



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Computer Vision and Image Understanding 96 (2004) 100–128

Computer Vision
and Image
Understanding

www.elsevier.com/locate/cviu

Temporal spatio-velocity transform and its application to tracking and interaction

Koichi Sato*, J.K. Aggarwal

*Computer and Vision Research Center, Department of Electrical and Computer Engineering,
The University of Texas at Austin, Austin, TX 78712, USA*

Received 7 August 2002; accepted 2 February 2004

Available online 3 September 2004

Abstract

This paper describes the temporal spatio-velocity (TSV) transform for extracting pixel velocities from binary image sequences. The TSV transform is derived from the Hough transform over windowed spatio-temporal images. We present the methodology of the transform and its implementation in an iterative computational form. The intensity at each pixel in the TSV image represents a measure of the likelihood of occurrence of a pixel with instantaneous velocity in the current position. Binarization of the TSV image extracts blobs based on the similarity of velocity and position. The TSV transform provides an efficient way to remove noise by focusing on stable velocities, and constructs noise-free blobs. We apply the transform to tracking human figures in a sidewalk environment and extend its use to an interaction recognition system. The system performs background subtraction to separate the foreground image from the background, extracts standing human objects and generates a one-dimensional binary image sequence. The TSV transform takes the one-dimensional image sequence and yields the TSV images. Thresholding of the TSV image generates the human blobs. We obtain the human trajectories by associating the segmented blobs over time using blob features. We analyze the motion-state transitions of human interactions, which we consider to be combinations of ten simple interaction units (SIUs). Our system recognizes the 10 SIUs by analyzing the shape of the human trajectory. We illustrate the TSV transform and its application to real images for human segmentation, tracking and interaction classification.

© 2004 Elsevier Inc. All rights reserved.

* Corresponding author. Fax: 1-512-471-5532.

E-mail addresses: koichisato@alumni.utexas.net (K. Sato), aggarwaljk@mail.utexas.edu (J.K. Aggarwal).

Keywords: Temporal spatio-velocity transform; Hough transform; Spatio-temporal; Windowing; Human segmentation; Tracking; Interaction recognition

1. Introduction

Human motion analysis is an important area of research in computer vision. Its potential applications include surveillance and tracking. Aggarwal and Cai [1] reported on methods for analyzing human motion. In general, human motion analysis consists of the following three steps:

- Image segmentation—partitioning the image into various objects,
- motion estimation—estimating velocities of objects over time,
- object tracking—tracking the moving objects in a sequence of images.

The present research introduces an object-tracking method that employs pixel velocity extraction from each object being tracked. This method we call the temporal spatio-velocity transform (TSV transform). We show that its application to human tracking has advantages over much of the existing work on the subject. In the remainder of the introduction we give a brief introduction to the TSV, discuss previous work on tracking and interaction and explicate the structure of the paper.

The TSV transform is a combination of a windowing operation and a Hough transform. It is applied to a binary spatio-temporal image to extract pixel velocities. The transformed image is thresholded to obtain blobs with similar spatial position and instantaneous velocity. For a given sequence of images, we process each image to obtain a sequence of binary images, which efficiently represent humans. In tracking persons, we first identify human blobs, where identification roughly means association over time. Blob features employed in the identification are area, horizontal size, horizontal projection, vertical texture, and relative acceleration among blobs. Identification generates trajectories that help recognize interactions. In particular, we consider trajectory shapes that fit the patterns in simple interaction units (SIUs). SIUs are simplified versions of complex interactions expressed as changes between two states, moving and being still. This approach to tracking was introduced in [2–4].

Previous approaches to object tracking and interaction classification have employed a number of segmentation, motion estimation, and tracking algorithms. Oliver et al.'s [5] study is an important paper in this research area. Our research is partially based on their results. They focused on (i) segmenting humans from several candidate background images using principle component analysis, (ii) estimating their position and velocity by applying a first order Kalman filter, (iii) identifying human blobs using the Gaussian Probability Density Function, and (iv) recognizing two-person interactions by applying the Coupled Hidden Markov Model to extracted features such as distance, velocities, and moving direction. Niyogi and Adelson [6] tracked human blobs moving at a constant speed by detecting straight lines in spatio-temporal images using the Hough Transform, assuming that the

object is moving at a constant speed and therefore its trajectory appears as a straight line on the spatio-temporal sliced image. Haritaoglu et al. [7] and Haritaoglu and Davis [8] successfully tracked human involving multiple persons by combining several techniques. Cai and Aggarwal [9] employed 3D geometry in multiple perspectives, motion detection, image segmentation, and Bayesian pattern recognition to track persons using multiple cameras.

Among the previous studies on activity tracking, Ayers and Shah [10] detected actions of people in a room. Davis and Bobick [11] and Bobick [12] used temporal templates to recognize human activities. Yamato et al. [13] achieved human activity recognition through application of the Hidden Markov Model. Syeda-Mahmood [14] used the idea of an action cylinder. Syeda-Mahmood et al. [15] segmented actions in general motion sequences of 3D objects using velocity curve space, and then computed the hierarchical description of action boundaries.

In addition to object tracking, pixel velocity extraction is an important component of our algorithm. It is thus helpful to discuss some of the significant contributions on this issue. Horn and Schunk [16] proposed an optical flow approach to extract pixel velocities using grayscale or color image sequences under the smooth surface constraints. Kim et al. [17] advanced Horn and Schunk's approach using a fast convergent method. Chong et al. [18] proposed a time-delay-based image-velocity computation that extracted the pixel velocity from grayscale or color image sequences using several sets of delay lines.

The remaining sections of the paper are organized as follows: Section 2 describes the TSV transform and its application to one-dimensional images, Section 3 considers tracking using the TSV transform, Section 4 addresses interaction recognition using trajectories, Section 5 discusses the results of our experiment with real sequences, and Section 6 presents our conclusions.

2. TSV transform

In our tracking system, the TSV transform plays an important role by extracting pixel velocities to form blobs corresponding to objects in the images. The proposed transform groups pixels with similar velocities and then constructs a velocity-stable blob by thresholding. In this section, we outline the definition and an implementation of the TSV transform to one-dimensional binary images.

2.1. Principles of TSV

The TSV transforms a one-dimensional sequence X after expanding X into a spatio-temporal image XT as shown in Fig. 1. The transform performs a windowing operation on the spatio-temporal image and applies the Hough transform to the windowed spatio-temporal image to obtain the TSV image.

Let the one-dimensional sequence be given by $K_n(x)$ where n denotes the frame number in the image sequence. The spatio-temporal image is a concatenation of the images for various values of n : $K_n(x)$. The trajectory of a point P in the

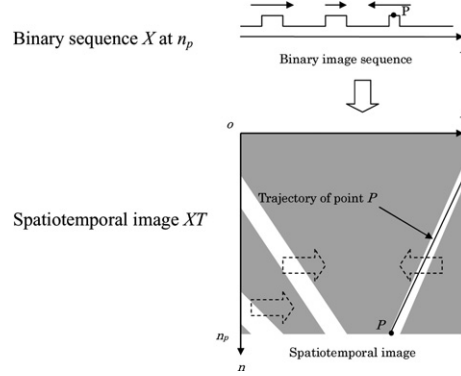


Fig. 1. Binary image sequence and spatio-temporal image.

spatio-temporal plane is a curve that is uniquely determined by the velocity and the position. The instantaneous slope of the curve is the velocity at the point P . If the velocity is constant, the curve is a straight line. Two parameters, velocity (v) and position (p), are used in the Hough transform to extract the pixel trajectory.

Let

$$x = v(n - n_p) + p, \quad (1a)$$

where n_p is the current frame number and v, p are the pixel velocity and position of the point P in the n_p th frame, respectively, and x is the pixel position in the n th frame. Eq. (1a) is the relationship between variables x and n using parameters v and p . Thus, a parametric pair v and p represents a line. For each line in the x – n spatio-temporal plane, the set of possible parametric pairs (p, v) can be expressed as a point in the (p, v) parametric plane: (see Fig. 2)

$$p = -(n - n_p)v + x. \quad (1b)$$

A function of (p, v) , $V_{n_p}(p, v)$, is obtained by counting the number of pixels for which $K_n(x) = 1$ along the corresponding line given by Eq. (1a),

$$V_{n_p}(p, v) = \sum_n \sum_x K_n(x) \delta(x - v \cdot (n - n_p) - p), \quad \text{where } \delta(a) = \begin{cases} 0 & (a \neq 0), \\ 1 & (a = 0). \end{cases} \quad (2)$$

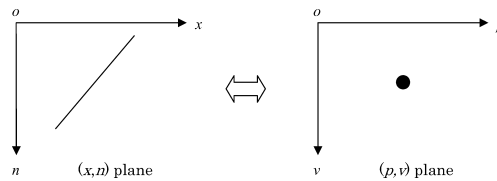


Fig. 2. The relationship between the image plane and parametric plane.

So,

$$V_{n_p}(p, v) = \sum_n K_n(v \cdot (n - n_p) + p). \quad (3)$$

Since each point on (p, v) represents a line in spatio-temporal space, the spatio-temporal lines may be extracted by selecting the points at which the vector value (p, v) makes a local maximum for the function $V_{n_p}(p, v)$. Up to this point, for simplicity, we have assumed that the velocity is constant and the trajectory is a straight line. To extract a slightly curved trajectory, as in the case where there is acceleration, we weight the spatio-temporal image before taking the Hough transform so that the “present” image is weighted more. In other words, we take a windowing operation before performing the Hough transform. In doing so, we employ an exponential window. The spatio-temporal image $K_n(x)$ is windowed to $L_{n_p}(x, n)$ at n_p to give:

$$L_{n_p}(x, n) = K_n(x) \cdot F_{n_p}(n), \quad (4)$$

where

$$F_{n_p}(n) = \begin{cases} (1 - e^{-\lambda})e^{\lambda(n-n_p)} & n \leq n_p \text{ (past or current)} \\ 0 & n > n_p \text{ (not defined : future)}, \end{cases} \quad (5)$$

and λ is the time constant. This constant determines the weighting ratio over time that controls the influence of acceleration on the resulting TSV image. We will discuss this relationship between the time constant value and acceleration in Section 2.3 and [Appendix A](#).

Now, we generate a TSV image by applying the Hough transform to the windowed spatio-temporal image $L_{n_p}(x, n)$,

$$V_{n_p}(p, v) = \text{Hough}_{x=v(n-n_p)+p} \{L_{n_p}(x, n)\} = \sum_n L_{n_p}(v(n - n_p) + p, n). \quad (6)$$

The resulting TSV image generates a measure of the likelihood of occurrence of a pixel with instantaneous velocity v and current position p .

2.2. Implementation of TSV

The computation of the Hough transform using Eq. (6) is a time consuming process. Therefore, we use a simpler version of the transform.

By rewriting (6), we have

$$V_{n_p}(p, v) = \sum_{n=-\infty}^{n_p} \{K_n(v(n - n_p) + p)\} (1 - e^{-\lambda})e^{\lambda(n-n_p)}. \quad (7)$$

Here, if we calculate the $V_{n_p-1}(p, v)$, we get

$$\begin{aligned} V_{n_p-1}(p, v) &= \sum_{n=-\infty}^{n_p-1} \{K_n(v(n - n_p + 1) + p)\} (1 - e^{-\lambda})e^{\lambda(n-n_p+1)} \\ &= V_{n_p}(p + v, v)e^{\lambda} - K_{n_p}(p + v)(1 - e^{-\lambda})e^{\lambda}, \end{aligned} \quad (8)$$

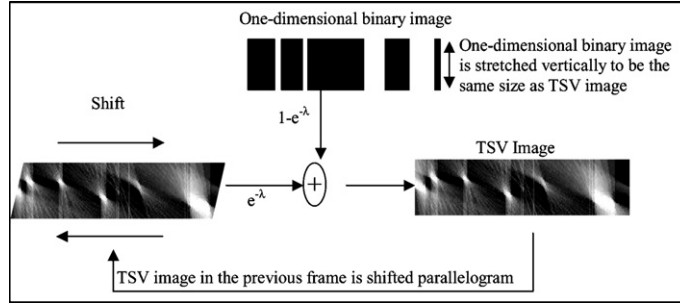


Fig. 3. System flow of the computation of the TSV transform.

or,

$$V_{n_p}(p+v, v) = e^{-\lambda} V_{n_p-1}(p, v) + (1 - e^{-\lambda}) K_{n_p}(p+v). \quad (9)$$

By shifting the horizontal position coordinate in Eq. (9) by $-v$, we obtain

$$V_{n_p}(p, v) = e^{-\lambda} V_{n_p-1}(p-v, v) + (1 - e^{-\lambda}) K_{n_p}(p). \quad (10)$$

This equation shows that the TSV transform can be obtained by iterative operations: (i) horizontal shifting, (ii) multiplication by a constant coefficient, and (iii) addition, as shown in Fig. 3.

Fig. 3 shows the TSV image where the velocity axis corresponds to the desired number of velocity bins. The horizontal axis is the spatial axis compatible with the binary image sequence. The one-dimensional binary image sequence is vertically duplicated so that it is the same size as the TSV image. In parallel, the TSV image in the previous frame is shifted in the p -axis by $-v$, whereas the v axis is unchanged. Then, we multiply the duplicated binary image and the shifted TSV images by specific coefficients and added the two images.

By binarizing the resulting TSV image with a threshold Th_V , we obtain a binary TSV image.

$$\tilde{V}_{n_p}(p, v) = \begin{cases} 1 & \text{if } V_{n_p}(p, v) \geq Th_V, \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

We perform eight-neighborhood connected component processing to obtain the blobs over the binary TSV images.

Fig. 4 shows the result of the TSV transform. Image 4B shows the TSV transform of the one-dimensional sequence and Fig. 4C gives the binarized transform image.

2.3. The acceleration range

The acceleration range applicable to the TSV transform is controlled by the time constant λ in Eq. (10) in Section 2.2. If an object is accelerating, then the value of $V_{n_p}(p, v)$ is small. The larger the acceleration, the smaller the value of $V_{n_p}(p, v)$

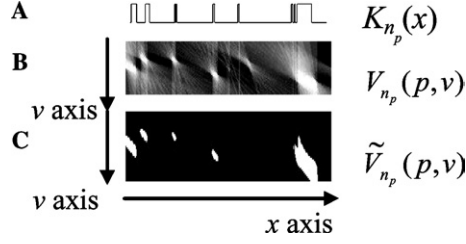


Fig. 4. Example of the TSV transform.

becomes. Moreover, once acceleration exceeds a certain value, the object fails to appear as a blob in the TSV image. In [Appendix A](#), we derive the range of acceleration for the TSV transform to highlight an object. For a given value of λ , the range of acceleration a is given by

$$-8w\lambda^2 \ln \left(\frac{1}{1 - Th_v} \right)^{-2} < a < 8w\lambda^2 \ln \left(\frac{1}{1 - Th_v} \right)^{-2}, \quad (12)$$

where w is the width of the blob. Expression (12) shows the range of the acceleration of a blob based on the width w , time constant λ , and threshold Th_v .

2.4. The influence of distance on extraction results

In the application discussed in Section 3, humans are located at different distances from the camera, so their apparent acceleration may vary. The effect of the acceleration of a person in world coordinates can be obtained using formula (B.2) in [Appendix B](#). When the distance from the camera Y is much larger than the difference between the object height Z and the camera height Z_c , the x coordinate is approximated as,

$$x = F \frac{X}{Y \cos \alpha}. \quad (13)$$

Since the person's width w (width W in world coordinate), velocity v (velocity V in world coordinate), and acceleration a (acceleration A in world coordinate) are all x coordinate variables, each variable may be approximated similarly,

$$w = \frac{FW}{Y \cos \alpha}, \quad (14)$$

$$v = \frac{FV}{Y \cos \alpha}, \quad (15)$$

$$a = \frac{FA}{Y \cos \alpha}. \quad (16)$$

By substituting these values into expression (12), the following inequalities for A and W are obtained,

$$-8 \frac{FW}{Y \cos \alpha} \lambda^2 \ln \left(\frac{1}{1 - Th_V} \right)^{-2} < \frac{FA}{Y \cos \alpha} < 8 \frac{FW}{Y \cos \alpha} \lambda^2 \ln \left(\frac{1}{1 - Th_V} \right)^{-2}, \quad (17)$$

$$-8W\lambda^2 \ln \left(\frac{1}{1 - Th_V} \right)^{-2} < A < 8W\lambda^2 \ln \left(\frac{1}{1 - Th_V} \right)^{-2}. \quad (18)$$

Formula (18) does not contain the object distance Y . We can state an important characteristic of the TSV transform that it is not affected by the distance between the object and the camera.

2.5. Advantages of TSV transform

The TSV transform suppresses noise in the spatio-temporal image, because it acts as a low-pass filter. Fig. 5 shows the frequency response of the low-pass filter. As higher frequencies are attenuated, it suppresses noise.

To illustrate the above phenomenon, consider the following binary image sequences with simulated noise:

In Fig. 6, two objects (10 pixels wide) are moving at different velocities: 1 and -0.5 pixels per frame, respectively. Sequence (A) is a noise-free image sequence. We have added 5 and 10% ‘pixel noise’ to (B) and (C) images respectively. ‘Pixel noise’ corresponds to a pixel inverted from 1 to 0 or 0 to 1.

Fig. 7 shows the resulting TSV images for the sequences in Fig. 6 at the 100th, 135th, and 200th frames. The images in Fig. 8 are the result of binarization with a threshold of 0.7. It is evident from Fig. 8 that the binarized TSV transform is relatively unaffected by the simulated noise.

Apart from noise suppression, the transform successfully separates occluded blobs depending on their positions and velocities. The TSV transform is capable

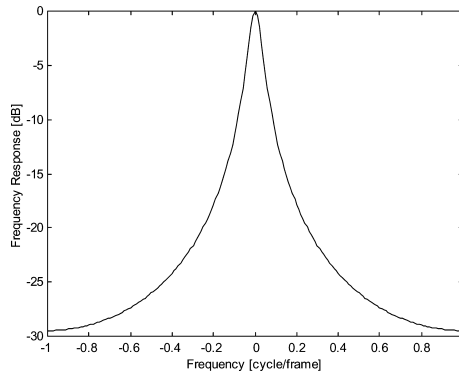


Fig. 5. Frequency response of the TSV transform.

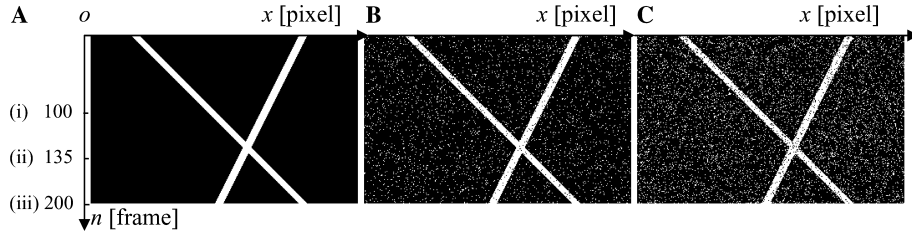


Fig. 6. One-dimensional spatio-temporal image with different noise levels.

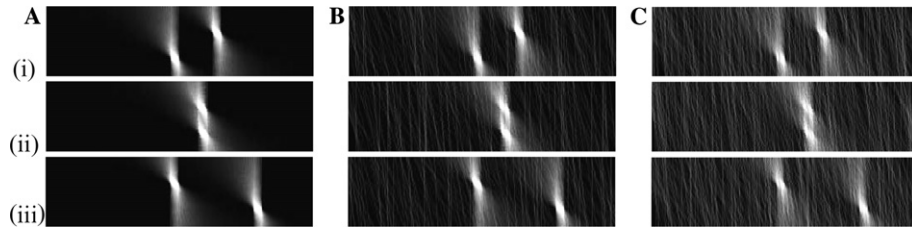


Fig. 7. The resulting TSV images performed over the sequences in Fig. 6.

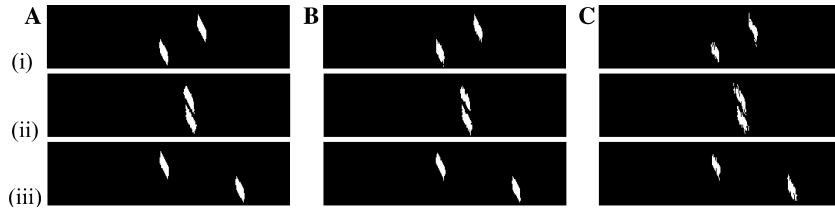


Fig. 8. The binarized TSV images performed over the sequences in Fig. 7.

of obtaining more than one velocity value for a given position by extracting velocity variation at each pixel corresponding to different objects occluding each other. This enables us to separate object blobs out of a composite blob, given the appropriate position and velocity values. Using the same image sequence as Fig. 6, Fig. 9 shows the 135th frame of the TSV image (Figs. 7 and 8) in which two blobs are completely occluded. The corresponding binarized TSV image, however, shows that by using an appropriate threshold, the occluded blobs have been separated into two blobs.

3. Tracking humans in lateral-view image sequences

3.1. Application overview

We focus on tracking persons walking along a sidewalk from lateral-view image sequences. There may be more than one sidewalk in the view of the camera. Each image may contain objects of different sizes, as the persons are viewed at different

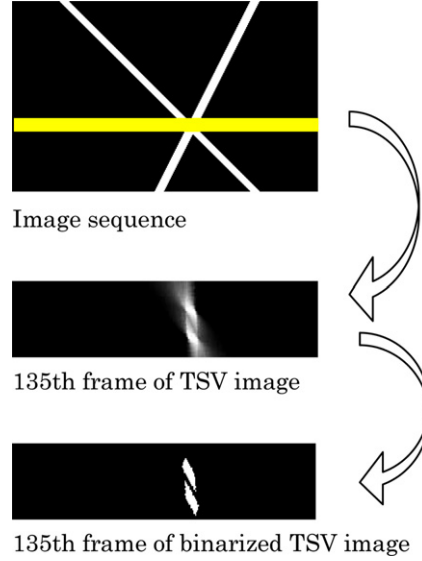


Fig. 9. Result of TSV transform when two blobs are occluded (135th frame of Figs. 7 and 8).

distances from the camera. Occlusion occurs frequently between two objects. Only movements parallel to the image plane are captured; movements perpendicular to the image plane are not captured. Fig. 10 shows an overview of our system, which involves four steps: (i) background subtraction that separates a foreground image from a background image; (ii) object extraction that picks out standing object blobs from the other foreground blobs and converts two-dimensional binary images to a one-dimensional binary image; (iii) the TSV transform analyzes pixel velocity and groups together pixels with similar velocities; and (iv) blob association identifies each blob. In the following we discuss each of these topics in detail.

3.2. Background subtraction

The background is fixed in the image sequences of interest in this paper, while the foreground contains moving human objects. We attempt to set them apart by the following procedure. We define that a pixel belongs to the foreground if the absolute difference between the pixel and the background pixel exceeds a fixed threshold,

$$S(x, y) = \begin{cases} 1 & |I(x, y) - B(x, y)| > Th_B, \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

where x, y are the horizontal and vertical coordinates of the images, $S(x, y)$ is the binarized image, $I(x, y)$ is the original image and $B(x, y)$ is the stored background image. $Th_B(n)$ is the threshold for binarization in n th frame. The value of the threshold is estimated dynamically in order to keep the amount of isolated noise at a 'reasonable' level. Table 1 shows the relationship between the image quality, threshold $Th_B(n)$ and the number of isolated pixel $N_i(n)$.

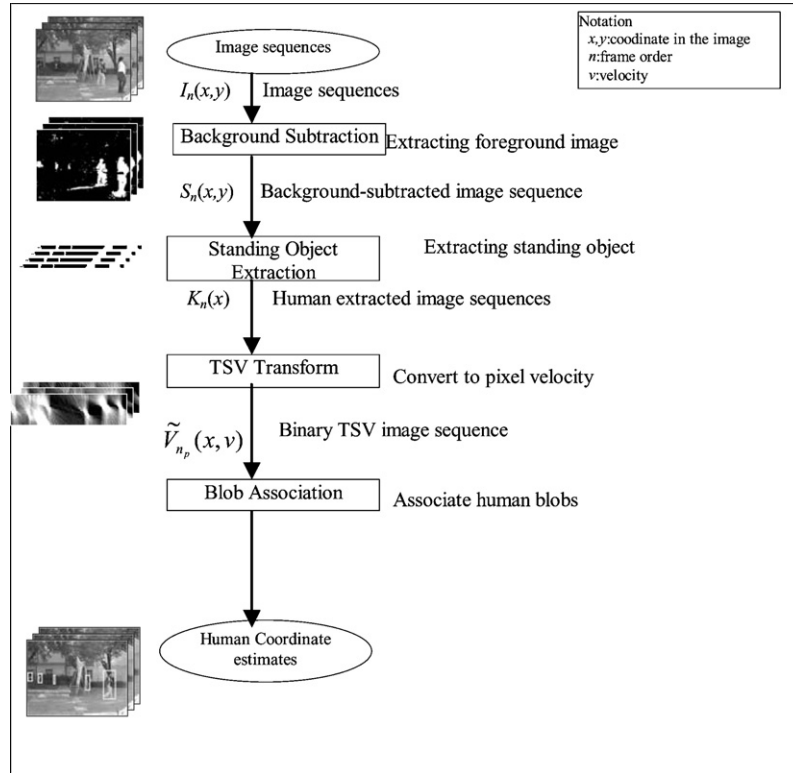

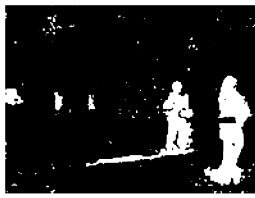


Fig. 10. System overview.

Table 1

The relationship between the image quality, the threshold $Th_B(n)$, the number of isolated pixels $N_i(n)$, and the binary image $S(x, y)$

	High threshold	Low threshold
Image quality	High	Low
Isolated pixels $N_i(n)$	Few	Many
The example binary image $S(x, y)$ (Background subtracted binary image)		

Isolated noise refers to '1'-valued pixels surrounded by '0'-valued pixels in the $S(x, y)$ image (see Fig. 11). We use an iterative algorithm to choose the threshold $Th_B(n)$. $Th_B(n)$ can be obtained using $Th_B(n - 1)$, as follows:

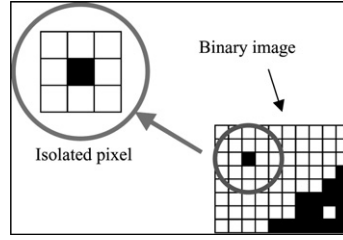


Fig. 11. The isolated pixel.

$$Th_B(n) = Th_B(n-1) + \tau\{N_i(n) - N_t\}, \quad (20)$$

where N_t is the desired number of isolated noise pixels and $\tau(>0)$ is the convergence coefficient. Through iterative computation, the threshold $Th_B(n)$ converges to an asymptotic value. At the same time, $N_i(n)$ converges to N_t . Thus, the asymptotic value of $Th_B(n)$ is a threshold that generates N_t isolated pixels in a binary image $S(x, y)$. By choosing suitable parameters for N_t and τ , this method keeps the threshold $Th_B(n)$ at an appropriate value that neither misses human blobs nor makes the binary image too noisy.

Usually, the threshold $Th_B(n)$ converges to a value that generates noisy, but useful images. In this computation, the target number N_t represents the quality of the image. We assume that, within a certain range of $Th_B(n)$, N_t , and $N_i(n)$, the quality of the image is a monotonically increasing function of the amount of the isolated noise $N_i(n)$, which in turn is a monotonically decreasing function of the threshold $Th_B(n)$. This feedback process enables us to separate the blobs with intensities similar to that of background. We control noise levels to be constant across all binary images.

3.3. Standing object extraction and TSV transform

The standing object extraction process allows us to obtain standing object blobs from a binarized image $S(x, y)$ using Region A, which is a stripe that covers a small area above and below the horizon line. The horizon line is set manually to correspond to the height of the camera lens Z_c . We define a standing object—corresponding to a human object—as a object that intersects with Region A. Since we are interested in standing objects, we remove any noise whose origin is not a standing object by limiting the viewing range within Region A. We take a vertical projection within Region A and re-binarize it at a certain threshold Th_H to generate a one-dimensional binary image sequence:

$$K(x) = \begin{cases} 1 & \sum_{y=y_a}^{y_b} S(x, y) > Th_H, \\ 0 & \text{otherwise,} \end{cases} \quad (21)$$

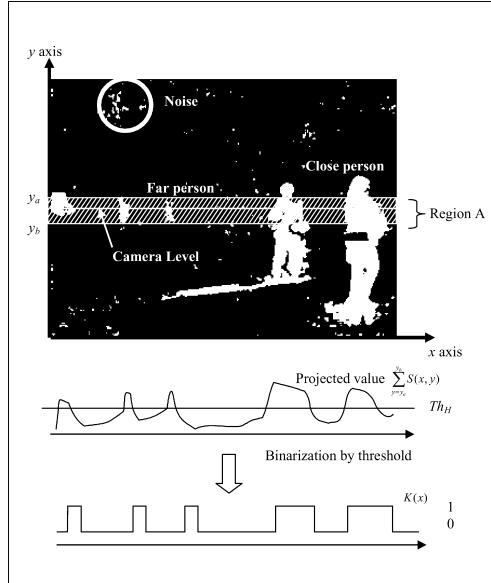


Fig. 12. An example of the object extraction process.

where Th_H is a threshold that is constant in all situations and y_a and y_b are the parameters of Region A. The parameters y_a and y_b for Region A are derived in [Appendix B](#).

Generally speaking, an object extraction method that is designed to track relatively large blobs tends to recognize smaller blobs as noise and ignore them. If used to capture small blobs, then such a system will suffer from noise interference problems. Our system successfully avoids this difficulty because it is independent of blob size: we extract standing objects solely based on whether they intersect the horizon line.

For example, the tree noise at the top left of [Fig. 12](#) is eliminated because it does not intersect with Region A, and so is the shadow of a person.

Now, TSV transform is applied to the one-dimensional binary image sequence $K_n(x)$. The resulting binary TSV image in the current n_p frame $\tilde{V}_{n_p}(p, v)$ is computed:

$$\tilde{V}_{n_p}(p, v) = \text{Binarize}[\text{TSV}\{K_{n_p}(x)\}]_{Th_V}. \quad (22)$$

as described earlier.

3.4. Human tracking

The human tracking process identifies various blobs. Until this stage, they are just compact groups of pixels. Tracking consists of iterative implementation of three steps: (i) extracting the features of the obtained blobs in the current frame, (ii) choos-

ing the blob that seems most likely to have the same features as one of the previously tracked blobs, based on Mahalanobis distance, and (iii) judging the most-likely candidate blob to see if it is similar enough to the previously tracked blob, based on a Bayesian posterior probability measure.

We take into account the following four features: the horizontal size of the blob (ζ_h), the blob area size (ζ_a), the vertical texture (χ_t), and the horizontal projection of the binary blob (χ_p) (see Fig. 13). These features are defined as follows:

$$\begin{aligned}\zeta_a &= \sum_{(x,y) \in \text{Blob}} S(x,y), \\ \zeta_h &= 2 \left[\frac{1}{\zeta_a} \sum_{(x,y) \in \text{Blob}} x^2 S(x,y) - \left(\frac{1}{\zeta_a} \sum_{(x,y) \in \text{Blob}} x S(x,y) \right)^2 \right]^{\frac{1}{2}}, \\ \chi_p(y) &= \sum_{x|(x,y) \in \text{Blob}} S(x,y), \\ \chi_t(y) &= \frac{\sum_{x|(x,y) \in \text{Blob}} I(x,y)}{\chi_p(y)}.\end{aligned}\tag{23}$$

The difference values ζ'_a , ζ'_h , χ'_t , and χ'_p from the features of the tracked objects are then calculated, as well as the blob acceleration in the image coordinate from the previously tracked blob (η). For every possible combination of the candidate blobs and the tracked blobs, we calculate these features. These variables are assumed to be independent of one another and are a measure of similarity to the tracked object, computed as follows:

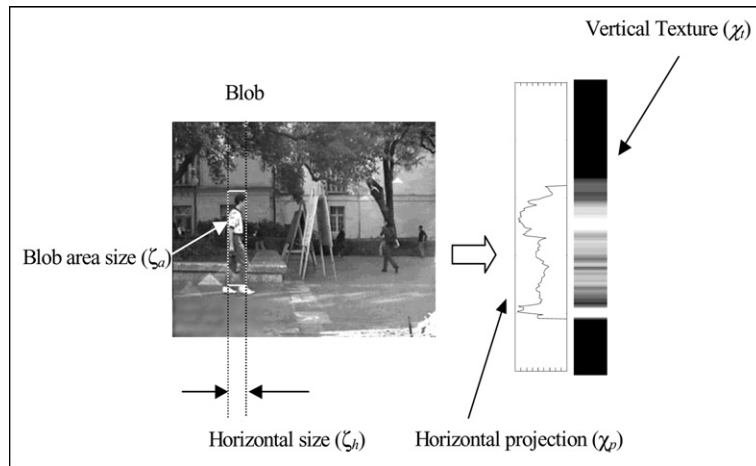


Fig. 13. Features used for identification.

Let

$$\begin{aligned}
 \zeta'_a &= \zeta_a - \zeta_{a \text{ tracking}}, \\
 \zeta'_h &= \zeta_h - \zeta_{h \text{ tracking}}, \\
 \chi'_t &= \sum_y |\chi_t \cdot \chi_p - \chi_{t \text{ tracking}} \cdot \chi_{p \text{ tracking}}|, \\
 \chi'_p &= \sum_y |\chi_p - \chi_{p \text{ tracking}}|, \\
 \eta &= 2 \frac{x - x_{\text{tracking}}}{(n - n_1)^2} - 2 \frac{v_{\text{tracking}}}{n - n_1},
 \end{aligned} \tag{24}$$

the notations of the above equation are described in Table 2.

Similar to Oliver's et al.'s work [2], we determine the most-likely candidate to be the blob that has the minimal Mahalanobis distance based on five features $(\zeta'_a, \zeta'_h, \chi'_t, \chi'_p, \eta)$.

$$D_M = \left\{ \frac{\zeta'^2_a}{\sigma_a^2} + \frac{\zeta'^2_h}{\sigma_h^2} + \frac{\chi'^2_t}{\sigma_t^2} + \frac{\chi'^2_p}{\sigma_p^2} + \frac{\eta^2}{\sigma_\eta^2} \right\}, \tag{25}$$

where σ_a^2 , σ_h^2 , σ_t^2 , σ_p^2 , and σ_η^2 are the variance for ζ'_a , ζ'_h , χ'_t , χ'_p , and η , respectively. These variances are obtained from the training set.

After selecting the most-likely candidate blob, we need to determine if it is similar enough to the tracked blob. In the training phase, we used pairs from training images and manually categorized whether they belong to the same tracked object (ω) or not ($\bar{\omega}$). We calculated the a priori conditional probability $p(\mathbf{u}|\omega)$, $p(\mathbf{u}|\bar{\omega})$ using parameter vector $\mathbf{u} = \{\zeta'_a, \zeta'_h, \chi'_t, \chi'_p, \eta\}$ using test sequence. In this paper, $p(\mathbf{u}|\omega)$, $p(\mathbf{u}|\bar{\omega})$ are assumed to have a normal distribution, and are obtained by computing the variance and mean for each category ω and $\bar{\omega}$. In the test phase, we calculated the posterior probability $P(\omega|\mathbf{u})$ and $P(\bar{\omega}|\mathbf{u})$ using Bayes' rule:

$$\begin{aligned}
 P(\omega|\mathbf{u}) &= \frac{p(\mathbf{u}|\omega)P(\omega)}{p(\mathbf{u})}, \\
 P(\bar{\omega}|\mathbf{u}) &= \frac{p(\mathbf{u}|\bar{\omega})P(\bar{\omega})}{p(\mathbf{u})}.
 \end{aligned} \tag{26}$$

The most-likely candidate blob joins the tracked blob if $P(\omega|\mathbf{u}) > P(\bar{\omega}|\mathbf{u})$.

Table 2
Notation for Eq. (24)

$\zeta_{a \text{ tracking}}$	The area size of the tracked human blob
$\zeta_{h \text{ tracking}}$	The horizontal size of the tracked human blob
$\chi_{t \text{ tracking}}$	The vertical texture of the tracked blob
$\chi_{p \text{ tracking}}$	The horizontal projection of the tracked blob
n_1	The frame number in which the last blob joins the tracked blob
v_{tracking}	The velocity of the tracked blob at the n_1 th frame
x_{tracking}	The horizontal position of the tracked blob at the n_1 th frame

4. Interaction modeling and classification

In this section, we develop a model for interactions and classify them using their trajectory shapes. The trajectory of the center coordinates of the blob defines the human trajectory. We first extract the interesting trajectory features. Once they are obtained, we detect the interaction moment and adopt the nearest mean method to classify the interactions into what we call SIUs, which we discuss in the following. The other components of the classification process are dealt with in the subsequent subsections.

4.1. Simple interaction unit

Human interactive activity is often complex. To recognize complex two-person interactions, we define SIUs. For the present paper, we consider every human activity to be a combination of two motion states: moving and being still, as well as the state of being in transition between these two states. Our model decomposes complex human interactions into simple components based on change of the two states. Denoting “moving” state as S_M and “being still” as S_S , a state transition is represented as either $S_S \rightarrow S_M$ or $S_M \rightarrow S_S$. When at least one of the two persons changes his or her state, the interaction is a SIU. Table 3 shows the possible state transitions and the resulting simple interaction units and Table 4 gives a short description of each of the SIUs. The icons in Table 4 appear in our results in Section 5.4.

Table 3
SIU and motion state transitions of each person

Person 2	Person 1			
	$S_S \rightarrow S_M$	$S_M \rightarrow S_S$	S_S	S_M
$S_S \rightarrow S_M$	APART			
$S_M \rightarrow S_S$	STOPGO	MEET		
S_S	LEAVE	STOP	STANDSTILL	
S_M	FOLLOW	WALKSTOP	PASS1	PASS2 TAKEOVER

Table 4
SIU types

SIU	Description	Icon
APART	Two people start walking in different directions	
FOLLOW	One stationary person starts following another walking person	
LEAVE	One person leaves another stationary person	
MEET	Two people meet from different directions	
PASS1	One person passes by another stationary person	
PASS2	Two people pass each other from the opposite direction	
STOP	One person stops in front of another stationary person	
TAKEOVER	One person passes by another person walking in the same direction	
WALKSTOP	Two people walk together, then one of them stops	
STOPGO	One person starts walking when the other stops	

4.2. Feature extraction

To classify the trajectory shapes we extract four features: (i) the relative distance $d(n)$ between two persons, (ii) the slope $s_r(n)$ of the relative distance $d(n)$, and (iii) and (iv) the slopes $s_1(n)$ and $s_2(n)$ of position $p_1(n)$, $p_2(n)$ (in the n th frame) of each human blob. $p_1(n)$, $p_2(n)$ are defined as the horizontal center coordinates of the blobs. These slopes are computed using a finite impulse response filter $f_r(n)$.

Let,

$$d(n) = |p_1(n) - p_2(n)|, \quad (27)$$

then,

$$s_r(n) = f_r(n) * d(n), s_1(n) = f_r(n) * p_1(n), s_2(n) = f_r(n) * p_2(n), \quad (28)$$

where

$$f_r(n) = \begin{cases} \frac{3n}{T(T+1)(2T+1)} & -T \leq n \leq T, \\ 0 & \text{otherwise} \end{cases}. \quad (29)$$

The slope filter $f_r(n)$ calculates the regression slope of the $(2T + 1)$ consecutive points centered at n . In the case of $s_r(n)$, we obtain the slope using a regression in [19] by taking the covariance between $d(n)$ and n over the variance of n , which is expressed as the convolution between $f_r(n)$ and $d(n)$.

Since the distance between each human and the camera may differ, the features are normalized by the distance. The horizontal size of the human is used to approximate the distance of the human from the camera. To reduce the effect of large variations in the trajectory features, we apply a hyperbolic tangent normalization. Let $\tilde{d}(n)$, $\tilde{s}_r(n)$, $\tilde{s}_1(n)$, and $\tilde{s}_2(n)$ be the normalized features of $d(n)$, $s_r(n)$, $s_1(n)$, and $s_2(n)$, respectively. We compute:

$$\begin{aligned} \tilde{d}(n) &= \tanh\left(\frac{d(n)}{D}\right) & \tilde{s}_r(n) &= \tanh\left(\frac{s_r(n)}{D}\right), \\ \tilde{s}_1(n) &= \tanh\left(\frac{s_1(n)}{D}\right) & \tilde{s}_2(n) &= \tanh\left(\frac{s_2(n)}{D}\right), \end{aligned} \quad (30)$$

where D is the alignment coefficient. D is twice the horizontal size as that of the blob.

4.3. Interaction moment detection and SIU recognition

Our system detects two types of interaction moments, namely the tangible and potential interaction moments, using five features $\tilde{s}_1(n)$, $\tilde{s}_2(n)$, $\tilde{d}(n)$, $p_1(n)$, and $p_2(n)$. We define the tangible interaction moment as the moment at which at least one of the values $(\tilde{s}_1(n) - Th_s)$ and $(\tilde{s}_2(n) - Th_s)$ change sign, provided that $\tilde{d}(n) < Th_d$. The potential interaction moment is defined as the moment at which the value $\{p_1(n) - p_2(n)\}$ changes sign. Upon detection of either a tangible or a potential interaction moment, the system performs the SIU recognition (Fig. 14).

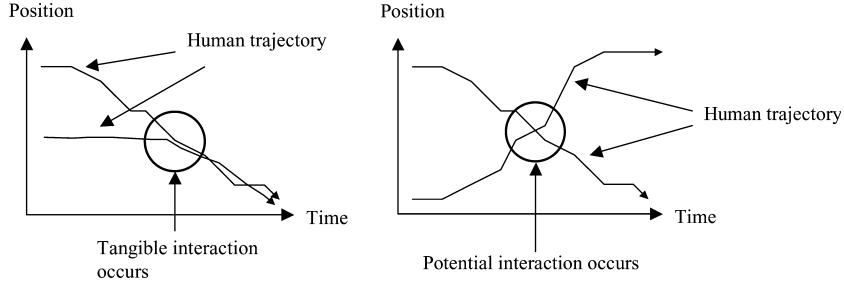


Fig. 14. A tangible interaction and a potential interaction.

When an interaction moment is detected, the system recognizes its SIU using the feature matrix \mathbf{M} . Let $\mathbf{C}(n)$ be the set of aligned features $\tilde{d}(n)$, $\tilde{s}_r(n)$, $\tilde{s}_1(n)$, and $\tilde{s}_2(n)$. Let n_p be the current frame. Matrix \mathbf{M} is generated by concatenating $(2l + 1)$ sets of $\mathbf{C}(n)$, which is taken every R th frame and centered at n_p . We obtain:

$$\begin{aligned} \mathbf{C}(n_p) &= [\tilde{d}(n_p) \quad \tilde{v}_r(n_p) \quad \tilde{v}_1(n_p) \quad \tilde{v}_2(n_p)]^T, \\ \mathbf{M} &= [\mathbf{C}(n_p - lR), \mathbf{C}(n_p + (-l + 1)R), \dots, \mathbf{C}(n_p + (l - 1)R), \mathbf{C}(n_p + lR)]. \end{aligned} \quad (31)$$

We use the nearest mean method to classify interactions. Let N_M be the number of interesting interaction types. The features of interaction models for the types $1, 2, \dots, N_M$ can be expressed as $\mathbf{M}_1, \mathbf{M}_2, \dots, \mathbf{M}_{N_M}$. Let the features of a type-unknown interaction be \mathbf{M}_u . \mathbf{M}_u is assigned the category label ω ($\omega \in \{1, 2, \dots, N_M\}$) that minimizes the difference between category model \mathbf{M}_ω and \mathbf{M}_u :

$$\min_{\omega} \{\|\mathbf{M}_\omega - \mathbf{M}_u\|\}. \quad (32)$$

Each category model \mathbf{M}_π for SIU π ($\pi \in \{1 \dots N_M\}$) is generated from training sequences. We manually categorized the training sequences into N_M SIU types. We generate the model \mathbf{M}_π for SIU model π by taking the average value of the feature matrix \mathbf{M}_{π_n} of the training sequence:

$$\mathbf{M}_\pi = \frac{1}{N_\pi} \sum_{n=1}^{N_\pi} \mathbf{M}_{\pi_n}, \quad (33)$$

where N_π is the number of training sequences that are manually categorized to π .

5. Results

In this section, we report results on human segmentation, human tracking, and human interaction recognition using a desktop personal computer. Section 5.1 discusses the system performance for segmentation of humans in a sequence that contains 100 frames. Sections 5.2 and 5.3 present the results of human motion tracking and interaction classification. We tracked a total of 124 sequences. Each sequence consists of 3000–10,000 images. Some sample sequences are presented in Section 5.4.

We captured the image sequences using a Sony VX-2000 digital video camera and converted them into Windows AVI files, in 15 frame/s sequences of 320×240 pixel grayscale images. A Pentium IV 1.2 GHz Windows XP machine executed our program, written in C++. The program performed the tracking and interaction recognition procedures using a video sequence (Windows AVI file) in real time, which is more than 15 frames per second. We have also developed an embedded implementation [4] consisting of four 1M-bit multi-port memory and a 16-bit microprocessor. The embedded system is used to perform the tasks of background subtraction, object extraction and the TSV transform at a rate of 100ms/frame (10frames/s).

5.1. Human segmentation

Fig. 15 shows a 1/10 sub-sampled image sequence from a total of 100 frames with the rate of 15 [frames/s]. Each image contains three sub-images: the one-dimensional binary image, the TSV image and the original image, arranged in the descending order. The vertical axis of the TSV image is the velocity axis; and the three horizontal axes correspond to the same coordinate. In the captured scene, two paths run perpendicular to the optical axis of the camera. People walking on these paths were recognized as human blobs in the sequence. The human blobs representing persons moving on the closer path are larger than those on the farther path. After segmentation, each segmented blob is marked by a rectangle superimposed on the original image. We use the blob width of the binary TSV image for the rectangle width; the height of the rectangle is four times the width in each case.

Fig. 15 shows that no blob is missed in the sequence. Also, this system segments (i) the relatively small blobs, (ii) the blobs with intensities similar to that of the background, and (iii) the blobs that are completely occluded.

5.2. Human motion tracking

We used a sequence of 1000 images to obtain the parameters for tracking, namely σ_a^2 , σ_h^2 , σ_t^2 , σ_p^2 , and σ_η^2 shown in Eq. (25) and $p(\mathbf{u}|\omega)$ and $p(\mathbf{u}|\bar{\omega})$ shown in Eq. (26) in

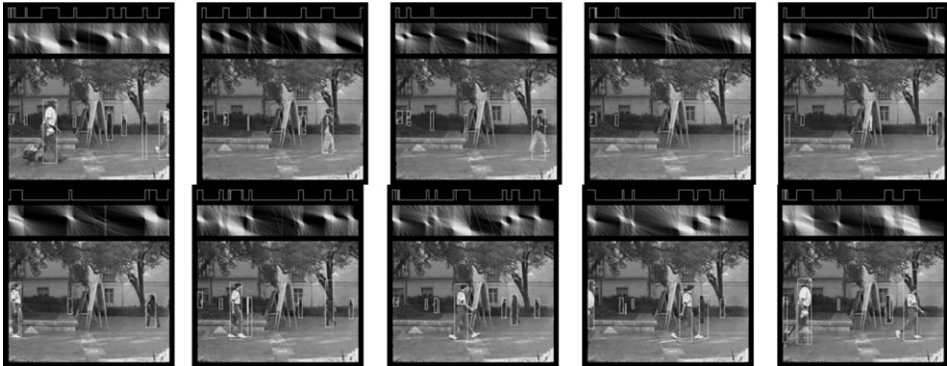


Fig. 15. Results of human segmentation.

Table 5
Results of human segmentation

Scene	Number of humans appearing in the image sequences	Number of correctly segmented humans	Number of lost humans
1	12	12	0
2	14	13	1
3	8	8	0
4	64	60	4
5	84	79	5
6	12	12	0
Total	194	184	10

Table 6
Result of human tracking

Number of persons tracked	159
Occurrences of correctly tracked person	123
Occurrences of losing track of person	5
Occurrences of tracking a wrong person	31

Section 3.4. We assigned 41 velocity bins for the TSV transform. We used 6 long sequences containing 182 persons. Table 5 shows the results of the human segmentation process (background subtraction, human extraction, TSV transform and binarization). Table 6 shows the results of tracking each human object.

Here, “correctly tracked person” refers to the number of persons whose motion was tracked as one person throughout.

In the TSV transform, we generate blobs by relating pixels that have similar positions and velocities. In principle, therefore, our system does not successfully separate blobs if their position and velocity values are similar. Many of the human segmentation errors shown above involve such blobs.

5.3. Interaction classification

We used 54 sequences to arrive at 10 interaction models. Each sequence represents one interaction sample. We computed each interaction model over 3–12 training sequences, depending upon the model. We present results from 70 sequences that include several SIUs. These test sequences were distinct from the training sequences. Table 7 shows the result of SIU recognition. The notation used in Table 7 is the same as Table 4.

“MEET,” “APART,” and “STOPGO” are the SIUs in which state transitions occur with both persons, whereas “STOP” and “LEAVE” have a state transition that involves one person. The start time of the state transitions involving two persons is an important factor. We assume that the two transitions are roughly simultaneous. However, a long interval between the state transition of one person and

Table 7
Result of interaction classification

SIU type	Number of training data	Number of test data	Correct	Percentage
APART	10	7	6	86
FOLLOW	3	5	3	60
LEAVE	4	13	10	77
MEET	12	11	9	82
PASS1	4	4	4	100
PASS2	4	9	7	78
STOP	5	11	9	82
TAKEOVER	4	2	1	50
WALKSTOP	6	3	3	100
STOPGO	2	5	4	80
Total	54	70	56	80.0

the transition of the other may cause difficulty in recognition. For example, we assume that two persons start walking roughly at the same time in “APART.” However, if one person starts walking and then the other person starts walking after a certain delay, the system may recognize the sequence as “LEAVE.” The recognition depends on the interval of the start timings. The larger the interval, the less likely it is that the interaction will be correctly recognized.

In “FOLLOW” and “WALKSTOP,” recognition depends on the distance between the participants, as well as on the start timing. A “FOLLOW” sequence is, for instance, distance-dependent. If the participating persons are positioned far from each other, the system may not recognize “FOLLOW.”

Incorrect recognition of “TAKEOVER” arises from tracking error. As discussed earlier, our system cannot successfully separate blobs if their position and velocity values are similar. The “TAKEOVER” sequence has blobs with such situations.

5.4. Interaction images

Fig. 16, the frames are ordered as (A), (B), (C), and then (D). Each image consists of four sub-images. The top-left sub-image is the original image, the top-right is the background subtracted binary image, the bottom-left is the TSV image and the bottom-right is the binary TSV image. In the original image, each tracked human is marked with a rectangle. When the system detects an interaction, an icon appears at the bottom edge of the original image. We use 10 icons as shown in Table 4. The vertical axes of the bottom two sub-images, corresponding to the TSV and the binary TSV images, respectively, are the velocity axes. The horizontal axes of all of the sub-images correspond to the same coordinate axis of the original image.

In the sequence in Fig. 16, two persons move from different directions and meet, then walk back to their original position. The system tracked these persons correctly and successfully recognized the SIUs “MEET” and “APART.” In addition, tree movement and shadows did not influence the binary images or the recognition of SIU’s.

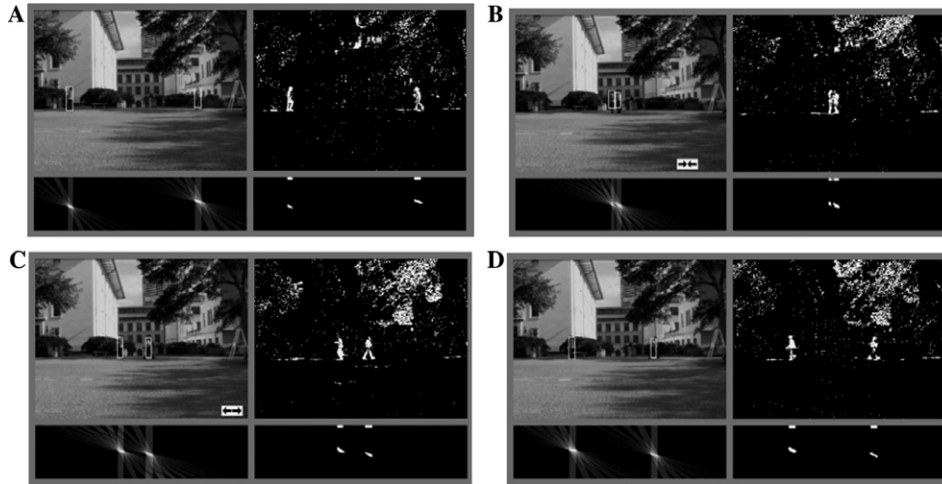


Fig. 16. Two persons meet from different directions and start walking in different directions.

In Figs. 17–19, frames are ordered left to right. Each frame contains the original image (upper) and the TSV image (lower).

The sequence in Fig. 17 shows three persons passing one another. In the first frame, the system recognize the two persons as a single person located at the left side of the sub-image, because their positions were very close. In the subsequent frames,



Fig. 17. Three people pass each other.

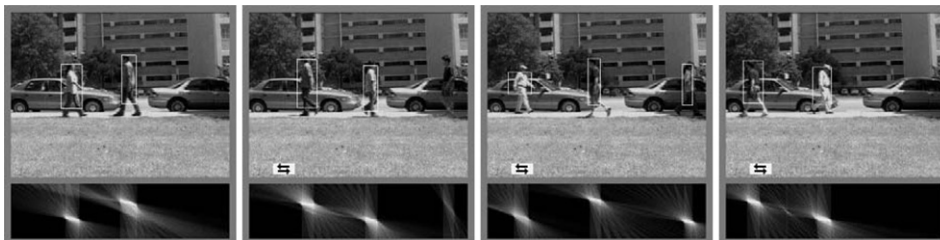


Fig. 18. A closer view of several people passing each other.

where the subjects were not as close, the system correctly recognized three persons. In addition the system recognized the SIU ‘PASS2’ successfully.

The image sequence in Fig. 18 provides a closer view of the persons involved in the sequence. In this sequence, the human blobs move faster than in the earlier two sequences. Also, this sequence contains four persons who pass one another several times. Despite the increased complexity of the scene, the system gives the correct result in both tracking and interaction recognition.

The sequence in Fig. 19 involves two persons, with one person following the other. The correct icon shows up in the right side image, showing the correct result.

The sequence in Fig. 20 involves two persons, with one person leaving the other who remains standing still. The ‘LEAVE’ icon $\blacklozenge \rightarrow$ appears at the bottom of the right side image. Again, the system produced correct tracking and recognition results.

The sequence in Fig. 21 is a relatively long sequence, involving seven persons. The interaction ‘MEET’ occurs in the fourth image, and ‘PASS2’ takes place in the second, third and seventh images. The respective appearance of the correct icon indicates that our system yields desirable results.



Fig. 19. One person follows another person.



Fig. 20. One person leaves after two persons are standing.

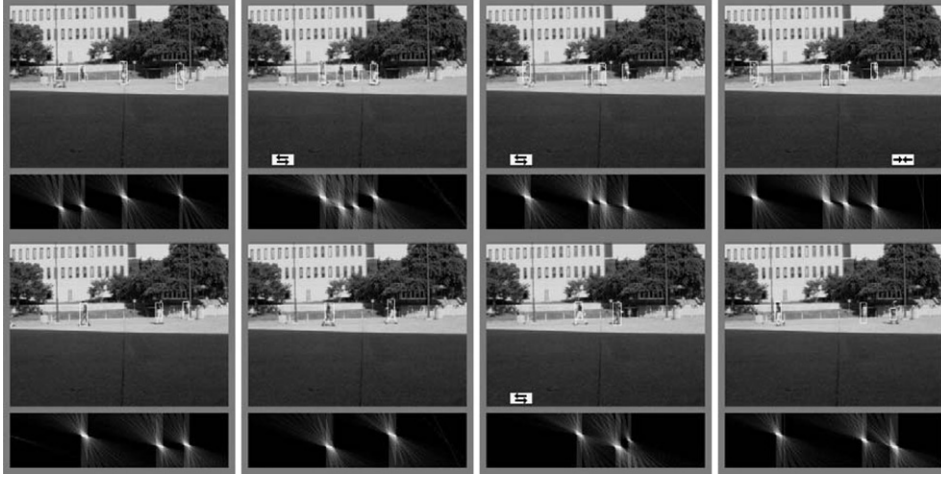


Fig. 21. Other longer scenes, several persons passing each other, meeting and leaving.

6. Conclusion

In this paper, we introduced the TSV transform and its application to human tracking and interaction recognition. The TSV transform, a combination of a windowing operation and a Hough transform over a spatio-temporal image, extracts pixel velocities from binary image sequences. It is noise suppressive and tracks occluding blobs correctly. In experiments, we segmented human blobs with more than 90% accuracy. Despite difficult conditions such as occluded or small objects, similar intensity levels for the background and the foreground, and the presence of much noise, the object persons were correctly tracked with more than 75% accuracy. Also, the system automatically recognizes 10 different interactions, defined as SIUs, with more than 80% accuracy. Its primary limitations is that our method cannot distinguish occluded persons moving at the same velocity, in which case the interaction is recognized as one person activity.

Appendix A

This appendix derives formula (12) that gives the bounds on acceleration. Suppose an object Φ with width w_Φ [pixels] moves at acceleration a [pixel/frame²]. Let n_Φ be the frame number, where $n_\Phi = 0$ at the current frame. Also, suppose the velocity and position of object Φ are 0 and $[0, w_\Phi]$ at the current frame ($n_\Phi = 0$), respectively (see Fig. 22). We may express the trajectory of object Φ as the following formulae,

$$x_0(n_\Phi) \leq x(n_\Phi) \leq x_1(n_\Phi), \quad (\text{A.1})$$

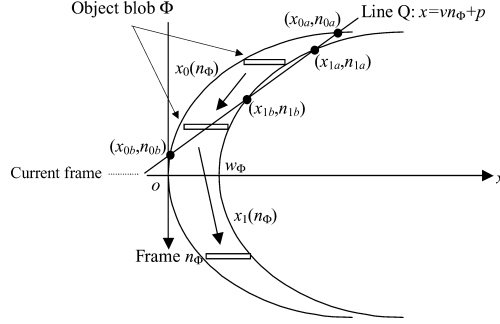


Fig. 22. Object blob movement with acceleration.

where

$$x_0(n_\phi) = \frac{1}{2} a n_\phi^2, \quad x_1(n_\phi) = \frac{1}{2} a n_\phi^2 + w_\phi.$$

Now we consider the maximum value of $V_n(p, v)$ at the present moment $n_\phi = 0$. ($n_\phi > 0$ is not defined for the future. We use only the $n_\phi \leq 0$ range, defined as present or past.)

Let line Q be $x = vn_\phi + p$ and the intersection points of line Q with $x = x_0(n_\phi)$ and $x = x_1(n_\phi)$ be (x_{0a}, n_{0a}) , (x_{0b}, n_{0b}) , and (x_{1a}, n_{1a}) , (x_{1b}, n_{1b}) , respectively.

So the n_{0a} , n_{0b} , n_{1a} , and n_{1b} can be computed:

$$\begin{aligned} n_{0a}, n_{0b} &= \frac{v - \sqrt{v^2 + 2ap}}{a}, \frac{v + \sqrt{v^2 + 2ap}}{a}, \\ n_{1a}, n_{1b} &= \frac{v - \sqrt{v^2 + 2a(p - w_\phi)}}{a}, \frac{v + \sqrt{v^2 + 2a(p - w_\phi)}}{a}. \end{aligned} \quad (\text{A.2})$$

From Eq. (7), $V_0(p, v)$ can be computed:

$$V_0(p, v) = \sum_{n_\phi=-\infty}^0 K_n(vn_\phi + p)(1 - e^{-\lambda})e^{\lambda n_\phi}, \quad (\text{A.3})$$

where in this case

$$K_{n_\phi}(x) = \begin{cases} 1 & (x, n_\phi) \text{ is within the range of (A.1),} \\ 0 & \text{otherwise.} \end{cases}$$

So (A.3) can be expressed using (x_{0a}, n_{0a}) , (x_{0b}, n_{0b}) , (x_{1a}, n_{1a}) , and (x_{1b}, n_{1b}) ,

$$V_0(p, v) = (1 - e^{-\lambda}) \left\{ \sum_{n_\phi=n_{0a}+1}^{n_{0b}} e^{\lambda n_\phi} - \sum_{n_\phi=n_{1a}+1}^{n_{1b}} e^{\lambda n_\phi} \right\}, \quad (\text{A.4})$$

where each term is equal to 0 if there is no solution with line M and $x = x_0(n_\phi)$ or $x = x_1(n_\phi)$.

Let D_0 and D_1 be defined as follows:

$$D_0 = \begin{cases} n_{0b} - n_{0a} & n_{0a}, n_{0b} \text{ are real,} \\ 0 & \text{otherwise,} \end{cases}, \quad D_1 = \begin{cases} n_{1b} - n_{1a} & n_{1a}, n_{1b} \text{ are real,} \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A.5})$$

$D_0 D_1$ is computed using Eqs. (A.2) and (A.5),

$$D_0 = \begin{cases} \frac{2\sqrt{v^2 + 2ap}}{a} & n_{0a}, n_{0b} \text{ are real,} \\ 0 & \text{otherwise,} \end{cases} \quad (\text{A.6})$$

$$D_1 = \begin{cases} \frac{2\sqrt{v^2 + 2a(p - w_\phi)}}{a} & n_{1a}, n_{1b} \text{ are real,} \\ 0 & \text{otherwise.} \end{cases}$$

So, Eq. (A.4) can be expressed using n_{0b} , n_{1b} , D_0 , and D_1 :

$$V_0(p, v) = e^{\lambda n_{0b}} (1 - e^{-\lambda D_0}) - e^{\lambda n_{1b}} (1 - e^{-\lambda D_1}). \quad (\text{A.7})$$

Since $\lambda > 0$, $V_0(p, v)$ is a monotonically increasing function in terms of n_{0b} and D_0 , and is a monotonically decreasing function in terms of n_{0b} and D_1 .

So $n_{0b} = 0$ and $D_1 = 0$ (when line Q is tangent to $x = x_1$ and (n_ϕ)) maximize $V_0(p, v)$. That is, $v = -\sqrt{2w_\phi a}$ and $p = 0$.

Thus, we obtain the value $V_0(p, v)$,

$$V_0(p, v) = 1 - e^{-2\sqrt{\frac{2w_\phi}{a}} \lambda}. \quad (\text{A.8})$$

The condition is that the value $V_0(p, v)$ is larger than the threshold Th_V ,

$$\lambda > -\frac{\ln(1 - Th_V)}{2\sqrt{\frac{2w_\phi}{a}}}. \quad (\text{A.9})$$

Thus,

$$a < 8w_\phi \lambda^2 \ln\left(\frac{1}{1 - Th_V}\right)^{-2}. \quad (\text{A.10})$$

For the negative range of a , we can derive similarly. The range of a is thus

$$-8w_\phi \lambda^2 \ln\left(\frac{1}{1 - Th_V}\right)^{-2} < a < 8w_\phi \lambda^2 \ln\left(\frac{1}{1 - Th_V}\right)^{-2}. \quad (12)$$

The condition for the arbitrary frame n_ϕ can be obtained analogously. Hence the acceleration range can be expressed as Eq. (12) Fig. 22.

Appendix B

This appendix derives the boundary of Region A expressed as y_a and y_b in Eq. (21). Region A is computed from physical camera parameters described in Fig. 23.

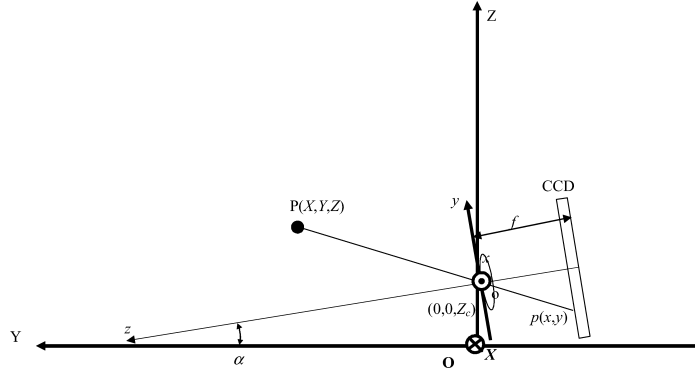


Fig. 23. Side view illustration of the relationship among coordinates.

(x, y)	CCD coordinate (horizontal, vertical)
o	CCD coordinate origin (lens center)
(X, Y, Z)	World coordinate. (horizontal, depth, and vertical)
O	Coordinate origin located at the ground position right under the camera
α	Angle of depression
F	Focal distance of camera
Z_c	Camera height

The relationships between the world coordinate (O - XYZ coordinate) and the CCD coordinate (the o - xyz coordinate) are:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90^\circ + \alpha) & \sin(90^\circ + \alpha) & 0 \\ 0 & -\sin(90^\circ + \alpha) & \cos(90^\circ + \alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -Z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & \sin \alpha & \cos \alpha & -Z_c \cos \alpha \\ 0 & \cos \alpha & -\sin \alpha & Z_c \sin \alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \quad (\text{B.1})$$

The coordinate (x, y) on the CCD surface, or the $z = -f$ plane, can be obtained by multiplying $-\frac{f}{z}$,

$$\begin{cases} x = \frac{fX}{Y \cos \alpha - (Z - Z_c) \sin \alpha} \\ y = -f \tan \alpha - \frac{f(Z - Z_c)(\tan^2 \alpha + 1)}{Y - (Z - Z_c) \tan \alpha} \end{cases} \quad (\text{B.2})$$

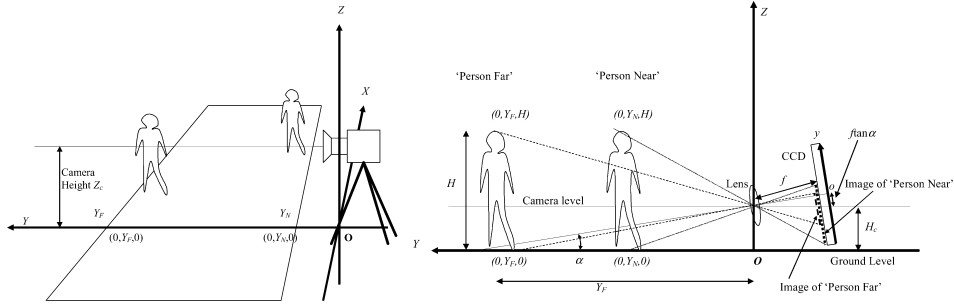


Fig. 24. The coordinates of the path and humans, and side view illustration of humans in relation to camera.

Fig. 24 shows the coordinates of the path and humans and a side view illustration of the humans in relation to the camera. ‘Person Far’ and ‘Person Near’ are the persons standing at the farthest and nearest locations from the camera, respectively. Y_F and Y_N are the distance of ‘Person Far’ and ‘Person Near’ from the camera, respectively. Suppose a person with a height H ($H > Z_c$) is standing between ‘Person Far’ and ‘Person Near.’ The head position y_h and foot position y_f in the image plane are:

$$\begin{aligned} y_h &= -f \tan \alpha - \frac{f(H - Z_c)(\tan^2 \alpha + 1)}{Y_P - (H - Z_c) \tan \alpha}, \\ y_f &= -f \tan \alpha + \frac{fZ_c(\tan^2 \alpha + 1)}{Y_P + Z_c \tan \alpha}, \end{aligned} \quad (\text{B.3})$$

where Y_P ($Y_N \leq Y_P \leq Y_F$) is the distance of the person from the camera.

Region A is the intersecting area of all the standing object images. So the following formulae for any value of Y_P ($Y_N \leq Y_P \leq Y_F$) must be satisfied:

$$y_f \leq y_b \leq y_a \leq y_h. \quad (\text{B.4})$$

From Eq. (B.4), it follows that the minimum value of y_h is equal to y_a and that the maximum value of y_f is equal to y_b . Since f , H , Z_c , and α are constant values, y_h is a monotone decreasing and y_f a monotone increasing function of Y_P . So y_a and y_b are the values of y_h and y_f when $Y_P = Y_F$. Thus,

$$\begin{aligned} y_a &= y_h|_{Y_P=Y_F}, \\ y_b &= y_f|_{Y_P=Y_F}, \end{aligned} \quad (\text{B.5})$$

or,

$$\begin{aligned} y_a &= -\frac{f(H - Z_c)(\tan^2 \alpha + 1)}{Y_F - (H - Z_c) \tan \alpha} - f \tan \alpha, \\ y_b &= \frac{fH(\tan^2 \alpha + 1)}{Y_F + Z_c \tan \alpha} - f \tan \alpha. \end{aligned} \quad (\text{B.6})$$

In other words, y_a and y_b are the foot and head points of ‘Person Far’ (Fig. 24). In our experiment, exact values for f and α are difficult to obtain by measurement. Instead, we obtain y_a and y_b directly from the image using Eq. (B.5).

References

- [1] J.K. Aggarwal, Q. Cai, Human motion analysis: a review, *Comp. Vis. Image Understanding J.* 73 (3) (1999) 428–440.
- [2] K. Sato, J.K. Aggarwal, Tracking and Recognizing Two-person Interaction in Outdoor Image Sequences, in: 2001 IEEE Workshop on Multi-Object Tracking, Vancouver, CA, July 2001, pp. 87–94.
- [3] K. Sato, J.K. Aggarwal, Tracking Persons and Vehicles in Outdoor Image Sequences using Temporal Spatio-Velocity Transform, in: Int. Workshop on Performance Evaluation of Tracking and Surveillance, Kauai, Hawaii, December 9, 2001.
- [4] K. Sato, B.L. Evans, J.K. Aggarwal, Designing an Embedded Video Processing Camera Using a 16-bit Microprocessor for Surveillance System, in: Int. Workshop on Digital and Computational Video, Clearwater, FL, November, 2002, pp. 151–158.
- [5] N. Oliver, B. Rosario, A. Pentland, A Bayesian computer vision system for modeling human interactions, in: Proc. Int. Conf. on Vision Systems '99, Gran Canaria, Spain, January 1999, pp. 255–272.
- [6] S. Niyogi, E. Adelson, Analyzing gait with spatio-temporal surface, in: Proc. 1994 IEEE Workshop on Non-rigid Motion and Articulated Objects, Austin, TX, 1994, pp. 64–69.
- [7] I. Haritaoglu, D. Harwood, L. Davis, W4: Who, When, Where, What: a real time system for detecting and tracking people, in: Third International Conference on Automatic Face and Gesture, Nara, Japan, April 1998, pp. 222–227.
- [8] I. Haritaoglu, L. Davis, Hydra: multiple people detection and tracking using silhouettes, in: IEEE Workshop on Visual Surveillance, Fort Collins, CO, June 1999, pp. 6–13.
- [9] Q. Cai, J.K. Aggarwal, Tracking human motion in structured environments using a distributed-camera system, *IEEE Trans. Pattern Anal. Mach. Intell.* 21 (12) (1999) 1241–1247.
- [10] D. Ayers, M. Shah, Recognizing human action in a static room, in: Proc. IEEE Comput. Soc. Workshop on Interpretation of Visual Motion, Princeton, NJ, 1998, pp. 42–46.
- [11] J. Davis, A. Bobick, The recognition of human movement using temporal templates, *IEEE Trans. Pattern Anal. Mach. Intell.* 23 (3) (2001) 257–267.
- [12] A.F. Bobick, Computers seeing action, in: Proc. Br. Mach. Vis. Conf., Edinburgh, Scotland, vol. 1, September 1996, pp. 13–22.
- [13] J. Yamato, J. Ohya, K. Ishii, Recognizing human action in time-sequential images using hidden Markov model, in: Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition, 1992, pp. 379–385.
- [14] T. Syeda-Mahmood, Segmenting Actions in Velocity Curve Space, in: 16th International Conference on Pattern Recognition, Quebec City, QC, Canada, vol. 4, August 2002, pp. 170–175.
- [15] T. Syeda-Mahmood, A. Vasilescu, S. Sethi, Recognizing action events from multiple viewpoints, in: IEEE Workshop on Detection and Recognition of Events in Video, Vancouver, BC, Canada, July, 2001, pp. 64–71.
- [16] B.K. Horn, B.G. Schunck, Determining Optical flow, *Artif. Intell.* 17 (1981) 185–273.
- [17] J.D. Kim, S.D. Kim, J.K. Kim, Fast convergent method for optical flow estimation in noisy image sequences, *Electron. Lett.* 25 (1) (1989) 74–75.
- [18] C.P. Chong, C.A. Salama, K.C. Smith, A novel technique for image-velocity computation, *IEEE Trans. Circuit Syst. Video Technol.* 2 (3) (1992) 313–318.
- [19] E. Williams, *Regression Analysis*, Wiley, New York, 1959, pp. 10–22.