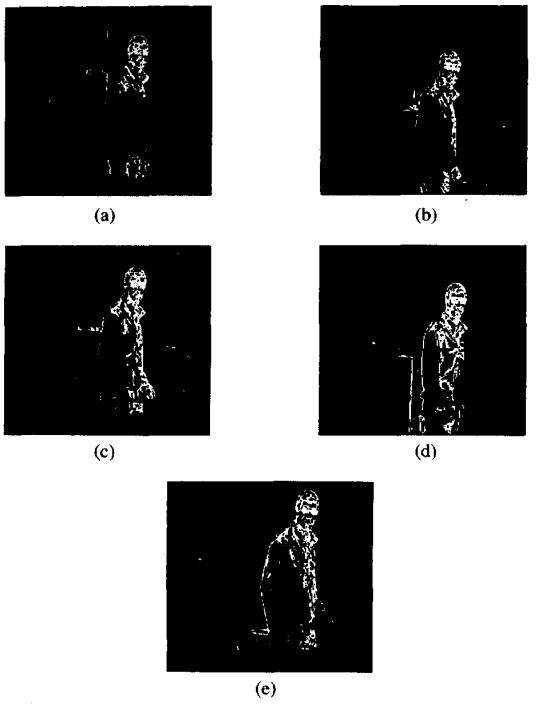


Fig. 14. Moving edges in pan/tilt image sequence (Approach 2).

By considering the centers of motion produced by the two motion detection techniques, we see that the actual position estimates are nearly identical. However, the methods do yield different moving edge images, and there is potential for significantly different results in certain conditions. In the image sequence, the moving object has rich texture due to the folds in clothing, while the background is characterized by homogeneous blocks of similar intensity broken by strong edges. The texture provides a dense area of weak edges that can be brought out by Approach 2, which subjectively seems to be the preferred approach for this image sequence. Yet in the results of both approaches, background edges that were occluded by the moving object in the previous frame are detected as moving edges. Since our system has only viewed these regions for a single frame, it is unable to determine whether these edges are static or dynamic until the next frame is processed. If the background had more varied texture, such as a wheat field or a chain-link fence, regions previously blocked by the moving object would have the same weak edges brought out by Approach 2. This would corrupt the moving edge signal and tend to produce a centroid of the moving object that lags the objects' true position.

In the results presented here, an 11×11 mask was used. It was found empirically that for a 9×9 mask false motion increases marginally, while for masks of size 7×7 and smaller, for this image sequence and position data, the motion detection degrades severely. The false motion detected in

both image sequences is due to either previous occlusion (as discussed above) or inaccurate background compensation. To eliminate false motion caused by inaccurate background compensation, it is important to select the appropriate sized filter for noise removal. However, an increase in filter size places additional computational burden on the filtering stage, as well as possibly eliminating the true motion signal. The relationship between position noise and filtering requirements is presented in Section VII.

VII. Analysis of Compensation Inaccuracy

As discussed in Section V, noisy position information corrupts the background compensation algorithm and necessitates additional noise removal. Morphological filtering has been presented as a technique to remove narrow regions of false motion from subtracted images. For effective noise removal to occur, the morphological erosion mask must be at least as wide as the regions of false motion. If the mask is not wide enough, some noise will remain after erosion and will be expanded to its original size during dilation. This means that no noise will be removed. This method of noise removal is therefore an *all-or-nothing* approach. The advantage of this scheme is that for acceptable noise levels, false motion is completely removed. However, if the noise exceeds the filter capacity, no noise removal takes place. Because of this behavior, it is important to use filters large enough to completely remove the expected noise. At the same time, for computational reasons, it is desirable to limit filtering to the minimum required. This motivates us to investigate the relationship of noise characteristics to filtering requirements.

Recall the mapping algorithm from Section IV

$$x_{t-1} = f\frac{x_t + \alpha \sin\theta y_t + f\alpha\cos\theta}{-\alpha\cos\theta x_t + \gamma y_t + f} \tag{2}$$

$$y_{t-1} = f\frac{-\alpha\sin\theta x_t + y_t - f\gamma}{-\alpha\cos\theta x_t + \gamma y_t + f}. \tag{3}$$

The error between the correct pixel position and that found with inaccurate angle information in the mapping algorithm can be expressed as

$$x_e = x_{t-1}(\alpha,\gamma) - x_{t-1}(\alpha+\Delta_\alpha, \gamma+\Delta_\gamma) \tag{4}$$

$$y_e = y_{t-1}(\alpha,\gamma) - y_{t-1}(\alpha+\Delta_\alpha, \gamma+\Delta_\gamma), \tag{5}$$

where x_e and y_e are the errors in mapped pixel position in the x and y directions, and Δ_α and Δ_γ are inaccuracies in measurement of the rotations α and γ, respectively.

For evaluating the error in pixel mapping, we consider several cases depending on the location of (x_t, y_t). In general, the error in the mapped pixel position is greater as we move farther from the center of the image. We use pixel positions in the image center to simplify the error equations when determining general error chracteristics, and border pixels to determine the worst-case behavior. To simplify our discussion θ is constrained to 0, i.e., the camera is assumed to be at the level position.

A. Compensation Error for Pan-Only Rotation

For the pan-only case, the tilt rotation (γ), is 0. Hence, (2) and (3) are reduced to

$$x_{t-1} = f\frac{x_t + f\alpha}{f - \alpha x_t},$$

$$y_{t-1} = f\frac{y_t}{f - \alpha x_t}.$$

To evaluate $x_{t-1}(\alpha+\Delta_\alpha)$ and $y_{t-1}(\alpha+\Delta_\alpha)$, we will approximate the function with a first-order Taylor series expansion,

$$x_{t-1}(\alpha+\Delta_\alpha) = x_{t-1}(\alpha) + \frac{\partial x_{t-1}}{\partial\alpha}\Delta_\alpha \tag{6}$$

$$y_{t-1}(\alpha+\Delta_\alpha) = y_{t-1}(\alpha) + \frac{\partial y_{t-1}}{\partial\alpha}\Delta_\alpha. \tag{7}$$

Substituting (6) and (7) into our equations for error in the compensated pixel position ((4) and (5)), we obtain

$$x_e = \frac{\partial x_{t-1}}{\partial\alpha}\Delta_\alpha = f\frac{x_t^2 + f^2}{(f-\alpha x_t)^2}\Delta_\alpha \tag{8}$$

$$y_e = \frac{\partial y_{t-1}}{\partial\alpha}\Delta_\alpha = f\frac{y_t x_t}{(f-\alpha x_t)^2}\Delta_\alpha. \tag{9}$$

The magnitude of the error in pixel position e is shown in Fig. 15 and can be expressed as

$$e^2 = x_e^2 + y_e^2. \tag{10}$$

For pan-only rotation, the error is predominantly in the x direction, since the change in the y component for pixels at different viewpoints is affected by changes in perspective only. Therefore,

$$e^2 \approx x_e^2 \quad e \approx x_e.$$

To determine the pan-angle error Δ_α for a given pixel mapping error, from (8) we obtain

$$\Delta_\alpha = \frac{x_e(f-\alpha x_t)^2}{f(x_t^2 + f^2)}.$$

Once Δ_α is determined, we can solve for y_e using (9) to verify our initial assumption that y_e is negligible. For our system, where $f = 890$ and the maximum $x_t = 255$ we can make a table of values Δ_α for given errors x_e, the corresponding y error for this position, y_e, and e for magniude of (x_e, y_e). The value for α used 5°, since this is a good cut-off point for the approximate $\sin\alpha = \alpha$ and hence is the upper bound for which our system is designed. The results are shown in Table I. Note that the relationship between Δ_α and x_e is linear.

B. Compensation Error for Pan and Tilt Rotations

The error in pixel mapping is more difficult to obtain if both pan and tilt rotations are made. However, to gain insight into the general characteristics of the error, we will consider a special case, where $(x, y) = (0, 0)$, which is the pixel that lies directly along the Z-axis of the camera coordinate system. For this case, the pixel mapping functions are

$$x_{t-1} = f\alpha$$

$$y_{t-1} = f\gamma,$$

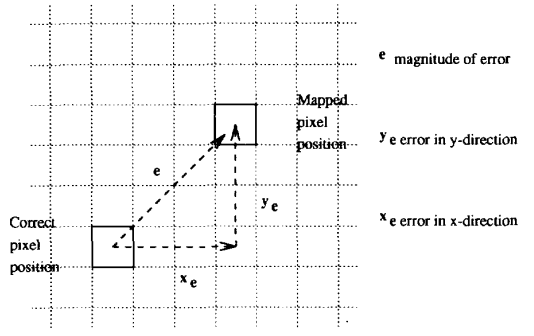


Fig. 15. Magnitude of pixel mapping error.

TABLE I
PAN-ONLY COMPENSATION ERROR

x_e (in pixels)	$\Delta\alpha$ (in degrees)	y_e (in pixels)	e (in pixels)
1	0.057	0.076	1.003
2	0.113	0.152	2.006
3	0.170	0.228	3.009
4	0.226	0.303	4.011
5	0.283	0.379	5.029
6	0.340	0.455	6.017
7	0.396	0.531	7.020
8	0.452	0.607	8.023
9	0.509	0.683	9.026
10	0.566	0.759	10.029
11	0.622	0.835	11.032

and our error equations become

$$x_e = f\Delta_\alpha$$
$$y_e = f\Delta_\gamma.$$

Thus, the magnitude of the error can be expressed by

$$e^2 = x_e^2 + y_e^2 = f^2\Delta_\alpha^2 + f^2\Delta_\gamma^2. \tag{11}$$

Plotting lines of constant error in terms of Δ_α and Δ_γ yields a series of concentric circles with radii of

$$r = \frac{e}{f}$$

for any constant error e.

Unfortunately, the error for the center pixel is not the worst case. To estimate the worst-case error, we use the worst-case x_e with no tilt error and the worst-case y_e with no pan error to determine the Δ_α and Δ_γ intercepts of the constant error curves. For each case, we again use a first-order Taylor series expansion. For x_e this is

$$x_e = \frac{\partial x_{t-1}}{\partial\alpha}\Delta_\alpha + \frac{\partial x_{t-1}}{\partial\gamma}\Delta_\gamma.$$

Since $\Delta_\gamma = 0$ for the Δ_α axis intercept, this simplifies to

$$x_e = \frac{\partial x_{t-1}}{\partial\alpha}\Delta_\alpha = f\frac{x_t^2 + \gamma y_t f + f^2}{(-\alpha x_t + \gamma y_t + f)^2}. \tag{12}$$

Similarly,

$$y_e = \frac{\partial y_{t-1}}{\partial\gamma}\Delta_\gamma = f\frac{\alpha x_t f - y_t^2 - f^2}{(-\alpha x_t + \gamma y_t + f)^2}. \tag{13}$$

TABLE II
WORST-CASE COMPENSATION ERROR

x_e (in pixels)	Δ_α (in degrees)	y_e (in pixels)	Δ_γ (in pixels)
1	0.055	1	0.055
2	0.110	2	0.110
3	0.165	3	0.165
4	0.220	4	0.220
5	0.275	5	0.275
6	0.330	6	0.330
7	0.385	7	0.385
8	0.440	8	0.440
9	0.495	9	0.495
10	0.550	10	0.550
11	0.605	11	0.605

For the worst-case error, $(x_t, y_t) = (255, -255)$. The angles of rotation were set to $\alpha = \gamma = 5°$. Solving for Δ_α and Δ_γ in (12) and (13), we generated Table II. The magnitude of the angle error for given x_e and y_e are the same, which implies that we have a circle again, but with slightly smaller radii. This signifies less angle error for a given error in pixel mapping. For an $n \times n$ morphological filter to remove noise caused by this compensation error e, the filter must be at least as wide as the error, that is

$$n \geq e.$$

C. Significance of Error Analysis

As shown in Section VII-A and VII-B the pixel mapping error is linearly dependent on the magnitude of the error in angle information ($\sqrt{\Delta_\alpha^2 + \Delta_\gamma^2}$). In this section we investigate the consequences of this error and the constraint it places on our system.

Maximum Speed of Tracking: We assume that the primary source of angle error in a real-time implementation is due to synchronization. For a fixed filtering strategy we can determine the upper bound on the speed of rotation for our system, and thus the maximal angular velocity of a target that can be successfully tracked. For rotation with an angular velocity of $\omega_{\max}$, the angular error caused by poor synchronization is

$$\theta_e = \omega_{\max}\Delta t, \tag{14}$$

where Δt is the error in timing. Since compensation error is linearly dependent on angular position error, the compensation error is

$$e = K\theta_e,$$

where K is a constant determined by the system parameters. In the example given in Section VII–B, $K = 1/0.054961$.

For a morphological mask of size $n \times n$, the error tolerance will be n. Thus, the boundary condition is

$$n = K\theta_e. \tag{15}$$

Substituting (14) into (15) and solving for $\omega_{\max}$, we obtain

$$\omega_{\max} = \frac{n}{K\Delta t}.$$

We can see that as synchronization error Δt increases, the maximum possible angular velocity decreases. On the other hand, as the size of the morphological filter n increases, so does $\omega_{\max}$.

Minimum Speed of Tracking: For a moving target with a very slow angular velocity relative to the camera, it is possible that the target will not be detected, since any motion caused by it will be removed by the morphological filter. If we consider a target moving at the slowest detectable speed, $\omega_{\min}$, the angle covered by this target each sample instant t_s will be

$$\theta_{\min} = \omega_{\min} t_s. \tag{16}$$

The distance on the image plane this angle will cover can be calculated by

$$d = f \tan \theta_{\min},$$

thus

$$\theta_{\min} = \arctan \frac{d}{f}.$$

If we are using an $n \times n$ filter mask, the object must move a minimum of $n + 1$ pixels to be identified. This implies

$$\theta_{\min} = \arctan \frac{n+1}{f}. \tag{17}$$

Substituting (16) into (17) and solving for $\omega_{\min}$ yields

$$\omega_{\min} = \frac{\arctan \dfrac{n+1}{f}}{ts}.$$

Thus, to detect a slowly moving object it is desirable to either decrease the filter size n, or increase the sample time t_s.

Filtering and Sampling Strategy: From the above analysis it follows that the desirable filter size is not identical for different moving objects. Ideally, we would like to set the filter size according to the camera motion and the estimated motion of the target. Initially, before any target is acquired, the camera may remain stationary with no filtering required. As an object is tracked and the angular velocity is estimated and predicted, the optimal filtering solution could be determined. The difficulty with this *adaptive* filtering strategy is that to implement different sized filters on a constantly changing basis is not realistically implementable on most pipeline image-processing architecture.

VIII. Conclusion

This work describes methods of tracking a moving object in real time with a pan/tilt camera. With images compensated for camera rotations, static techniques were applied to active image sequences. Since compensation is susceptible to errors caused by poor camera position information, morphological filters were applied to remove erroneously detected motion. A relationship between the level of noise and the size of the morphological filter was also derived. Our technique reliably detected an independently moving object even when the pan/tilt angles between two consecutive frames was as large as 3°.

We have alredy implemented motion detection, position estimation, and camera movements in real time with a MaxVideo20 system and the pan/tilt camera. Currently, we are working on implementing the robust compensation algorithm on the MaxVideo20 to improve continuous tracking results.

In the future, we plan to investigate the application of an active zoom lens in our system. For instance, wide angle is preferable for erratically moving objects, whereas zooming in improves position estimates for a relatively stable object. The use of color images will also be considered to enhance the motion detection algorithm.

Appendix

A. Camera Rotation Compensation Algorithm

In Section IV, the need for a mapping function between pixels of images at different camera orientations was explained. As stated, the equations that achieve this are

$$x_{t-1} = f \frac{x_t + \alpha \sin\theta y_t + f\alpha \cos\theta}{-\alpha \cos\theta x_t + \gamma y_t + f} \tag{18}$$

$$y_{t-1} = f \frac{-\alpha \sin\theta x_t + y_t - f\gamma}{-\alpha \cos\theta x_t + \gamma y_t + f}. \tag{19}$$

Proof: Any point in the reference frame can be expressed as a vector P_r and is related to its position in the camera frame by the transformation

$$P_r = T_c P_c,$$

where P_c is the point in the camera frame and T_c is the 4×4 transform relating the camera frame to the reference frame. The current camera frame, $T_c(t)$, is a result of a pan/tilt rotation of the previous camera position $T_c(t-1)$. Any point in the reference frame can be represented in terms of camera frames before and after motion

$$P_r = T_c(t)P_c(t) = T_c(t-1)P_c(t-1).$$

It follows that the position of a point in one camera frame can be related to its position in the other by

$$P_c(t-1) = T_c(t-1)^{-1} T_c(t) P_c(t). \tag{20}$$

Since $T_c(t)$ is the result of applying pan and tilt rotations to $T_c(t-1)$,

$$T_c(t) = Rot_Y(\alpha) Rot_X(\gamma) T_c(t-1)$$

and

$$T_c(t-1)^{-1} = T_c(t)^{-1} Rot_Y(\alpha) Rot_X(\gamma). \tag{21}$$

By substituting (21) into (20) we

$$P_c(t-1) = T_c(t)^{-1} Rot_Y(\alpha) Rot_X(\gamma) T_c(t) P_c(t). \tag{22}$$

Since the sampling is assumed to be fast, the angles α and γ are assumed to be small. For small angles, we can use the approximations

$$c\alpha, c\gamma \approx 1 \quad s\alpha, s\gamma \approx \alpha, \gamma \quad s\alpha, s\gamma \approx 0.$$

Since we know the current tilt from level position θ, we can determine the current camera transformation and its inverse as

$$T_c(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & -s\theta & \rho_y \\ 0 & s\theta & c\theta & \rho_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

$$T_c(t)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\theta & s\theta & -c\theta\rho_y - s\theta\rho_z \\ 0 & -s\theta & c\theta & s\theta\rho_y - c\theta\rho_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{24}$$

Substituting (23) and (24) into (22) and simplifying the resulting matrix using trigonometric identities, we obtain

$$\begin{aligned} X_{t-1} &= X_t + \alpha s\theta Y_t + \alpha c\theta Z_t + \alpha\rho_z \\ Y_{t-1} &= -\alpha s\theta X_t + Y_t - \gamma Z_t + \gamma s\theta\rho_y - \gamma c\theta\rho_z \\ Z_{t-1} &= -\alpha c\theta X_t + \gamma Y_t + Z_t + \gamma c\theta\rho_y - \gamma s\theta\rho_z. \end{aligned}$$

Using the perspective projection equation, we have

$$x_{t-1} = f\frac{X_t + \alpha s\theta Y_t + \alpha c\theta Z_t + \alpha\rho_z}{-\alpha c\theta X_t + \gamma Y_t + Z_t + \gamma c\theta\rho_y - \gamma s\theta\rho_z}. \tag{25}$$

Now, dividing the top and bottom of the right-hand side of (25) by Z_t and simplifying,

$$x_{t-1} = f\frac{x_t + \alpha s\theta y_t + f\alpha c\theta + \dfrac{f\alpha\rho_z}{Z_t}}{-\alpha c\theta x_t + \gamma y_t + f + f\dfrac{\gamma c\theta\rho_y + \gamma s\theta\rho_z}{Z_t}}.$$

Similarly, the expression for y_{t-1} can be obtained. Assuming that depth is large compared to the other parameters, we can neglect the last terms of both numerator and denominator and obtain (18) and (19).

REFERENCES

[1] P. Allen, B. Yoshimi, and A. Timcenko, "Real-time visual servoing," Tech. Rep. CUCS-035-90, Columbia Univ., Pittsburgh, PA, Sept. 1990.

[2] Y. Aloimonos, "Purposive and qualitative active vision," in *Proc. DARPA Image Understanding Workshop*, Pittsburgh, PA, 1990, pp. 816–828.

[3] Y. Aloimonos and d. Tsakiris, "On the mathematics of visual tracking," *Image Vision Computing*, vol. 9, no. 4, pp. 235–251, 1991.

[4] Y. Aloimonos, I. Weiss, and A. Bandopadhay, "Active vision," *Int. J. Comput. Vision*, vol. 2, pp. 333–356, 1988.

[5] R. Bajcsy, "Active perception," in *Proc. IEEE*, vol. 76, no. 8, pp. 996–1005, 1988.

[6] D. H. Ballard and C. M. Brown, *Computer Vision*. New York: Prentice-Hall, 1982.

[7] B. Bhanu and W. Burger, "Qualitative understanding of scene dynamics for moving robots," *Int. J. Robotics Research*, vol. 9, no. 6, pp. 74–90, 1990.

[8] A. J. Bray, "Tracking objects using image disparities," *Image Vision Computing*, vol. 8, no. 1, pp. 4–9, Feb. 1990.

[9] R. A. Brooks, *Model-based Computer Vision*. Ann Arbor, MI: UMI Research Press, 1984.

[10] J. H. Duncan and T. Chou, "Temporal edges: The detection of motion and the computation of optical flow," in *Proc. Second Int. Conf. Comput. Vision*, Tampa, FL, Dec. 1988, pp. 374–382.

[11] C. Fermueller and Y. Aloimonos, "Tracking facilitates 3-D motion estimation," *Biological Cybern.*, vol. 67, pp. 259–268, 1992.

[12] D. B. Gennery, "Tracking known three-dimensional objects," in *Proc. AAAI 2nd Nat. Conf. on A.I.*, Pittsburgh, PA, 1982, pp. 13–17.

[13] B. K. P. Horn and B. G. Schunk, "Determining optic flow," *Artificial Intell.*, vol. 17, pp. 185–203, 1981.

[14] K. Kanatani, "Camera rotation invariance of image characteristics," *Comput. Vision, Graphics, Image Processing*, vol. 39, no. 3, pp. 328–354, Sept. 1987.

[15] H. G. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," in *Proc. Roy. Soc. London, B*, vol. 208, pp. 385–397, 1980.

[16] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intell.*, vol. 31, no. 3, pp. 355–395, 1987.

[17] P. Maragos, "Tutorial on advances in morphological image processing and analysis," *Opt. Eng.*, vol. 26, no. 7, pp. 623–632, July 1987.

[18] R. C. Nelson, "Qualitative detection of motion by a moving observer," in *Proc. IEEE Comput. Society Conf. Computer Vision and Pattern Recognit.*, Maui, HI, June 1991, pp. 173–178.

[19] P. D. Picton, "Tracking and segmentation of moving objects in a scene," in *Proc. 3rd Int. Conf. Image Processing and its Applicat.*, Coventry, England, 1989, pp. 389–393.

[20] K. Prazdny, "Determining the instantaneous direction of motion from optic flow generated by a curvilinearly moving observer," *Comput. Vision Graphics, Image Processing*, vol. 17, pp. 238–248, 1981.

[21] R. W. Roach and J. K. Aggarwal, "Computer tracking of objects moving in space," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, no. 2, pp. 127–135, 1979.

[22] R. J. Schalkoff and E. S. McVey, "A model and tracking algorithm for a class of video targets," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-4, no. 1, pp. 2–10, 1982.

[23] B. G. Schunk, "The image flow constraint equation," *Comput. Vision, Graphics, Image Processing*, vol. 35, pp. 20–40, 1986.

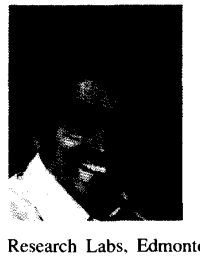
[24] R. S. Stephens, "Real-time 3-D object tracking," *Image Vision Computing*, vol. 8, no. 1, pp. 91–96, 1990.

[25] S. Ullman, "Analysis of visual motion by biological and computer system," *IEEE Comput.*, vol. 14, no. 8, pp. 57–69, 1981.

[26] B. Wilcox, D. B. Gennery, B. Bon, and T. Litwin, "Real-time model-based vision system for object acquisition and tracking," *SPIE*, vol. 754, 1987.

Don Murray was born in Edmonton, Alberta, Canada, on March 3, 1963. He received the B.Sc. degree in 1986 and the M.Sc. degree in 1992, both in electrical engineering, from the University of Alberta, AB, Canada.

He is currently in the DEA Program for Robotics and Computer Vision at the University of Nice, Nice, France. His research interests include computer vision, statistical image processing, and active vision.



Anup Basu (S'89–M'90) received the B.S. degree in mathematics and statistics and the M.S. degree in computer science, both from the Indian Statistical Institute, and the Ph.D. degree in computer science from the University of Maryland, College Park.

He has been with Tata Consultancy Services, India, and the Biostatistics Division, Strong Memorial Hospital, Rochester, NY. He is currently an Assistant Professor at the Computing Science Department, University of Alberta, Alberta, Canada, and an Adjunct Professor at Telecommunications Research Labs, Edmonton, Alberta, Canada. His research interests include computer vision, robotics, and teleconferencing.