Tracking Objects in Occluding Environments using Temporal Spatio-Velocity Transform

Koichi Sato and J. K. Aggarwal
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712, USA
koichisato@alumni.utexas.net , aggarwaljk@mail.utexas.edu

Abstract

This paper presents a methodology for tracking moving objects in an occluded environment when occlusion occurs. We analyze sequences in which physical obstacles such as fences and trees divide an object into several blobs, both spatially and temporally. Our system successfully tracks the divided blobs as one object and reconstructs the whole object. We use the temporal spatio-velocity (TSV) transform and a cylinder model of object trajectories. The TSV transform extracts pixels with stable velocities and removes noisy pixels with unstable velocities. The cylinder model connects several blobs into one object and associates blobs that are occluded for a long period of time. We present results in which moving persons and vehicles occluded by fences and trees are successfully tracked even when the occlusion lasts for as long as 100 frames.

1 Introduction

Tracking persons and objects is an important problem in computer vision that has received significant attention over the last several decades. In [1], Oliver *et al.* segmented humans by subtracting an eigenbackground, which was generated using Principal Component Analysis (PCA) from several static background images. They tracked humans based on their position and velocity, estimated using a Kalman filter and the Probability Density Function (PDF) in each color component of the human blob. In [2], Niyogi and Adelson segmented humans using Hough transform line detection on a spatio-temporal sliced image of background-subtracted image sequences. They assumed that the subject crosses the field of view at a constant speed; that is, the human trajectories appear as straight lines on the spatio-temporal sliced image. This method is useful for extracting a human trajectory; however, it does not apply to image sequences that contain persons or vehicles moving randomly (stopping and starting), in which case the object trajectories do not appear as straight lines.

In [3,4], Haritaoglu *et al.* tracked multiple people using the silhouette and texture of the human images. They tracked humans by matching the segmented contour of the humans as well as by matching the head position. Then, they identified each individual in the group of people using temporal texture templates. Syeda-Mahmood [5] used the idea of an action cylinder. Syeda-Mahmood, Vasilescu and Sethi [6] 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, an important component of our algorithm is pixel velocity extraction. It is thus helpful to discuss some of the significant contributions on this issue. Horn and Schunk [7] proposed an optical flow approach to extract pixel velocities using grayscale or color image sequences under the smooth surface constraints. Kim *et al.* [8] advanced Horn and Schunk's approach using a fast convergent method. Chong, Salama and Smith [9] 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.

Our previous system [10] segments and tracks persons and recognizes two-person interactions in outdoor side-view image sequences. The system segments an image containing a human using background subtraction, followed by object extraction using the temporal spatio-velocity (TSV) transform. Human tracking uses several features such as texture and blob size. The TSV transform is a technique to extract velocities of moving pixels from a sequence of binary images. The input sequence is composed of a one-dimensional binary image from which the TSV transform generates a two-dimensional grayscale image sequence with horizontal x axis and velocity v_x axes. Then, we extend the TSV transform to use two-dimensional velocity [11] in order to track objects in perspective view images. The TSV image, which is the output of the TSV transform, is a 4-dimensional image sequence that consists of horizontal and vertical position axes (x, y) and horizontal and vertical velocity axes (v_x, v_y).

0-7803-9424-0/05/$20.00 ©2005 IEEE.

We track an object by finding a cylinder that expresses the trajectory of the object in spatio-temporal domain.

In this paper, we consider an occluding environment in which the camera is located behind trees or fences that block the camera view. Such an environment causes problems for object tracking for the following reasons. A fence divides an object image into several small "pieces", which makes it difficult to recognize them as one object. Trees generate noise by the movement of leaves. The tracked objects can disappear for a long period of time because trees or fences block the camera view. To overcome these problems, we use the TSV transform and a cylinder model for object trajectories based on our previous paper [11]. Since the TSV transform effectively removes pixels with unstable velocities, our system can remove the noise from the movement of leaves. Divided image blobs are connected using a cylinder. We consider a trajectory of an object as a cylinder in the spatio-temporal image domain. The cross section of the cylinder is an ellipse that represents an object at a frame. By fitting the pixels into that cylinder, we successfully track the object.

The contributions of this paper are as follows: (1) Stable tracking in outdoor perspective-view images containing noise. (2) Velocity-based noise removal using the TSV transform. (3) Tracking objects that are occluded temporally and spatially for a "long" period of time.

This paper is organized as follows: in section 2, the basics of the TSV transform are described. In section 3, the cylinder fitting technique is briefly mentioned. In section 4, an overview of our system is presented. Results and conclusion are described in section 5 and section 6, respectively.

2 TSV Transform

The TSV transform is a method of extracting pixel velocities from binary image sequences. We proposed this transform as a means of tracking [10,11]. In this application, the TSV transform is used for extracting the velocity-stable pixels and separating the foreground object from backgrounds.

We consider the TSV transform over the binary image sequence $K_n(\mathbf{x})$ where n is the current frame, and $\mathbf{x}=(x, y)^T$ is the position at nth frame. The TSV transform is the combination of the Hough transform and a low-pass filter. Let an arbitrary frame be k, positioned at the kth frame $\mathbf{y}=(x, y)^T$. The Hough transform in terms of straight line $\mathbf{y}=\mathbf{v}(k-n)+\mathbf{x}$ over a binary spatio-temporal image $K_k(\mathbf{y})$ is defined as:

$$H(\mathbf{x},\mathbf{v})=\sum_{k=0}^{n} K_k(\mathbf{v}(k-n)+\mathbf{x}), \quad (1)$$

The Hough transform extracts straight lines that are the trajectories of pixels moving at a constant speed from frames 0-n. However, within a certain range, objects do not always move at a constant speed. In order to extract pixels moving at a constant speed only around nth frame, we apply a windowing operation over the binary image by an exponential window $F_n(k)$,

$$F_n(k)=\begin{cases}(1-e^{-\lambda})e^{\lambda(k-n)} & k\le n\\ 0 & k>n.\end{cases} \quad (2)$$

The TSV image $V_n(\mathbf{x}, \mathbf{v})$ is obtained as follows.

$$V_n(\mathbf{x},\mathbf{v})=\sum_{k=-\infty}^{n} F_n(k)K_k(\mathbf{v}(k-n)+\mathbf{x}), \quad (3)$$

or,

$$V_n(\mathbf{x},\mathbf{v})=(1-e^{-\lambda})\sum_{k=-\infty}^{n} e^{\lambda(k-n)}K_k(\mathbf{v}(k-n)+\mathbf{x}). \quad (4)$$

In actual implementation, we use the simplified version of (4),

$$V_n(\mathbf{x},\mathbf{v})=e^{-\lambda}V_{n-1}(\mathbf{x}-\mathbf{v},\mathbf{v})+(1-e^{-\lambda})K_n(\mathbf{x}). \quad (5)$$

Figure 1 illustrates the implementation diagram of the TSV transform.

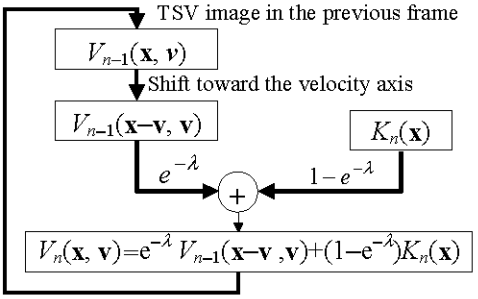

Figure 1. Implementation Diagram of the TSV Transform

2.1 Time Constant

The TSV image extracts the pixel velocities; however, if the pixels have a large acceleration, TSV may fail to extract the velocity. The time constant λ in the TSV transform determines the acceleration range of pixels. When λ is large, it accepts a smaller acceleration range, and *vice versa* when λ is small. We can determine the value of λ using the acceptable acceleration a, object width w_Φ, and the threshold of the TSV T_V. The details are described in [10],

$$\lambda > -\frac{\ln(1-T_V)}{2\sqrt{\frac{2w_\Phi}{a}}}. \quad (6)$$

In a perspective-view image, object acceleration is not proportional to the actual acceleration. A closer object shows larger acceleration than a farther one. Suppose two objects move at the same acceleration, but one object is distant and the other is close to the camera. The acceleration of the closer object appears larger than that of the more distant object in the perspective-view image sequences. In order for the TSV transform to accept a constant acceleration range, we use different time constant values for different vertical locations in the image.

Specifically, we set a higher time constant value at a lower vertical location.

The time constant map for such an image sequence would look like Figure 2, in which the intensity of each pixel represents the time constant value. (High intensity means a high time constant value). As discussed earlier, object acceleration is larger when the object comes closer to the camera, thus closer parts are darker in our time-constant map.

Figure 2. Time Coefficient for the TSV transform in perspective-view image

2.2 Noise removal of unstable velocities

As we discussed above, the TSV transform extracts velocity stable pixels. Thus, pixels with unstable velocity are removed. Figure 3 shows every fifth frame (330msec) of an image sequence that contains vehicles moving horizontally. Each arrow shows the position of a vehicle. Figure 4 shows the binary images corresponding to the image sequence in Figure 3, and Figure 5 is the result of the TSV transform over the images in Figure 4.

We use horizontal velocities v_x for the TSV transform. This means that the TSV results are originally three-dimensional images consisting of x, y and v_x. In this experiment, however, we superimposed them along the velocity axis into two-dimensional images shown in Figure 5. Because of the movement of leaves, the binary images have so much noise that it is difficult for us to tell the object silhouettes from the noise. If we have information on the velocity range of the objects, the TSV transform efficiently removes the noise. In the sequence shown in Figure 4, the vehicles move at velocities with a certain range. Thus, we set the TSV velocity range to that velocity range. We can see in Figure 5 that the noise from the movement of leaves is suppressed effectively using the TSV transform.

3 Spatio-Temporal Cylinder Model

In the spatio-temporal domain consisting of positional coordinates $\mathbf{x}=(x, y)^T$ and temporal coordinate n, the trajectory of a moving blob forms a shape like a curved rod whose cross section is the blob shape. We approximate the blob shape as an ellipse and the trajectory as a cylinder. We assume that objects do not change their acceleration in a short period of time. So we approximate the cylinder axis using a second-degree polynomial expression.

$$\mathbf{x} = \frac{1}{2}\mathbf{a}n^2 + \mathbf{v}n + \mathbf{p}, \qquad (7)$$

where $\mathbf{x}=(x, y)^T$ is the position at nth frame, $\mathbf{a}=(a_x, a_y)^T$ is the acceleration, $\mathbf{v}=(v_x, v_y)^T$ is the velocity and $\mathbf{p}=(x_0, y_0)^T$ is the position at frame 0. We compute the parameters using pixels in recent T frames where T is the number of frames. Let a pixel position in an object be $\mathbf{x}_i$ and its frame number be n_i. The parameters $\mathbf{a}$, $\mathbf{v}$, and $\mathbf{p}$ are computed by regression analysis,

Figure 3. Original Images

Figure 4. Binary Images after Background Subtraction

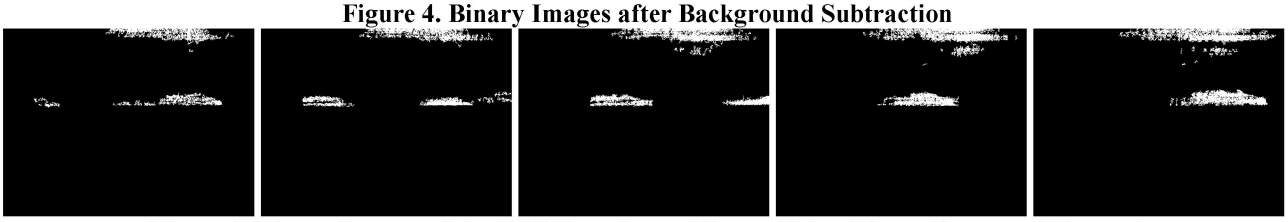

Figure 5. Binary Images of the TSV Transformed Image Using Binary Images of Figure 4

$$\begin{bmatrix} \mathbf{a}/2 \\ \mathbf{v} \\ \mathbf{p} \end{bmatrix} = \left\{ \underset{i}{\text{avg}} \begin{bmatrix} n_i^4 & n_i^3 & n_i^2 \\ n_i^3 & n_i^2 & n_i \\ n_i^2 & n_i & 1 \end{bmatrix} \right\}^{-1} \left\{ \underset{i}{\text{avg}} \begin{bmatrix} \mathbf{x}_i n_i^2 \\ \mathbf{x}_i n_i \\ \mathbf{x}_i \end{bmatrix} \right\}, \quad (8)$$

where $\underset{i}{\text{avg}}(\ )$ represents the average. We approximate the sectional shape of the cylinder as an ellipse. Its axis direction and radii are obtained using eigenvalue computation. Let a matrix $\mathbf{M}$ be

$$\mathbf{M} = \underset{i}{\text{avg}} \left\{ \left(\mathbf{x}_i - \frac{1}{2}\mathbf{a}n_i^2 - \mathbf{v}n_i - \mathbf{p} \right) \left(\mathbf{x}_i - \frac{1}{2}\mathbf{a}n_i^2 - \mathbf{v}n_i - \mathbf{p} \right)^T \right\}. \quad (9)$$

The eigenvectors $\mathbf{e}_1$ and $\mathbf{e}_2$ of $\mathbf{M}$,

$$\mathbf{e}_{1,2} = \text{eigenvector}\{\mathbf{M}\}, \quad (10)$$

yields the direction of the ellipse axis. The long and short radii can be computed using eigenvalues,

$$\text{radius}_{long,short} = 2\sqrt{\text{eigenvalue}\{\mathbf{M}\}}, \quad (11)$$

The details of the cylinder parameter computation are described in the appendix.

3.1 Merging Process

We assume that one cylinder represents one object. However, there are cases where parts of a body are recognized as multiple cylinders. A way to avoid this is to merge such cylinders by assuming that blobs moving at same velocity and in similar locations belong to one object. In order to judge whether two given cylinders belong to one object, we use the surface and cross section areas of the cylinders.

Let us consider two cylinders A and B. Let the lengths of the long and short radii of the cylinders be expressed as a_A, b_A, a_B and b_B, which are computed using equation (11). Suppose that Cylinder A exists from the n_{SA}th frame until the n_{EA}th frame and Cylinder B from the n_{SB}th frame until the n_{EB}th frame. Note that $(n_{EA}-n_{SA}) \le T$ and $(n_{EB}-n_{SB}) \le T$, as the only pixels between the nth and the $(n-T)$th frames are used for computation. The sectional area C_A, C_B and surface areas S_A, S_B are then expressed as follows.

$$\begin{cases} C_A = a_A b_A \pi \\ C_B = a_B b_B \pi \\ S_A = C_A + (n_{EA} - n_{SA})(a_A + b_A)\pi \\ S_B = C_B + (n_{EB} - n_{SB})(a_B + b_B)\pi \end{cases}. \quad (12)$$

Now, suppose that we determine whether cylinders A and B can be merged. We compute the parameters according to equation (11) using all pixels that belong to only A or B. Let the composed lengths of the long and short radii be a_C, b_C. Then, we obtain the slice C_c and the surface area S_c as follows.

$$\begin{cases} C_C = a_C b_C \pi \\ S_C = C_C + \{\max(n_{EA}, n_{EB}) - \min(n_{SA}, n_{SB})\}(a_C + b_C)\pi \end{cases}. \quad (13)$$

Merger depends on the following criterion,

$$\text{Merge if } C_C \le C_A + C_B \text{ and } S_C \le S_A + S_B. \quad (14)$$

Note that we do not use an adjustment parameter such as thresholding and or a coefficient.

4 System Overview

The system overview shown in Figure 6 illustrates the steps for object tracking. We begin by binarizing the original image. Applying the TSV transform to the binarized image, we extract only pixels with stable velocities. Each pixel joins the cylinder with the closest coordinates.

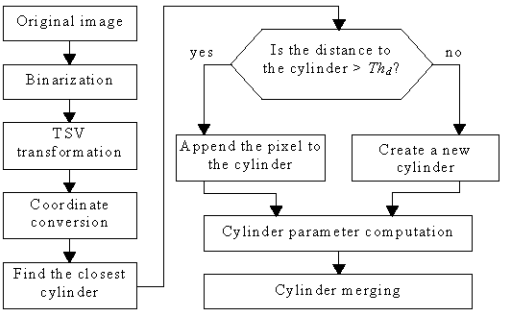

Figure 6 System Diagram

4.1 Binarization

We use simple background subtraction to extract foreground blobs. Let the original image at the current frame be $\mathbf{I}_n(\mathbf{x})$ and a fixed background image be $\mathbf{B}(\mathbf{x})$. The foreground binary image $K_n(\mathbf{x})$ is obtained where $\mathbf{I}_n(\mathbf{x})$ is more than a fixed threshold Th_B,

$$K_n(\mathbf{x}) = \begin{cases} 1 & \|\mathbf{I}_n(\mathbf{x}) - \mathbf{B}(\mathbf{x})\| > Th_B \\ 0 & \text{otherwise} \end{cases}. \quad (15)$$

The binary image sequence $K_n(\mathbf{x})$ may contain a number of noise pixels, which will be removed at a later stage, described below. using the TSV transform.

4.2 TSV Transform

Since the TSV transform extracts velocity-stable pixels within a specific velocity range, in this work we use it to effectively remove velocity-unstable noise such as tree movements, and successfully extract foreground objects whose velocities are different from background objects.

We apply the transform to the binary image sequence obtained in the previous section.

$$V_n(\mathbf{x}, \mathbf{v}) = \text{TSV}\{K_n(\mathbf{x})\}. \quad (16)$$

The obtained TSV image $V_n(\mathbf{x}, \mathbf{v})$ is a four-dimensional image consisting of $\mathbf{x}=(x, y)^T$ and $\mathbf{v}=(v_x, v_y)^T$. By thresholding this image using a fixed value T_V, we get the binary image sequence $V_n^*(\mathbf{x}, \mathbf{v})$.

$$V_n^*(\mathbf{x}, \mathbf{v}) = \begin{cases} 1 & \text{if } V_n(\mathbf{x}, \mathbf{v}) \ge T_V \\ 0 & \text{otherwise} \end{cases}. \quad (17)$$

The binary TSV image $V_n^*(\mathbf{x}, \mathbf{v})$ is also four-dimensional. Each image $V_n^*(\mathbf{x}, \mathbf{v}_i)$ at a velocity value $\mathbf{v}_i$ is a two-dimensional image from which only pixels moving with $\mathbf{v}_i$ are extracted. We perform an OR-operation to obtain pixels with the entire velocity range used in the TSV transform,

$$V_n^{\#}(\mathbf{x}) = \underset{\mathbf{v}}{\text{OR}}\{V_n^*(\mathbf{x}, \mathbf{v})\}. \quad (18)$$

We set the velocity range so that it covers the possible velocity values that an object and its parts may have. For example, in an image sequence of a highway, every vehicle will be moving within a range of $\mathbf{v} \in [60,120]$ (km/h). Thus, we set the velocity range for the TSV transform to that value, in order to extract objects moving within that range.

4.3 Coordinate Conversion

The location of an object in an image and its actual position are not linearly related. Object size in an image is smaller when the object is distant from the camera, are larger when the object is close. We use cylinders to represent the object trajectories in which the object size does not change over frames. We convert the image coordinate into the world coordinate by assuming that objects are on the ground. The relationship between the image coordinate (x, y) and the world coordinate (X, Y) is

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} \dfrac{-Hx}{f\sin\alpha + y\cos\alpha} \\ \dfrac{-f\cos\alpha + y\sin\alpha}{f\sin\alpha + y\cos\alpha} H \end{pmatrix}. \quad (19)$$

where f is the focal length, α is the tilt angle, and H is the camera height, which are obtained from the training image sample.

4.4 Cylinder Formation

The previous processes use only the pixel data in the current frame, whereas the cylinder formation uses the pixel data in recent T frames. For each pixel in recent T frames, we find the closest cylinder using a pixel distance d. Let the coordinate of a point be $\mathbf{x}$, the frame n, and the cylinder parameters be $\mathbf{a}$, $\mathbf{v}$ and $\mathbf{p}$ as denoted in the formula (8). The distance d is computed,

$$d = \|\mathbf{a}n^2 + \mathbf{v}n + \mathbf{p} - \mathbf{x}\|. \quad (20)$$

These parameters are computed using T frames.

The closest cylinder is the candidate for the pixel to join. If the distance d is smaller than a fixed threshold Th_d, the pixel joins to the candidate cylinder, namely;

Append the pixel to the cylinder if $d < Th_d$, (21)

otherwise, the system creates a new cylinder. The initial values for its parameters are as follows:

$$\begin{cases} \mathbf{a} = 0 \\ \mathbf{v} = \text{median of the velocity range} \\ \mathbf{p} = \text{pixel position} \end{cases} \quad (22)$$

After processing all points in the current frame, the system computes the cylinder parameter based on the points in the cylinder.

5 Results

We used the image sequences with the specifications in Table 1. Figure 7 shows the result of a tracking experiment involving a person moving behind trees. Figure 8 shows

Table 1. Specifications

Size	640x480
Color depth	R(8bit) G(8bit) B(8bit)
Frame rate	15 [frames/sec]
CPU	Pentium 4, 1GHz

Figure 7. Tracking a moving person behind trees

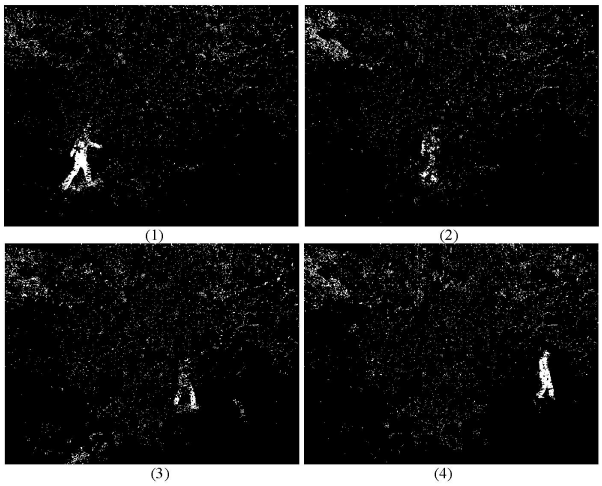

Figure 8. Binary image after background subtraction

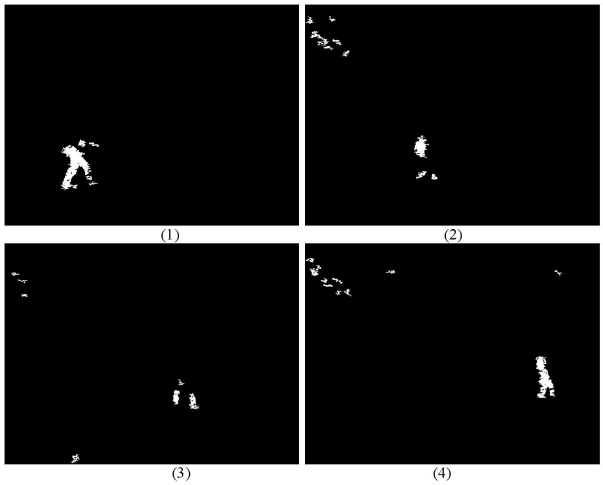

Figure 9. Binary image after the TSV transform

binary images of the images in Figure 7 after background subtraction. Figure 9 presents binary images of the images in Figure 8 after the TSV transform. Ellipses represent results. Images are shown in chronological order. Tree leaves occlude the moving person at (2) and (3). We use a velocity range of 0.4 to 0.7 [pixel/frame] so that the range covers all possible velocities of the parts of an ordinary walking human such as the torso and limbs. If we compare Figures 8 and 9, we see that the TSV transform successfully removes noisy pixels that are velocity-unstable. Because the cylinder model covers the object throughout the occlusion, the system tracks the moving person correctly.

Figure 10 shows the result of a tracking experiment that involves moving vehicles behind trees. We set a camera beside a one-way road without signals. Thus, vehicles move horizontally, and their speeds are mostly constant. For this sequence, we use a velocity range of between –35 and –25 [pixel/frame], which covers only moving vehicles. Figure 11 shows the result of background subtraction. As the image contain so much noise, it is difficult to distinguish the vehicle silhouette from the noise. Figure 12 is the result of the TSV transform over the binary image shown in Figure 11. The TSV transform eliminates all noise that is out of the vehicles' velocity range.

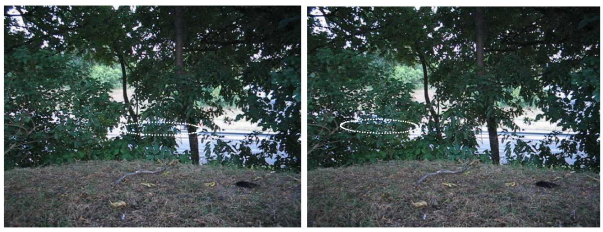

Figure 10. The result of tracking vehicles behind trees

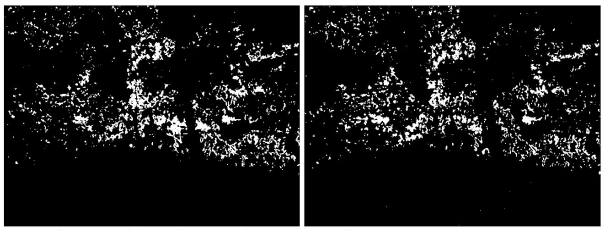

Figure 11. Binary images after the background subtraction

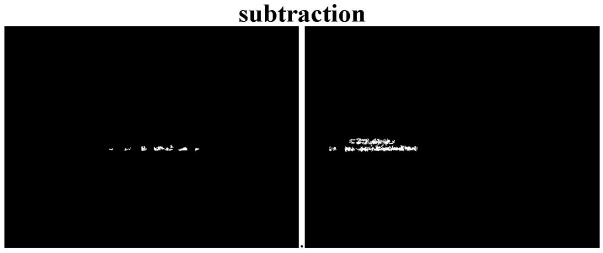

Figure 12 Binary images after the TSV transform

Figures 13, 16 and 19 show the result of a tracking experiment that involves vehicles behind fences, Figures 14, 17 and 20 show their respective binary images after background subtraction, and Figures 15, 18 and 21 are binary images after the TSV transform over these images. Unlike the trees that partially occluded the field of view in the sequences shown in Figures 7 and 10, the occlusion caused by fences is uniform. Figures 13 and 16 show sequences of vehicles occluded by a mesh screen fence, moving away from the camera, and moving toward the camera, respectively. In Figure 19, the camera is located behind a chain link fence and the vehicles are moving toward the camera. The velocity ranges of the TSV transform are [–1, –8], [4, 22], and [1, 10] [pixel/frame], respectively. These ranges separate the object velocity from the background velocity, which is zero. In all cases, the moving vehicles are tracked correctly.

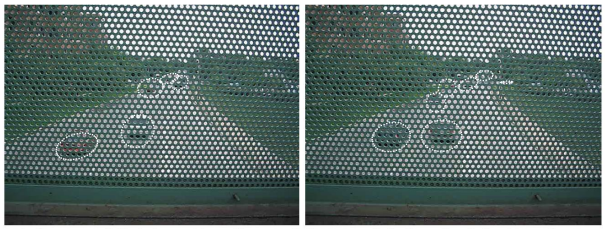

Figure 13. Result of tracking vehicles behind a fence (1)

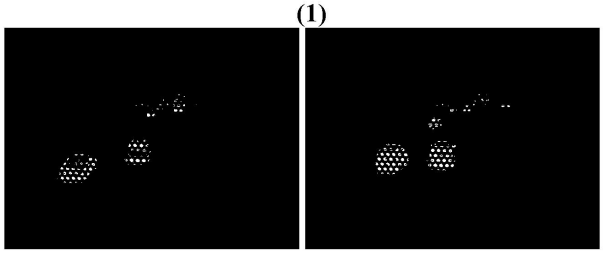

Figure 14. Binary image after background subtraction of Figure 13.

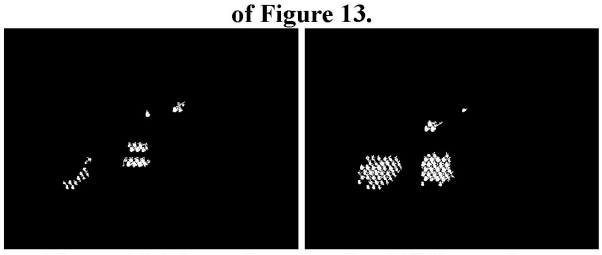

Figure 15. Binary images after TSV transform

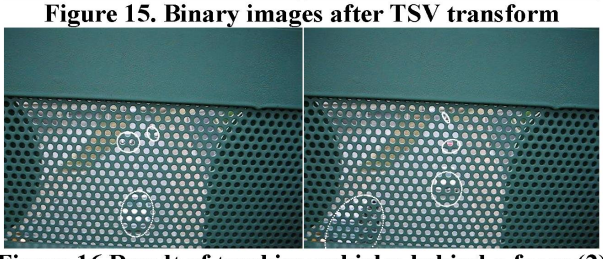

Figure 16 Result of tracking vehicles behind a fence (2)

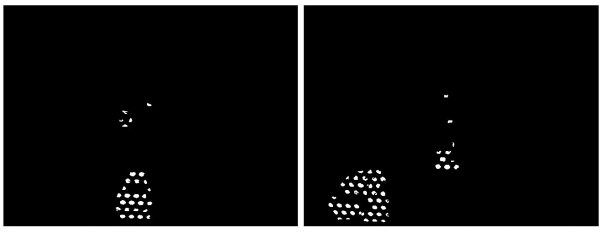

Figure 17. Binary images after background subtraction of Figure 16.

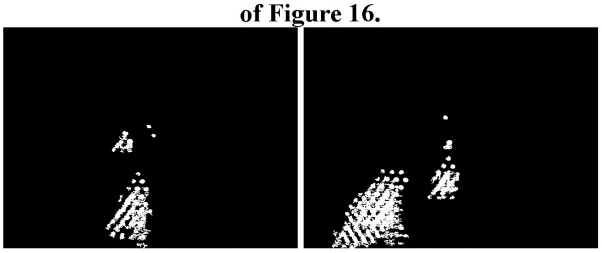

Figure 18. Binary images after the TSV transform

Figure 19. Result of tracking vehicles behind a chain link fence (3).

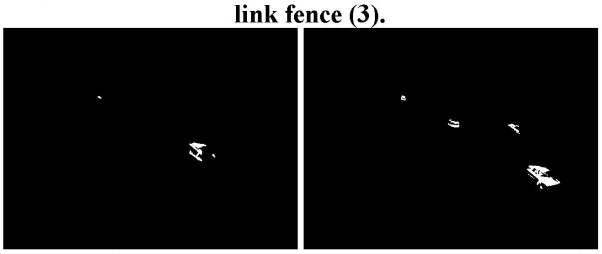

Figure 20. Binary image after background subtraction of Figure 19.

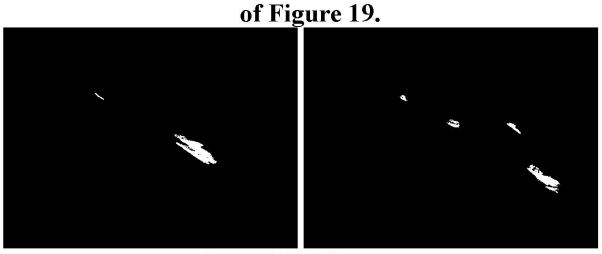

Figure 21. TSV Image

6 Conclusion

We presented a methodology for tracking moving persons and vehicles in outdoor image sequences in an occluded environment. We use the TSV transform to reduce noise from tree movements by removing the velocity-unstable pixels, and to separate objects whose velocities are different from background ones. We track objects using the cylinder model that expresses the object trajectories. This model connects small blobs that are divided by fences or trees spatially or temporally. As a result, our system tracked objects that are constantly occluded more than 30 pixel length or 100 frames.

7 References

[1] N. Oliver, B. Rosario and A. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions", *IEEE Transactions on Pattern Analysis and Machine Interlligence*, Vol. 22, No.8, pp. 831-843, Aug. 2000.

[2] S. Niyogi and E. Adelson, "Analyzing gait spatio-temporal surface", *Procedings of the 1994 IEEE Workshop*, pp.64-69, 1994.

[3] I. Haritaoglu, D. Harwood and L. Davis, "W4: Who, When, Where, What: A real time system for detecting and tracking people", *Third International Conference on Automatic Face and Gesture, Nara,* pp 222-227, April 1998.

[4] I. Haritaoglu and L. Davis, "Hydra: Multiple people detection and tracking using silhouettes", *IEEE Workshop on Visual Surveillance*, pp.6-13, 1999.

[5] T. Syeda-Mahmood, "Segmenting Actions in Velocity Curve Space", *16th International Conference on Pattern Recognition*, Quebec City, QC, Canada, vol. 4, pp.170-175, August 2002.

[6] T. Syeda-Mahmood, A. Vasilescu and S. Sethi, "Recognizing action events from multiple viewpoints", *IEEE Workshop on Detection and Recognition of Events in Video*, Vancouver, BC, Canada, pp.64-71, July, 2001.

[7] B. K. Horn, and B. G. Schunck, "Determining Optical flow", *Artificial Intelligence*, vol. 17, pp. 185-273, 1981.

[8] J.D. Kim, S.D. Kim, J.K. Kim, "Fast convergent method for optical flow estimation in noisy image sequences", *Electronics Letters,* vol. 25, no. 1, pp.74 –75, January. 1989.

[9] C. P. Chong, C. A. Salama, and K. C. Smith, "A Novel Technique for Image-Velocity Computation", *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 2, no. 3, pp. 313-318, September 1992.

[10] K. Sato and J. K. Aggarwal, "Temporal spatio-velocity transform and its application to tracking and interaction", *Computer Vision and Image Understanding*, vol. 96, issue 2, Nov. 2004, pp. 100-128.

[11] K. Sato and J. K. Aggarwal, "Tracking objects using temporal spatio-velocity transform", *Proc. IEEE Work. on Performance Eval. of Tracking and Surveillance*, Kauai, Hawaii, Dec. 2001.

Appendix Cylinder Parameter Computation

1. Computation of Cylinder Axis

We approximate the points that constitute an object to the curve in (7) by means of regression. Let **x** be the position coordinate of each point and n the temporal coordinate of the point. We begin by calculating the average value δ, the distance between (7) and **x**,

$$\delta = \underset{i}{\text{avg}}\left(\left\{\mathbf{x}_i - \frac{1}{2}\mathbf{a}n_i^2 - \mathbf{v}n_i - \mathbf{p}\right\}^2\right). \quad (23)$$

We obtain $\mathbf{a}/2$, $\mathbf{v}$, and $\mathbf{p}$ that minimizes δ by computing

$$\frac{\partial\delta}{\partial(\mathbf{a}/2)} = \frac{\partial\delta}{\partial\mathbf{v}} = \frac{\partial\delta}{\partial\mathbf{p}} = 0. \quad (24)$$

So, we get

$$\operatorname*{avg}_i \begin{pmatrix} n_i^4 & n_i^3 & n_i^2 \\ n_i^3 & n_i^2 & n_i \\ n_i^2 & n_i & 1 \end{pmatrix} \begin{bmatrix} \mathbf{a}/2 \\ \mathbf{v} \\ \mathbf{p} \end{bmatrix} = \operatorname*{avg}_i \begin{bmatrix} \mathbf{x} n_i^2 \\ \mathbf{x} n_i \\ \mathbf{x} \end{bmatrix}, \tag{25}$$

or

$$\begin{bmatrix} \mathbf{a}/2 \\ \mathbf{v} \\ \mathbf{p} \end{bmatrix} = \left[\operatorname*{avg}_i \begin{pmatrix} n_i^4 & n_i^3 & n_i^2 \\ n_i^3 & n_i^2 & n_i \\ n_i^2 & n_i & 1 \end{pmatrix} \right]^{-1} \operatorname*{avg}_i \begin{bmatrix} \mathbf{x}_i n_i^2 \\ \mathbf{x}_i n_i \\ \mathbf{x}_i \end{bmatrix}. \tag{26}$$

2. Computing Cylinder Sectional Shape

We approximate the sectional shape of the cylinder to an ellipse. The parameters of the ellipse are: the long radius, the short radius and the direction of the ellipse axis. Let each point in the cylinder be $\mathbf{x}_i$ and the frame number of $\mathbf{x}_i$ n_i. We define a vector $\mathbf{y}_i$ that is the relative vector from the cylinder axis. $\mathbf{y}_i$ can be expressed using $\mathbf{x}_i$, n, acceleration $\mathbf{a}$, velocity $\mathbf{v}$, and position $\mathbf{p}$,

$$\mathbf{y}_i = \mathbf{x}_i - \left(\frac{1}{2}\mathbf{a} n_i^2 + \mathbf{v} n_i + \mathbf{p} \right). \tag{27}$$

The centroid of $\mathbf{y}$ is obtained by averaging,

$$\operatorname*{avg}_i \{\mathbf{y}_i\} = \operatorname*{avg}_i \{\mathbf{x}_i\} - [\frac{1}{2}\mathbf{a} \cdot \operatorname*{avg}_i \{n_i^2\} + \mathbf{v} \cdot \operatorname*{avg}_i \{n_i\} + \mathbf{p}]. \tag{28}$$

Substituting (26) into (28), we get

$$\operatorname*{avg}_i \{\mathbf{y}_i\} = 0. \tag{29}$$

In order to find the parameters of the ellipse, we consider a vector $\mathbf{e}$ whose length is one ($\mathbf{e}^2=1$). The inner product between $\mathbf{y}_i$ and $\mathbf{e}$ varies depending on the direction of $\mathbf{e}$. The absolute of the inner product $|\mathbf{y}_i^{\mathrm{T}}\mathbf{e}|$ has a maximum value when the direction $\mathbf{e}$ is parallel to $\mathbf{y}_i$, and takes the value of zero when $\mathbf{e}$ is perpendicular to $\mathbf{y}_i$. We find the direction of the ellipse using the variance of $\mathbf{y}_i^{\mathrm{T}}\mathbf{e}$. The variance takes a maximum value when e is parallel to the long axis and takes a minimal value when $\mathbf{e}$ is parallel to the short axis. The variance σ^2 is obtained

$$\sigma^2 = \operatorname*{avg}_i \{(\mathbf{y}_i^T \mathbf{e} - \operatorname*{avg}_i \{\mathbf{y}_i^T \mathbf{e}\})^2\} \tag{30}$$

Since $\operatorname*{avg}_i \{\mathbf{y}_i\}=0$ from (29),

$$\begin{aligned} \sigma^2 &= \operatorname*{avg}_i \{(\mathbf{y}_i^T \mathbf{e})^2\} \\ &= \mathbf{e}^T \operatorname*{avg}_i \{\mathbf{y}_i \mathbf{y}_i^T\} \mathbf{e}. \end{aligned} \tag{31}$$

Let matrix $\mathbf{M}$ be $\operatorname*{avg}_i \{\mathbf{y}_i \mathbf{y}_i^T\}$,

$$\sigma^2 = \mathbf{e}^T \mathbf{M} \mathbf{e}. \tag{32}$$

We find $\mathbf{e}$ that maximizes/minimizes the variance σ^2 under the condition of $\|\mathbf{e}\|=1$, we find such $\mathbf{e}$ using Lagrange's multiplier λ,

$$\frac{\partial}{\partial \mathbf{e}} \left[\mathbf{e}^T \mathbf{M} \mathbf{e} + \lambda \left(\mathbf{e}^T \mathbf{e} - 1 \right) \right] = 0, \tag{33}$$

or

$$\{\mathbf{M} - \lambda \mathbf{I}\} \mathbf{e} = 0. \tag{34}$$

Equation (34) is the same as an eigenvector problem of $\mathbf{M}$, where $\mathbf{e}$ is an eigenvector and λ is an eigenvalue. Let the eigenvalues of the $\mathbf{M}$ be λ_1 and λ_2 ($\lambda_1 > \lambda_2 > 0$) and the eigenvector according to each eigenvalue λ_1 and λ_2 $\mathbf{e}_1$ and $\mathbf{e}_2$ respectively. The $\mathbf{e}_1$ maximizes the variance, which is parallel to the long axis, and $\mathbf{e}_2$ minimizes the variance, which is parallel to the short axis.

Now we find the long and short radii from the eigenvalues λ_1 and λ_2. First we find the relationship between eigenvalues and radii. We assume that the shape of an object in a frame is an ellipse. Suppose the long axis of the ellipse is parallel to x-axis and its long and short radii are a and b, respectively. Let a point inside the ellipse be $\mathbf{z}=(x, y)^T$, the point satisfies,

$$\mathbf{z}^T \mathbf{P} \mathbf{z} \le 1, \tag{35}$$

where

$$\mathbf{P} = \begin{pmatrix} a^{-2} & 0 \\ 0 & b^{-2} \end{pmatrix}. \tag{36}$$

Let matrix $\mathbf{M}_\mathbf{z}$ be $\operatorname{avg}\{\mathbf{z}\mathbf{z}^T\}$, which is computed,

$$\mathbf{M}_\mathbf{z} = \operatorname{avg}\{\mathbf{z}\mathbf{z}^{\mathrm{T}}\} = \frac{\iint_{\mathbf{z}^T\mathbf{P}\mathbf{z} \le 1} \mathbf{z}\mathbf{z}^T d\mathbf{z}}{\iint_{\mathbf{z}^T\mathbf{P}\mathbf{z} \le 1} d\mathbf{z}}. \tag{37}$$

Let $\mathbf{z}$ express using the polar coordinate system r and θ as follows,

$$\mathbf{z} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix}. \tag{38}$$

Inequality (35) is same as,

$$0 \le r \le 1, \tag{39}$$

$\mathbf{z}\mathbf{z}^T$ is converted by (37),

$$\mathbf{z}\mathbf{z}^T = r^2 \begin{pmatrix} a^2\cos^2\theta & ab\cos\theta\sin\theta \\ ab\cos\theta\sin\theta & b^2\sin^2\theta \end{pmatrix}, \tag{40}$$

Since $\det\left[\frac{\partial(r,\theta)}{\partial \mathbf{z}}\right] = abr$, $abrd\theta dr = d\mathbf{z}$. So $\mathbf{M}_\mathbf{z}$ can be computed,

$$\mathbf{M}_\mathbf{z} = \frac{\int_0^1 \int_{-\pi}^{\pi} r^2 \begin{pmatrix} a^2\cos^2\theta & ab\cos\theta\sin\theta \\ ab\cos\theta\sin\theta & b^2\sin^2\theta \end{pmatrix} abrd\theta dr}{\int_0^1 \int_{-\pi}^{\pi} abrd\theta dr}, \tag{41}$$

or

$$\mathbf{M}_\mathbf{z} = \frac{1}{4} \begin{pmatrix} a^2 & 0 \\ 0 & b^2 \end{pmatrix}, \tag{42}$$

The eigenvalues λ_1 and λ_2 of $\mathbf{M}_\mathbf{z}$ be $a^2/4$ and $b^2/4$, respectively. Thus, the relationship between eigenvalues and the long and short radii of the ellipse is

$$\lambda_1 = \frac{a^2}{4} \text{ and } \lambda_2 = \frac{b^2}{4}, \tag{43}$$

or

$$a = 2\sqrt{\lambda_1} \text{ and } b = 2\sqrt{\lambda_2}. \tag{44}$$

To summarize, the direction of the ellipse axis is the eigenvector of $\operatorname{avg}\{\mathbf{y}_i\mathbf{y}_i^T\}$ and the long and short radii are twice the square root of the eigenvalues.