http://www.zju.edu.cn/jzus; E-mail: jzus@zju.edu.cn
ISSN 1009-3095 *Journal of Zhejiang University SCIENCE* V.5, No.1, P.99-105, Jan., 2004

A novel method for tracking pedestrians from real-time video*

HUANG Jian-qiang(黄建强)†, CHEN Xiang-xian(陈祥献), WANG Le-yu(汪乐宇)

(*Department of Instrumentation Science and Engineering, Zhejiang University, Hangzhou 310027, China*)

†E-mail: abraham_hjq@yahoo.com

Received Oct.24, 2002; revision accepted Jan.10, 2003

Abstract: This novel method of Pedestrian Tracking using Support Vector (PTSV) proposed for a video surveillance instrument combines the Support Vector Machine (SVM) classifier into an optic-flow based tracker. The traditional method using optical flow tracks objects by minimizing an intensity difference function between successive frames, while PTSV tracks objects by maximizing the SVM classification score. As the SVM classifier for object and non-object is pre-trained, there is need only to classify an image block as object or non-object without having to compare the pixel region of the tracked object in the previous frame. To account for large motions between successive frames we build pyramids from the support vectors and use a coarse-to-fine scan in the classification stage. To accelerate the training of SVM, a Sequential Minimal Optimization Method (SMO) is adopted. The results of using a kernel-PTSV for pedestrian tracking from real time video are shown at the end. Comparative experimental results showed that PTSV improves the reliability of tracking compared to that of traditional tracking method using optical flow.

Key words: Pedestrian tracking, Machine learning, Pyramid implementation, Virtual instrument
Document code: A **CLC number:** TH873.7

INTRODUCTION

Object tracking is a challenging and important task in computer vision. Some of the more popular applications include video surveillance instrument, robotic control, and autonomous vehicular navigation. Tracking algorithms find how an image region moves from one frame to the next. This implies the existence of an error function to be minimized, such as the sum of squared differences (SSD) between the two image regions. This error function is the result of making the "constant brightness assumption". Yet, quite often we are interested in tracking a particular class of objects such as pedestrians. In this case we can train a classifier in advance to distinguish between an object and the background. Then the question is how to combine the tracker and the classifier. Our solution is to replace the error function of the tracker. Instead of minimizing the SSD error, the tracker will try to maximize the classification score.

In the video surveillance instrument, we detect and track the pedestrian from a video sequence. The detection module of the instrument (which is not described in this paper) relies on a kernel-SVM that was trained on thousands of images of pedestrian and non-pedestrian. Once detected, the system tracks the pedestrian over time. Here we will use optic-flow (Bergen *et al.*, 1993) as the particular tracker. For classification we use the SVM (Osuna *et al.*, 1997; Morik *et al.*, 1999; Xin *et al.*, 2002) technique.

The problem tackled by the PTSV is often referred to as transformation invariance in machine learning. Simard *et al.* (1993) used the nearest-neighbor classifier with a "tangent distance" metric and applied it to character recognition. Later, Vasconcelos and Lippman (1998) worked on face images by extending "tangent distance" to work in a multi-resolution framework. Final-

* Project supported by Japanese Monbusho Scholarship Program and Zhejiang Provincial Scientific Research Foundation for Return Overseas Chinese Scholars(No.2004-4)

ly, Scholkopf *et al*. (1998) introduced "tangent distance" to SVM by deriving a kernel that enforces local transformation invariance. Good solution to transformation invariance can improve the robustness of a virtual instrument system (Zhou and Wang, 2001).

Our approach is similar to that of Black and Jepson (1998) in that we find the transformation parameters as part of the classification stage. We extend their work by using SVM instead of eigenvectors. This does not complicate the learning phase to achieve transformation invariance and falls naturally within a multi-resolution manner.

SUPPORT VECTOR MACHINE

Consider a set of data, $\{\boldsymbol{X}_i, y_i\}$, such that $\boldsymbol{X}_i$ is an input and $y_i \in \{-1, +1\}$is a target output. A support vector machine is a model that is calculated as a weighted sum of kernel function outputs. The kernel function of an SVM is written as $k(\boldsymbol{X}_a, \boldsymbol{X}_b)$ and it can be an inner product, Gaussian, polynomial, or any other function that obeys Mercer's condition (Gunn, 1998).

Formally, we consider the family of decision functions

$$f(\boldsymbol{X}) = \operatorname{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{X} + b) \qquad (1)$$

and wish to minimize $\| \boldsymbol{w} \|$ such that $\operatorname{sgn}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{X} + b) = \operatorname{sgn}(y_i)$. The solution for $\boldsymbol{w}$ is found by solving the quadratic programming problem defined by the objective function and the constraints.

In the simplest case, where $k(\boldsymbol{X}_a, \boldsymbol{X}_b) = \boldsymbol{X}_a \cdot \boldsymbol{X}_b$ and the training data are linearly separable, computing an SVM for the data corresponds to minimizing $\| \boldsymbol{w} \|$ such that $y_i(W \cdot \boldsymbol{X}_i - W_0) - 1 \geqslant 0$, $\forall\, i$, whose solution is:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{X}_i$$

with constraint

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

In the nonlinear case, the decision function we deal with becomes:

$$f(\boldsymbol{X}) = \operatorname{sgn}\left(\sum_{j=1}^{l} y_j \alpha_j k(\boldsymbol{X}, \boldsymbol{X}_j) + b\right) \qquad (2)$$

The objective function (which should be minimized) for Eq.(2) is:

$$W(\alpha) = \sum_{i=1}^{l} \alpha_j - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(\boldsymbol{X}_i, \boldsymbol{X}_j) \qquad (3)$$

with constraints

$$\alpha_i \geqslant 0,\ i = 1, \cdots, l,\ \sum_{i=1}^{l} \alpha_i y_i = 0$$

Eq.(3) will always have a single minimum with respect to the Lagrange multipliers α_i. The minimum to Eq.(3) can be found with any of a family of algorithms, all of which are based on constrained quadratic programming. We used a faster Sequential Minimal Optimization (SMO) algorithm (Platt, 1999) in all of our experiments to speed up the training process.

In our cases, we used a Gaussian kernel function: $k(\boldsymbol{X}, \boldsymbol{X}_j) = \exp(-\| \boldsymbol{X} - \boldsymbol{X}_j \|^2 / 2\sigma^2)$. The choice of σ is usually made to reflect the smoothness of the feature space and the density of the training data, as there is no general choice of σ, we heuristically set σ to 16.

PEDESTRIAN TRACKING USING SUPPORT VECTOR

Recently there is some attention paid to multiple cameras tracking (Khan *et al*., 2001) for extending the field of view. In our pedestrian tracking system, the camera can turn around by the mechanical and electronic instrument. The camera is mounted on a high tower and will turn towards the tracked object. The framework in which PTSV will work is as follows. The detection module will detect possible candidates in the current frame and hand them over to the PTSV. The PTSV will refine their position so that a local maximum of SVM score is achieved. If the score is positive the candidate will be declared a pedestrian and a tracking process will start. The refined position in the current frame will serve as the initial guess in the next frame and so on.

We introduce the notations and develop PTSV for the simple case of 2D translation model. Then we extend PTSV to work on pyramids by introducing the Support Vector Pyramid.

1. Pedestrian tracking using support vector

Kernel-SVM is given by

$$\sum_{j=1}^{l} y_j \alpha_j k(\boldsymbol{I}, \boldsymbol{X}_j) + b \tag{4}$$

where $\boldsymbol{X}_j$ are the support vectors, y_i are their sign and α_j are their distance from the hyperplane. $k(\boldsymbol{I}, \boldsymbol{X}_j)$ is the kernel we choose to use, and $\boldsymbol{I}$ is the image region we wish to test.

Let $\boldsymbol{I}_{\text{init}}$ represent the initial guess of the object's position in a given image. Furthermore, let us assume that the initial guess is not too distant from the correct position ($\boldsymbol{I}_{\text{final}}$) of the object. Using first-order Taylor expansion we have

$$\boldsymbol{I}_{\text{final}} = \boldsymbol{I}_{\text{init}} + u\boldsymbol{I}_x + v\boldsymbol{I}_y \tag{5}$$

where $\boldsymbol{I}_x$, $\boldsymbol{I}_y$ are the x and y derivatives of (sub-)image and u, v are the motion parameters. By assumption we have that the SVM score of $\boldsymbol{I}_{\text{final}}$ is a local maximum. Put formally

$$\sum_{j=1}^{l} y_j \alpha_j k(\boldsymbol{I}_{\text{final}}, \boldsymbol{X}_j) = \max\left(\boldsymbol{I} \,\middle|\, \sum_{j=1}^{l} y_j \alpha_j k(\boldsymbol{I}, \boldsymbol{X}_j) \right) \tag{6}$$

where $\boldsymbol{I}$ are all possible (sub-)images (in vector form) in the neighborhood of (sub-)image $\boldsymbol{I}_{\text{final}}$ (we drop the constant b as it does not affect the solution). Putting Eq.(5) in Eq.(4), we get:

$$\sum_{j=1}^{l} y_j \alpha_j k(\boldsymbol{I} + u\boldsymbol{I}_x + v\boldsymbol{I}_y, \boldsymbol{X}_j) \tag{7}$$

Eq.(7) is the one that we want to maximize. For readability we will denote $\boldsymbol{I}_{\text{init}}$ as $\boldsymbol{I}$.

We used a Radial Basis Function given by kernel $k(\boldsymbol{X}, \boldsymbol{X}_j) = \exp(-\| \boldsymbol{X} - \boldsymbol{X}_j \|^2 / 2\sigma^2)$. A quadratic polynomial function kernel can also be used in the system. Take the u, v derivatives of Eq.(7) and let:

$$\begin{cases} \partial \boldsymbol{E} / \partial u = 0 \\ \partial \boldsymbol{E} / \partial v = 0 \end{cases} \tag{8}$$

After rearranging terms we have the following equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_x^2 & \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_x \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_y \\ \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_x \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_y & \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_x \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I} \\ \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I}_y \boldsymbol{X}_j^{\mathrm{T}} \boldsymbol{I} \end{bmatrix} \tag{9}$$

This equation resembles the standard optic-flow equations with the support vectors replacing the role of the second image in the equation. This means that all computations are done on a single frame each time and not on a pair of successive frames. Also, there is one major difference. In PTSV the image region to be tracked must be rescaled to the size of the support vectors, then we perform a number of PTSV iterations, using Eq.(9) to maximize the SVM score. This approach can handle small motions in the image plane. Larger motions must be handled in a coarse-to-fine manner, as described in the next subsection.

2. Pyramid implementation of PTSV

The image that needs to be classified is first sub-sampled to the size of the support vectors and then its SVM score is computed. Thus, PTSV works on the sub-sampled version of the test image. The misalignments must be small so that the first-order Taylor approximation, used by PTSV, will suffice to lock on the best position. However, if the motions are large, we can no longer hope for the approximation to work and pyramids must be used.

There are many kinds of image representation (Wang *et al.*, 2001; Liu and Huang, 2001; Hui *et al.*, 2002). Pyramid implementation is one of the efficient methods for representing an image. Because the support vectors are a sub-sampled image of the most problematic images in the learning set, we can smooth and subsample them to create a support vector pyramid for each support vector. In the classification stage we create a pyramid of the same size as the support vector pyramids from the test image. Then we run PTSV on successive levels of the pyramids in a coarse to fine manner.

EXPERIMENTS

The classification engine was trained on a set of approximately 2000 images of pedestrians and non-pedestrians. The images were digitized from a progressive scan video at a resolution of 320 × 240 pixels and 10 frames per second. Typical pedestrian size is about 60 × 20 pixels. The pedestrians and non-pedestrians were manually selected and reduced to the size of 32 × 12 pixels. Their mean intensity value was shifted to the value 0.5 (in the range [0...1]). We have even used Gaussian Radial Basis Function kernel to perform the learning phase. The classification rate was about 96% for the learning set, with about 1500 support vectors.

For speeding up the classification phase we used the Reduced Set Method (Burges, 1996) to reduce the number of support vectors from 1500 to 300. The Reduced Set Method shows that the number of support vectors can be reduced, through Principal Component Analysis on the support vectors in feature space. In practice we found that the 60 support vectors with the largest eigenvalues are sufficient for classification. For each of the 60 support vectors we created a 2 level Gaussian pyramid by performing a 2D smoothing followed by sub-sampling. Each test image was sub-sampled to a Gaussian pyramid with the bottom level of the pyramid being 32×12 pixels. We used PTSV with a 2D translation model on the pyramid to refine the image position. The score of the test image was taken to be the SVM score of the bottom level. PTSV takes as input an initial position of the frame. It then applies pyramid PTSV and outputs the position in the image, with the highest SVM score. This position then serves as the initial guess for the next frame.

Here we present results of four tests on a wide variety of pedestrians. No parameter was changed from test to test. In all cases the initial guess in the first image was supplied manually. In the real system the detection module will supply the initial guess.

In the first test we supplied the algorithm with a rough initial guess and tested how well it maximized the SVM score (Fig. 1). For each image in the figure we show how much did the SVM score improved and what was the motion from the initial guess to the final position. We found that translations up to about 10% of the pedestrian size were handled by the algorithm.

Fig.1 Examples of the initial guess (dashed line) and the final position (solid line). The image size in all cases is 320 × 240 (The SVM score of the initial guess and the final position as well as the amount of motion between the initial and final position are shown. The best position in the coarser level serves as a good strarting point for the next level)

(a) Init: −2.3; Final:2.9; Motion (−1.8, −4.6); (b) Init: −3.5; Final:0.7; Motion(2.9,6.2)

The second test shows how the 2D translation motion model we use handles an approaching pedestrian. In this sequence of 301 frames the pedestrian is approaching the camera (Fig. 2). Note also that the viewpoint changes but still PTSV manages to keep track of the rear of the pedestrian with high SVM scores.

In the third test (Fig. 3) we compared the PTSV against a simple SSD tracker. The SSD tracker works by minimizing the SSD error between successive frames. It uses pyramids to account for large motions and uses the following optic flow equation to estimate the motion parameters:

$$\begin{pmatrix} \sum \boldsymbol{I}_x^{\mathrm{T}}\boldsymbol{I}_x & \sum \boldsymbol{I}_x^{\mathrm{T}}\boldsymbol{I}_y \\ \sum \boldsymbol{I}_x^{\mathrm{T}}\boldsymbol{I}_y & \sum \boldsymbol{I}_y^{\mathrm{T}}\boldsymbol{I}_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -\sum \boldsymbol{I}_t^{\mathrm{T}}\boldsymbol{I}_x \\ -\sum \boldsymbol{I}_t^{\mathrm{T}}\boldsymbol{I}_y \end{pmatrix} \tag{10}$$

Where $\boldsymbol{I}_x$, $\boldsymbol{I}_y$ are the x, y derivatives of the first image and $\boldsymbol{I}_t$ is the time derivative between the two frames. For benchmarking, the SVM score of pedestrian's region detected by SSD is calculated.

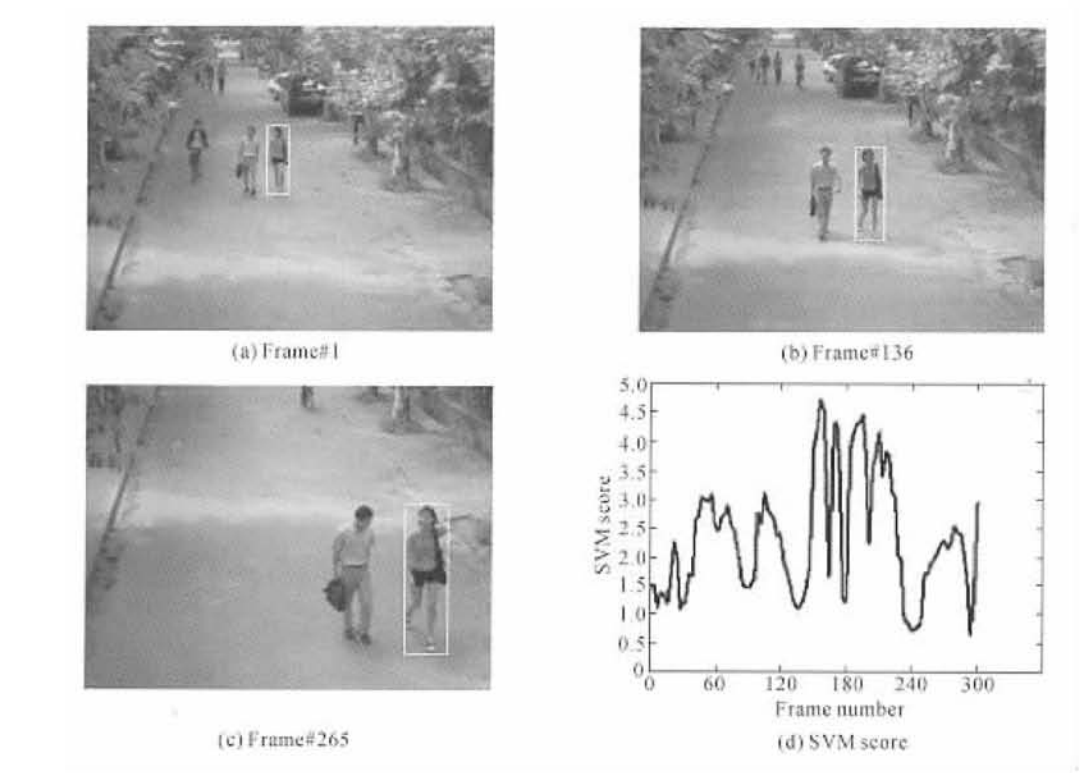

Fig.2 SVT tracking (a),(b),(c) are three frames from a 301 frames sequence; (d) shows the SVM score of the tracked region using the SVT tracker

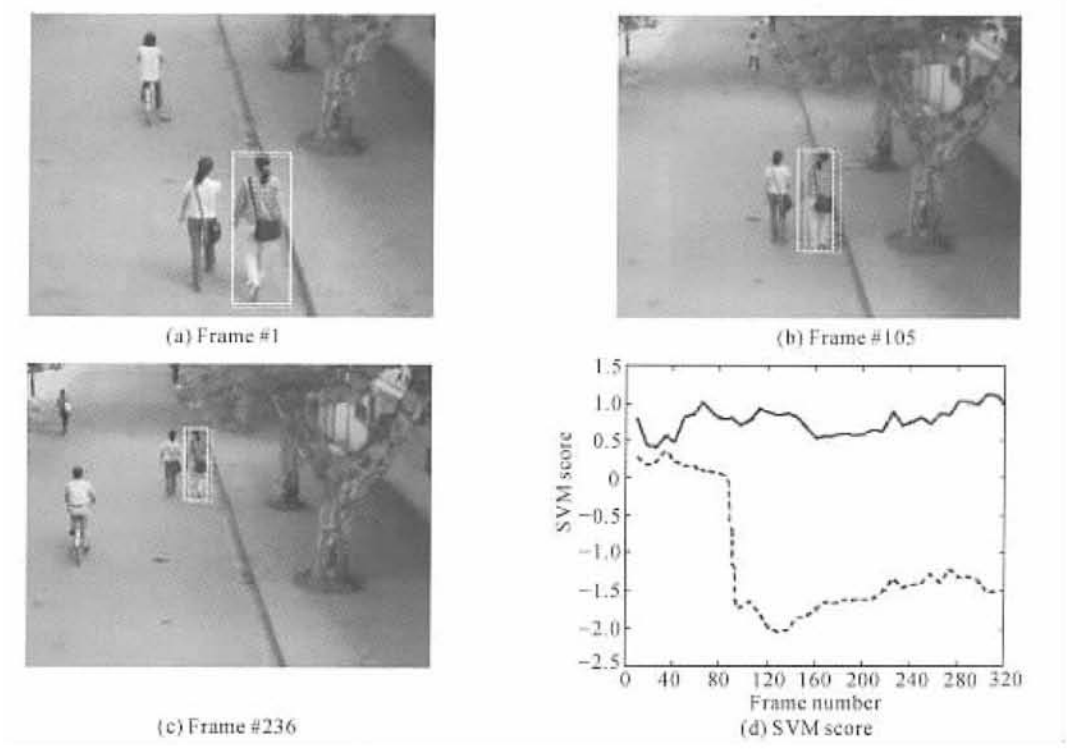

Fig.3 (a)(b)(c) are three frames from a 320 frames sequence(The solid rectangle denotes the SVT tracking result; the dashed rectangle denotes a simple SSD tracker); (d) shows the SVM score of the tracked region using the SVT tracker (solid line) and SSD tracker (dashed line). As can be seen, SVT is far superior to the SSD tracker

As we can see, up to frame 80 both trackers perform comparably. However at frame 80, the tracking diverges by six pixels. This is enough to drastically lower the SVM score obtained by the SSD tracker. As can be seen it never recovers from this miss and it keeps drifting away. The PTSV tracker on the other hand keeps getting high SVM scores throughout the sequence.

The last test shows a challenging case of partially occluded pedestrian (Fig.4), the two pedestrians are very close to each other and illumination also changes because of the sunlight through the trees. The SSD tracker fails to track the pedestrian and drift away after 300 frames without getting positive SVM score along the way. The PTSV on the other hand remains attached to the pedestrian while maintaining a positive SVM score.

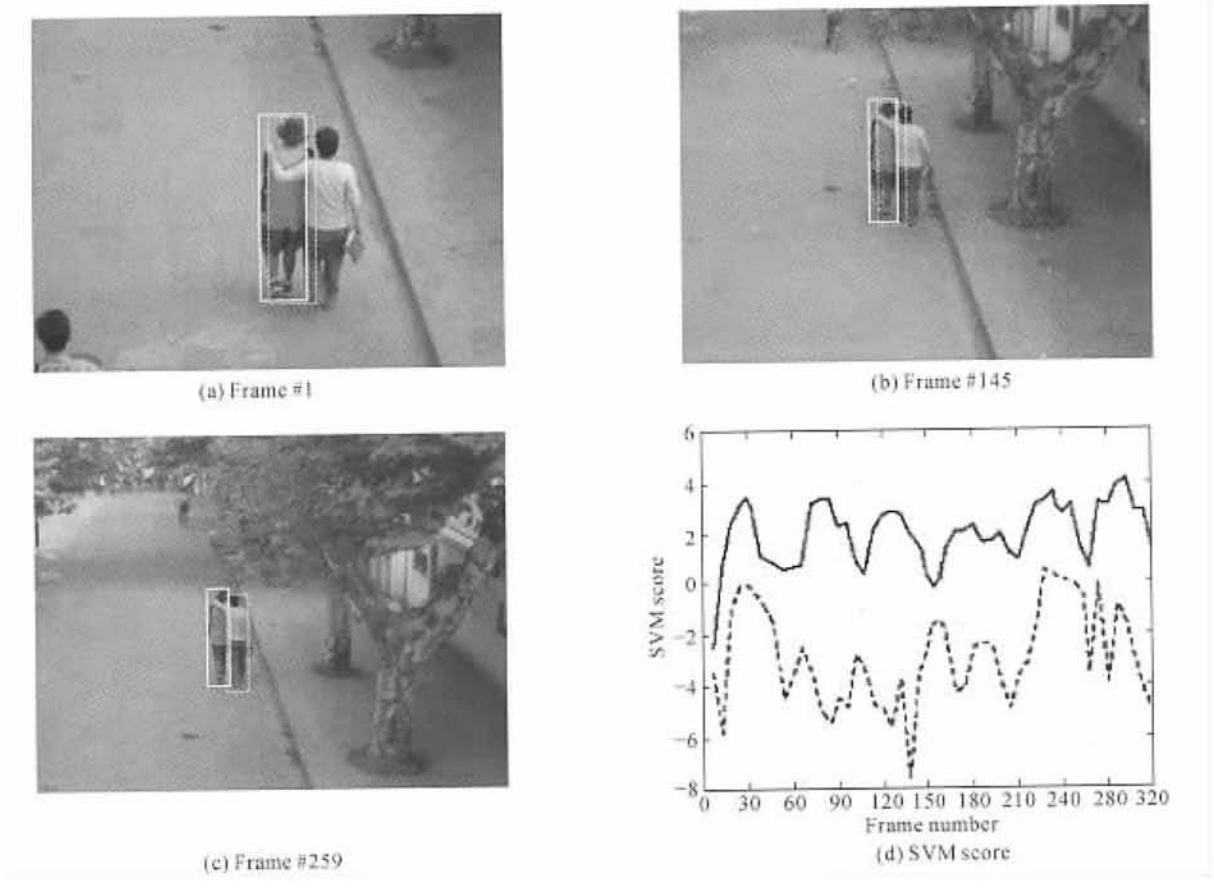

Fig.4 (a), (b), (c) are three frames from a 300 frames sequence (The solid rectangle denotes the SVT tracking result; the dashed rectangle denotes a simple SSD tracker); (d) show the SVM score of the tracked region using the SVT tracker (solid line) and SSD tracker (dashed line). As can be seen, the SSD tracker is fooled while the SVT tracker is much more stable

CONCLUSIONS

Here we combine the SVM classifier and the optic-flow tracker to achieve a Pedestrian Tracking using Support Vector (PTSV) mechanism. While this might slow down the computations required for real-time tracking it offers an effective method to align a test image with the SVM support vectors, thus PTSV can be regarded as extending SVM to have some inherent invariance to image transformations. PTSV works by maximizing the SVM classification score, instead of minimizing an intensity difference function between successive frames. What is more, we created a Gaussian pyramid from every support vector, terming it "Support Vector Pyramid", that allows PTSV to handle large motions in the image plane. The algorithm was tested on real video sequences for the purpose of pedestrian tracking. In the video surveillance instrument, we have

used PTSV with a Gaussian Radial Basis Function kernel. However, the PTSV paradigm can be used with various motion models up to a 2D affine transformation and various kernels such as B splines, Multi-Layer Perceptron, Fourier Series and so on. Future work can use other kernels to improve the tracking robustness of the video-tracking instrument.

References

Bergen, J.R., Anandan, P., Hanna, K., J. and Hingorani, R., 1993. Hierarchical Model-Based Motion Estimation. *In*: Motion Analysis and Image Sequence Processing. Sezan, M.I. and Lagendijk, R.L. (eds), Kluwer Academic Press, Dordrecht, Netherlands, p. 257 – 232.

Burges, C., 1996. Simplified Support Vector Decision Rules. Proceedings of the 13th International Conference on Machine Learning, San Mateo, Canada, p.71 – 77.

Black, M. J. and Jepson, A., 1998. EigenTracking: robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, **26**(1): 63 – 84.

Gunn, S., 1998. Support Vector Machines for Classification and Regression. ISIS Technical Report ISIS-1-98, Image Speech Intelligent System Research Group, University of Southampton, England.

Hui, H., Zhou, H. and Wang, L.Y., 2002. Optimal Gabor Filters Design for Fingerprint Recognition. Proceedings of SPIE, Annual Meeting 2002, Seattle, Washington, USA, **4790 – 85**: 351 – 356.

Khan, S., Javed, O., Rasheed, Z. and Shah, M., 2001. Human Tracking in Multiple Cameras. The Eighth IEEE International Conference on Computer Vision, Vancouver, Canada, p.331 – 336.

Liu, J.F. and Huang, D.R., 2001. Zerotrees and pyramidal lattice vector quantization for wavelet image coding. *Journal of Image and Graphic*, **6**(A): 229 – 232.

Morik, K., Brockhausen, P. and Joachims, T., 1999. Combining Statistical Learning with a Knowledge-based Approach - A Case Study in Intensive Care Monitoring, Proc. 16th International Conf. on Machine Learning, Morgan Kaufman Publishers, San Mateo, Canada, p. 268 – 277.

Osuna, E., Freund, R. and Girosi, F., 1997. Training Support Vector Machines: An Application to Face Detection. Proc. of IEEE Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, p. 130 – 136.

Platt, J., 1999. Using Sparseness and Analytic QP to Speed Training of Support Sector Machines. *In*: Advances in Neural Information Processing System, M. S. Kearns, S. A. Solla, D. A. Cohn (eds), MIT Press, USA, **11**: 126 – 134.

Simard, P., LeCun, Y. and Denker, J., 1993. Efficient Pattern Recognition using a New Transformation Distance. *In*: Advances in Neural Information Processing System, Lippmann, P.L., Moody, J.E., Touretzky, D.S (eds), Morgan Kaufman Publishers, San Mateo, CA, p. 50 – 58.

Scholkopf, B., Simard, P., Smola, A. and Vapnik, V., 1998. Prior Knowledge in Support Vector Kernels. *In*: Advances in Neural Information Processing Systems, M. I. Jordan, M. J. Kearns, S. A. Solla (eds), MIT Press, USA, **10**: 640 – 646.

Vasconcelos, N. and Lippman, A., 1998. Multiresolution Tangent Distance for Affine-invariant Classification. *In*: Advances in Neural Information Processing Systems, M. I. Jordan, M. J. Kearns, S. A. Solla (eds), MIT Press, USA, **10**: 843 – 849.

Wang, Z.Y., Chi, Z.R., Deng, D. and Cho, S.Y., 2001. Adaptive Processing of Tree-Structure Image Representation. IEEE Pacific Rim Conference on Multimedia, Beijing, China, p. 989 – 995.

Xin, D., Wu, Z.H. and Pan, Y.H., 2002. Probability output of multi-class support vector machines. *Journal of Zhejiang University SCIENCE*, **3**(1): 131 – 134.

Zhou, H. and Wang, L.Y., 2001. Virtual instrument system software architecture description language. *Journal of Zhejiang University SCIENCE*, **2**(4): 411 – 415.