Multi-Object Tracking Using Dynamical Graph Matching

Hwann-Tzong Chen Horng-Horng Lin Tyng-Luh Liu
Institute of Information Science, Academia Sinica
Nankang, Taipei 115 Taiwan
{pras, hhlin, liutyng}@iis.sinica.edu.tw

Abstract

We describe a tracking algorithm to address the interactions among objects, and to track them individually and confidently via a static camera. It is achieved by constructing an invariant bipartite graph to model the dynamics of the tracking process, of which the nodes are classified into objects and profiles. The best match of the graph corresponds to an optimal assignment for resolving the identities of the detected objects. Since objects may enter/exit the scene indefinitely, or when interactions occur/conclude they could form/leave a group, the number of nodes in the graph changes dynamically. Therefore it is critical to maintain an invariant property to assure that the numbers of nodes of both types are kept the same so that the matching problem is manageable. In addition, several important issues are also discussed, including reducing the effect of shadows, extracting objects' shapes, and adapting large abrupt changes in the scene background. Finally, experimental results are provided to illustrate the efficiency of our approach.

1. Introduction

There has been considerable work on visual tracking for a variety of applications. We will concentrate mostly on real-time tracking systems/algorithms.

The CONDENSATION introduced by Isard and Blake is to track curves even in clutter background [7], [8]. They modeled objects as a set of parameterized curves in terms of B-splines, then used factored sampling to predict the positions of curves during tracking. The algorithm is superior to previous Kalman filter-based approaches. More recently, Toyama and Blake have established a probabilistic exemplar-based framework, the *Metric Mixture* model, to combine the exemplars in a metric space with a probabilistic treatment for visual tracking [15].

Paragios and Deriche [12], [13] addressed the problem of simultaneously tracking several non-rigid objects and estimating their motion parameters using a coupled front propagation model, of which it integrates boundary and region-based information. Their implementation for solving the PDEs used a level set approach to deal with topological changes of the moving front. In [9], Isard and MacCormick adopted multi-blob likelihoods as the observation models for both background and foreground, and described a Bayesian tracker via particle filtering to track multiple objects efficiently.

Another vein of approaches in visual tracking is based on *frame differencing* and *shape analysis*. Pfinder [17] is a real-time system to perform person segmentation, tracking and interpretation. To find and follow the head and body parts of a person, the system can build up a blob model dynamically using a multi-class statistical model of color and shape. Haritaoglu *et al.* [6] proposed the W^4 system that combines shape analysis and statistical techniques to track people and their part structures in an outdoor environment. To handle interactions among the tracked people, they used *appearance models* to resolve the ambiguities. The *Backpack* system [5] was designed to work under the control of W^4 silhouette model. The basic steps of Backpack are histogram projection, shape periodicity analysis and symmetry analysis. A non-symmetric region which has insignificant periodicity is classified as an object carried by a person.

Unlike most background extraction tracking algorithms, Lipton [10] combined temporal differencing and template correlation matching to perform target tracking. In [11] a system for color image sequences was presented. The approach is similar to W^4 built upon a background model combining pixel RGB and chromaticity values with local image gradients.

The *mean shift* was used by Comaniciu *et al.* to track objects by modeling them as probability distributions [3]. It does not require a static camera, and can track objects, even with partial occlusions. In [2], Chen and Liu have proposed a new tracking algorithm based on *trust-region* methods. They showed that a trust-region tracker should perform better than a *line-search* tracker. In particular, tracking with mean shift is a typical line-search one since the iterative optimization process is driven by mean shift vectors, i.e., the iterates are restricted to the approximated gra-

dient directions. Another tracking method based on the *co-inference* between shape and color models has been recently presented by Wu and Huang [18]. The tracking system was implemented with a sequential Monte Carlo technique to approximate the co-inference process between the models.

1.1. Our Approach

A variety of issues must be investigated when designing a reliable real-time multi-object tracking system. We concentrate on resolving the ambiguities caused by the interactions among the objects. More specifically, our approach contributes to this field of research by addressing the following three problems.

- *Shape contour extraction and shadow deletion:* Shadows caused by indoor lighting and interactions between objects should be detected and removed so that they will not interfere with the performance of the tracking system. We have used a two-pass shadow deletion algorithm to reduce the effects of shadows. Furthermore, a contour extraction scheme motivated by the level set method is developed to derive the shape/silhouette of each target object.

- *Dynamical bipartite graph and best assignment:* During tracking, each target object is represented with its color distribution. To simulate the process of multi-object tracking and account for interactions, an *invariant bipartite graph* is constructed. The invariant property makes sure that the two classes of nodes in the graph will have the same number, i.e., the number of target objects currently in the scene. Since our goal is to track objects without mixing up their identities, a multiple mode detection method via kernel analysis is used to segment the foreground pixels in an interaction area such that each object can be tracked individually. Once the bipartite graph is available, it is convenient to find the optimal matching, of which it corresponds to the best identity assignment to all detected objects.

- *Scene background change and automatic adaption:* There are two types of background change to be discussed. Illumination change during tracking is of the first type concerned us most since this happens gradually and persistently. We use a scheme combining short-time updating and iterative training to guarantee that at any moment of a tracking process, the statistical quantities used for the reference background are mostly derived from real data rather than by approximation. The other type of background change is more drastic, e.g. an object that is initially part of the reference background later becomes active, then starts to move and interacts with other objects in the scene. How to detect and handle such events is quite difficult, and it will be explained in detail later.

2. Foreground Separation and Contour Extraction

We assume a stationary background scene where the interactions of multiple objects occur. When the system starts to perform tracking, the first few image frames will be used to compute some statistical quantities about the scene. By a reference background, we mean the background scene and the derived statistical quantities. To detect moving objects in an image frame, the algorithm uses foreground/background extraction, shadow deletion and shape contour extraction to separate foreground objects from the background scene.

2.1. Foreground/Background Separation

The initial background training is carried out, say over the first N image frames, where we assume that there is no object undertaking large/significant movements in the scene. Then for each pixel $p_{j,k}$, its intensity mean $\mu_{j,k}$ and unbiased sample variance $\sigma^2_{j,k}$ can be computed using the following iterative formula. For image frame $f = 2, \ldots, N$, we have

$$\begin{aligned} [\mu_{j,k}]_f &= [\mu_{j,k}]_{f-1} + \frac{1}{f}([I_{j,k}]_f - [\mu_{j,k}]_{f-1}), \\ [\sigma^2_{j,k}]_f &= \frac{f-2}{f-1}[\sigma^2_{j,k}]_{f-1} + \frac{1}{f}([\mu_{j,k}]_{f-1} - [I_{j,k}]_f)^2, \end{aligned} \tag{1}$$

where $[\cdot]_f$ denotes the corresponding value at frame f, and $[\mu_{j,k}]_1 = [I_{j,k}]_1$ and $[\sigma^2_{j,k}]_1 = 0$. Equations (1) can be shown by straightforward calculations that they yield the sample mean and unbiased sample variance for the first N image frames. Right after the training stage, the system is ready to perform real-time tracking. In each new image frame, the foreground pixels can be obtained by comparing their intensity values to the corresponding mean values. A pixel $p_{j,k}$ is extracted as a foreground pixel if its intensity $I_{j,k}$ satisfies $|I_{j,k} - \mu_{j,k}| > \alpha \cdot \sigma_{j,k}$ where the parameter α can be adjusted to yield more or less foreground pixels.

In general, the extracted foreground pixels are raw and sensitive to noise. Thus several low-level image processing techniques are used to refine the foreground. To efficiently manage the low-level image processing, we create a support map to represent the foreground region. A support map is a binary image where pixel values are set to 1 if they belong to foreground or set to 0 otherwise. Firstly, one iteration of erosion is applied to support map to eliminate single-pixel noise. Secondly, the support map is divided into 8×8

blocks. A block is marked as valid block if it contains more than 20 foreground pixels. Thirdly, connected components of valid blocks are constructed to locate the corresponding minimal bounding box enclosing each's foreground pixels.

After the low-level image processing, several steps of high-level grouping are performed to obtain the final shape contours and bounding boxes. We first eliminate bounding boxes which consist of less than 5 blocks since we don't want to track small objects. Bounding boxes with small width/height ratios or at inadequate locations are ignored. The rest of bounding boxes are verified to see if they can be grouped into larger ones. Such grouping step is necessary for handling partial occlusions or undetected foreground due to similar color to the background. Once the bounding boxes are determined, the foreground pixels inside each bounding box can be further refined by one iteration of dilation.

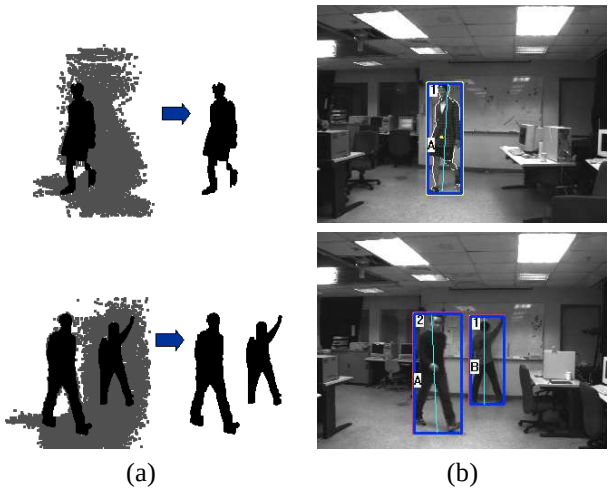
As mentioned before, it is preferable that the system can adapt to the illumination change automatically. Thus, during tracking, we periodically update the background statistics for pixels outside the bounding boxes (those are background pixels). Let $[\mu_{j,k}]_{old}$ and $[\sigma^2_{j,k}]_{old}$, respectively, be the old intensity mean and variance before updating has occurred at pixel $p_{j,k}$. Then the update rules for pixel $p_{j,k}$ at image frame f are

$$
\begin{aligned}
[\mu_{j,k}]_f &= [\mu_{j,k}]_{f-1} + \frac{1}{N}([I_{j,k}]_f - [\mu_{j,k}]_{old}) \\
[\sigma^2_{j,k}]_f &= [\sigma^2_{j,k}]_{f-1} + \frac{1}{N-1}([I_{j,k}]^2_f - [\mu_{j,k}]^2_{old}) \\
&\quad - \frac{N}{N-1}([\mu_{j,k}]^2_f - [\mu_{j,k}]^2_{f-1}) - \frac{1}{N}[\sigma^2_{j,k}]_{old}\,.
\end{aligned}
$$

With the above formulae, if some pixel has been updated N times, not necessary over consecutive frames, then the newly computed mean and variance will be the exact ones rather than by approximation. Again, this can be easily verified by straightforward calculation. Thus, if some pixel's referenced mean and variance have been updated N times, they will be replaced by the newly computed ones. In this way, the related statistical quantities of the reference background will be more accurate.

2.2. Shadow Deletion

Lighting in an indoor environment can cause serious problem of shadows (e.g. Figure 1(a)). Typically the intensity changes within shadow areas are significant enough to make some pixels to be incorrectly detected as foreground ones. Thus a bounding box will be enlarged by shadows since it would contain more foreground pixels. Such phenomenon is rather undesirable especially when most statistical features are determined by the foreground pixels. Nevertheless, the problem of shadows is common to all background-extraction based tracking algorithms.



(a) (b)

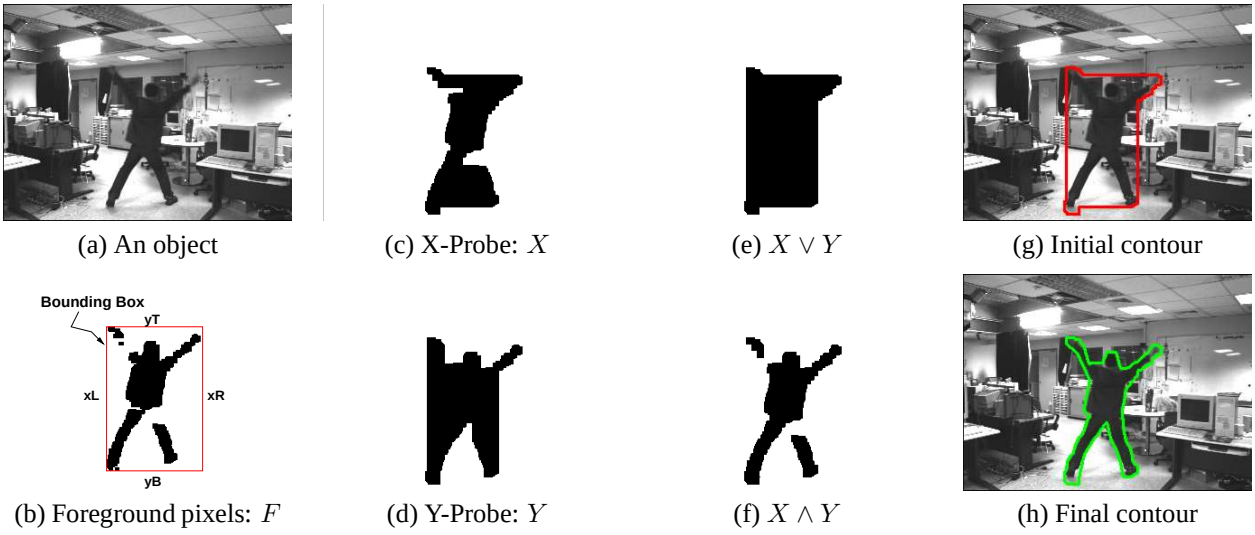
Figure 1. (a) Shadows caused by lighting and moving object interactions in an indoor environment. (b) Foreground object segmentations derived after shadow deletions.

McKenna et al. have used chromatic information to detect and eliminate shadows [11]. It is also possible to use more than one camera and depth information to identify the plane of shadow as described in [6]. More complicated is to remove shadows in a grayscale image without using any information from stereo. Our approach follows the ideas proposed in [14], i.e., to model the shadows as regions of constant contrast change. This gives us two heuristics to detect shadows.

- The pixel intensity values within shadow regions decrease in most cases, when compared to the means computed in the reference background.
- The intensity reduction rate changes smoothly between neighboring pixels, i.e., the photometric gain does not vary much in a shadow region. Furthermore, it is also true most shadow regions do not have strong edges.

The two criteria will fail and mistakenly remove correct foreground pixels when pixels of foreground objects are darker than the background and have a uniform gain with respect to the surface they occlude. However, it only occurs occasionally and should be considered as an exception.

We use a two-pass shadow deletion algorithm, where the foreground pixels are scanned *horizontally* and *vertically*, respectively. Such scheme makes sure most of the shadow pixels will be investigated and deleted. Therefore to determine that a pixel is within a shadow region or not, we check



(a) An object (c) X-Probe: X (e) $X \vee Y$ (g) Initial contour

(b) Foreground pixels: F (d) Y-Probe: Y (f) $X \wedge Y$ (h) Final contour

Figure 2. Shape contour extraction for a disconnected set of foreground pixels.

whether the intensity reduction rates decrease smoothly in a small neighborhood. To improve the accuracy and robustness, information of neighboring pixels are considered to support the decision. After eliminating shadow pixels from foreground pixels, each bounding box needs to be adjusted to fit the corrected foreground data.

2.3. Shape Contour Extraction

Once the foreground pixels are identified, we are ready to extract the shape contours. Foreground pixels enclosed by a bounding box are considered to belong to a same object. However, they may not form a connected component as it could happen that some part of the object may have color distribution similar to the background (e.g., see Figure 2 (a),(b)). To overcome such circumstance, we design a fast contour extraction algorithm motivated by the basic idea of level set methods. Recall that the level set methods are designed to solve the initial value problems, where in our case we focus on the 2D closed curve evolution problem. More precisely, the level set equation is

$$\phi_t + V|\nabla\phi| = 0 \quad \text{given} \quad \phi(\mathbf{x}, t=0)\,, \tag{2}$$

where V is the speed, and ϕ is the level set function and its zero level set at any time t, $\{\mathbf{x} \mid \phi(\mathbf{x}, t) = 0, \mathbf{x} \in R^2\}$ is the curve's locus at time t. In practice, for real-time multi-object tracking application, it is not feasible to solve the above equation accurately. Instead, we try to come out with a good approximating solution for (2) in one time step. To achieve such effectiveness, one needs to first have a good guess for the initial curve $\phi(\mathbf{x}, 0) = 0$, then a reasonable mechanism to assign the speed V for points on the initial curve. Nevertheless, it turns out that the two issues can be simplified a lot as we are solving a level-set evolution problem over a binary image. The details of contour extraction scheme are summarized as follows.

1. *Finding the initial contour:* To find the initial contour inside a bounding box, two probes are carried out to estimate the object's shape. The first one is performed by horizontal scan. We first create a binary image, called X, with the same dimension as the bounding box's and set all its pixel values to 0. The probe is executed row by row, starting from the first one. Each row is scanned simultaneously using two pointers at the two ends xL and xR (see Figure 2(b)). Each pointer will continue to move toward the center until it reaches a foreground pixel or the two pointers meet each other. When a row scan is completed, we check if there are any pixels in between the two pointers (including the two pointers), then set the pixels at the corresponding positions in X to 1. When the row probe is done, a dilation operation is applied to X for robustness. The column probe can be performed in a similar manner, and the resulting shape will be named as Y. Finally, we set the initial contour to be the boundary of the union of two probe sets, i.e.,

$$\{\mathbf{x} \mid \phi(\mathbf{x}, 0) = 0\} = \partial(X \vee Y)\,, \tag{3}$$

where $X \vee Y = \{\mathbf{x} \mid X(\mathbf{x}) = 1 \text{ or } Y(\mathbf{x}) = 1\}$. Analogously, we define $X \wedge Y = \{\mathbf{x} \mid X(\mathbf{x}) = 1 \text{ and } Y(\mathbf{x}) = 1\}$ (see Figure 2(e),(f)). Notice that an initial contour yielded according to (3) will always include the object's silhouette.

2. *Estimating the speed function:* Since the curve evolution is assumed to be completed in one time step, we only need to figure out the speed for points on the initial contour. Following a counter-clockwise order, for each $\mathbf{x}$ on the contour, its speed $V(\mathbf{x})$ will be set to 0, if $F(\mathbf{x}) = 1$, that is, $\mathbf{x}$ is a foreground pixel (see Figure 2(b)). The other case is more complicated that one needs to check whether $\mathbf{x}$ is on a vertical edge or a horizontal one. Suppose $\mathbf{x} = (x, y)^t$ is on a vertical edge. Then its speed will be defined as

$$V(\mathbf{x}) = \min\{|x - x_i| \mid (x_i, y)^t \in \partial(X \wedge Y)\}.$$

Likewise, for $\mathbf{x}$ on a horizontal edge, its speed can be defined in a similar way. In case that there is a gap in $X \wedge Y$, the above definition may not be appropriate. However a smoothness property is imposed on V so that if a jump in $V(\mathbf{x})$ is too large then $V(\mathbf{x})$ will be adjusted to its previous neighbor's speed.

In all our experiments, except for few frames, the outcomes of the shape contour extraction are quite satisfactory, e.g., in Figure 2(g),(h), the initial contour and the final contour are shown, respectively.

3. Tracking with Dynamical Graph Matching

After applying background/foreground separation and shape contour extraction, every detected object is enclosed by a shape contour. Our task is now to identify each object by taking account of information provided by the current image frame as well as the tracking outcome so far.

Before proceeding to discuss how the detected objects are tracked, we need to define a representation model to characterize an object. In our approach, each object is represented by a probability distribution of intensity values via histogram analysis. The intensity space is divided into n bins, and a well-defined single-valued bin assignment function b is defined uniquely by pixel's intensity value as $b : \mathbf{x} \mapsto \{1, \ldots, n\}$, where $\mathbf{x}$ is any pixel in an image. Suppose now the detected shape contour of an object is C, and the area enclosed is denoted as $A(C)$. Then it can be represented with the following probability distribution,

$$p(u) = \frac{1}{|A(C)|} \sum_{\mathbf{x} \in A(C)} \delta(b(\mathbf{x}) - u)\,,$$

where δ is the Kronecker delta function, and it is clear that $\sum_{u=1}^{n} p(u) = 1$.

3.1. Dynamical Graph Matching

Intuitively, it makes sense to use a bipartite graph to model a multi-object tracking problem. When a new frame is under investigated, what we have is the previously tracking history left behind and the currently detected objects. The two classes of objects are well divided, and finding a best matching among them is the key to determine their identities. Thus, we classify the two classes of nodes in the bipartite graph as *profile nodes* and *object nodes*, where they correspond to the past and the present, respectively. More precisely, both types of nodes have the same type of data structure, called *profile* and *object*, respectively , where position, intensity distribution, and dimension of its enclosing bounding box are stored.

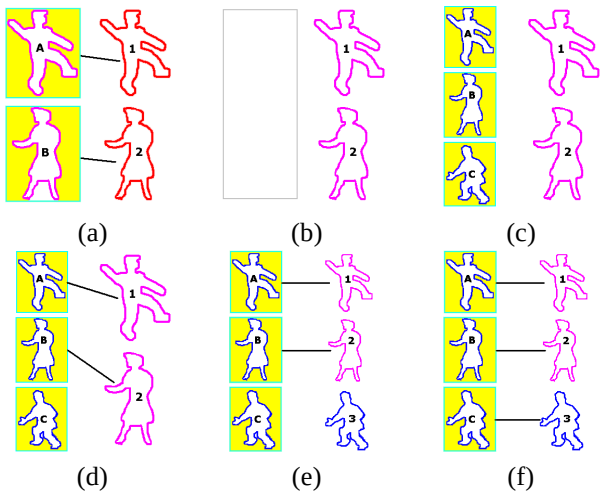


Figure 3. (a) End of frame $f-1$. The matching information is used to update the profiles used in frame f.(b)Beginning of frame f. The bipartite graph only has 2 profile nodes. (c)After contour extraction, 3 objects are detected so the graph now has 3 new object nodes. (d) A bipartite matching is carried out. Since there is one object node left unmatched, a profile node is created to match it as shown in (e) and (f).

During tracking, say at the beginning of frame f (for illustration, see Figure 3), the profile nodes are constructed first according to the tracking outcome from last frame. Notice that the number of profile nodes reveals how many objects are in the scene in the beginning of frame f. (Later, we will explain some of the nodes may need to be counted according to its *multiplicity* due to interaction.) After the foreground pixels and shape contour extraction, the number of objects currently in the scene is determined. Thus, the same number of object nodes are created. We then use a bipartite matching algorithm to find the best match to resolve the identities. If there are any unmatched object nodes

left, this implies that new objects have been detected so new profiles will be created to track them.

For convenience, the object nodes are named in an alphabetical order depending on the raster order at each image frame. To name the profile nodes, the system maintains a *non-decreasing* global numeral counter. Each time a profile data structure is created the counter will be increased by 1. Since objects may enter or leave the scene indefinitely, the number of nodes in the graph changes dynamically. However, an invariant property is always maintained that the numbers of nodes of both types are kept the same. During tracking, when interactions of multiple objects cause them to be segmented inside one single bounding box, the corresponding object node in the graph will be represented with a multi-person icon, and such nodes will be counted according to their multiplicities (see Figure 5(c),(g)). In the following, we summarize the details of the tracking via bipartite matching algorithm.

1. The matching cost/dissimilarity between a profile node and an object node is measured by the *Kullback-Leibler distance*,

$$D(p(u)||q(u)) = \sum_{u=1}^{n} p(u) \log \frac{p(u)}{q(u)} ,$$

   where $p(u)$ and $q(u)$ are the corresponding intensity probability distribution of the profile and object, respectively. After finding the optimal bipartite match, the position, and the dimension of every matched profile are set to the same values of its corresponding object except that its new representation model is updated by averaging the intensity distributions from each matching pair.

2. If there exist unmatched object nodes, then new profiles are is created for each newly detected object b

   If there exist unmatched object nodes, then a new profile is created for each of them, by setting all of its features to be the same as the associated object node's. Note that when an object is entering the scene, the properties of its dimension change continuously. Thus it is appropriate to assume that an object will only be tracked/matched when it completely enters the scene.

3. When an object leaves the scene, its corresponding profile will become unmatched. To detect such event, an aging tag is maintained in the profile data structure. An aging tag will be reset to 0 every time a profile is matched to some object. However, the aging tag of an unmatched profile will be increased by 1 to indicate the profile is about to be deleted. A profile node will be deleted from the bipartite graph if its aging value exceeds a threshold.

4. After finding the optimal bipartite matching, we check all profiles to see if there are some profiles getting too close to others. If so, this is a good indication that interactions are likely to happen soon. Those profiles will be marked as "TBM" (to-be-merged). In the next image frame, the TBM profiles will be processed separately to see if any interaction has occurred or not. More precisely, we check every bounding box's position and dimension to see if it covers more than one TBM profile significantly. In case this is true, an interaction has occurred, and the number of objects involved is exactly the number of profiles covered by the bounding box. To track the objects engaged in an interaction individually, an *adaptive kernel smoothing* technique is used to detect the modes of the horizontal projection of the distribution for the foreground pixels inside the bounding box [1], [16]. Specifically, we apply the *iterative plug-in* scheme, as suggested in [1], to the projected distribution to find its modes. If the number of derived modes is no less than the number of objects, then the objects are tracked separately (see Figure 5(d),(h)). Only when insufficient modes are detected, some of the objects will be tracked as a whole (see Figure 5(c),(g)). The system will resume to track these interacting objects separately when sufficient modes are detected again, or they no longer take part in the interaction.

Note that the whole process of bipartite matching must fulfill the invariant property that the number of profiles and bounding boxes must be kept equal. By maintaining the invariant property, it becomes more manageable to track objects with interactions, and yields a more stable system.

3.2. Background Change Adaption

Most background-extraction based tracking systems are vulnerable to abrupt changes in the reference background. Of course, it is not possible to build a tracking system to account for all sorts of background change scenarios. We concentrate on problems caused by objects that are *almost* stationary during the training stage, and later start to move indefinitely.

The key idea is to integrate background differencing with inter-frame temporal differencing. When the algorithm detects a new object, and it is not matched to any of the existing profiles, if this is not due to an object entering the scene or an object breaking from others (see Figure 5(a),(b)), it must be caused by some object that is part of the background. To handle such events, an object of this kind will not be processed until the object moves to a certain distance away from its original location. This can be detected, say at frame f, when it starts to separate into two bounding boxes. We then check the set of *moving pixels*, consisting of the

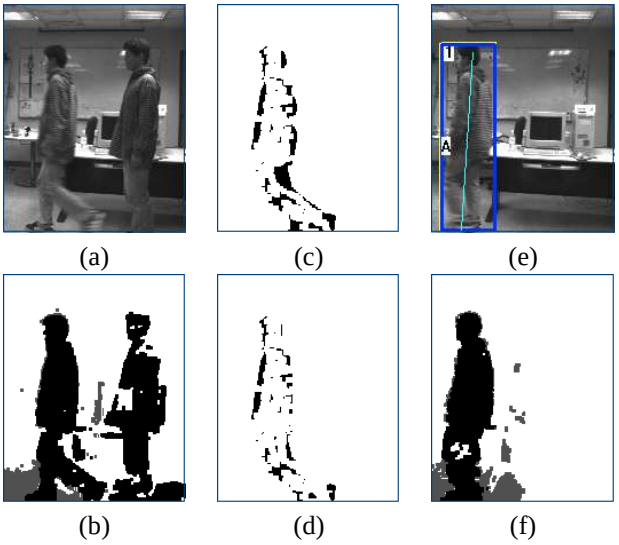


Figure 4. (a) Two image frames, say frame 1 and f, of a video sequence are pasted together to show a person's motion at frame f. The same person on the right is cropped from frame 1 to indicate he is originally part of the referenced background. (b) The result of frame differencing between frame f and the reference background. (c) The absolute inter-frame temporal differencing between frame f and $f-1$. (d) The *moving pixels* are obtained by an intersection of (b) and (c). (e) The object has been detected. (f) The background referencing indicates that the reference background has been updated.

intersection between the background differencing and the absolute temporal differencing between frame $f-1$ and f (see Figure 4). Only the bounding box containing more moving pixels should be kept, and a profile will be generated to match it. It is because that the deleted bounding box is caused by the background change. To adapt to the background change, for each pixel $p_{j,k}$ inside the deleted bounding box of frame f, the corresponding reference mean $\mu_{j,k}$ will be set to $p_{j,k}$'s intensity value, and $\sigma^2_{j,k}$ set to 0. In the following frames, we have to re-train the mean and variance for each $p_{j,k}$ inside this region using the iterative training rules (1). However, it is not required to wait until the re-training process to complete for the system to extract foreground pixels in the region. In our experiments, the system starts to extract foreground pixels only 2 frames after the reference background has been updated, and the results are fairly good.

4. Experimental Results and Discussion

We have presented a tracking system using shape contour extraction and dynamical graph matching. Overall, the approach is effective and promising. Our system runs comfortably at 20fps on a P-III 1GHz PC. For illustration, two sets of tracking results are provided. The first is to demonstrate that the tracker can deal with interactions. The second is to show that it works reliably even when there are significant changes in the reference background. We are currently investigating into adding other possible profile features to assure the tracker's performance, and extending the algorithm for tracking via a non-static camera.

Acknowledgments

This work was supported in part by an NSC grant 90-2213-E-001-016 and the Institute of Information Science, Academia Sinica, Taiwan.

References

[1] A. Berlinet and L. Devroye, "A Comparison of Kernel Density Estimates," *Publications de I'Institut de Statistique de I'Universite de Paris*, Vol. 38(3), pp.3-59, 1994.

[2] H-T. Chen and T-L. Liu, "Trust-Region Methods for Real-Time Tracking," *8th ICCV*, Vol. 2, pp. 717-722, 2001.

[3] D. Comaniciu, V. Ramesh and P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift," *CVPR*, Vol. 2, pp. 142-149, Hilton Head, SC, 2000.

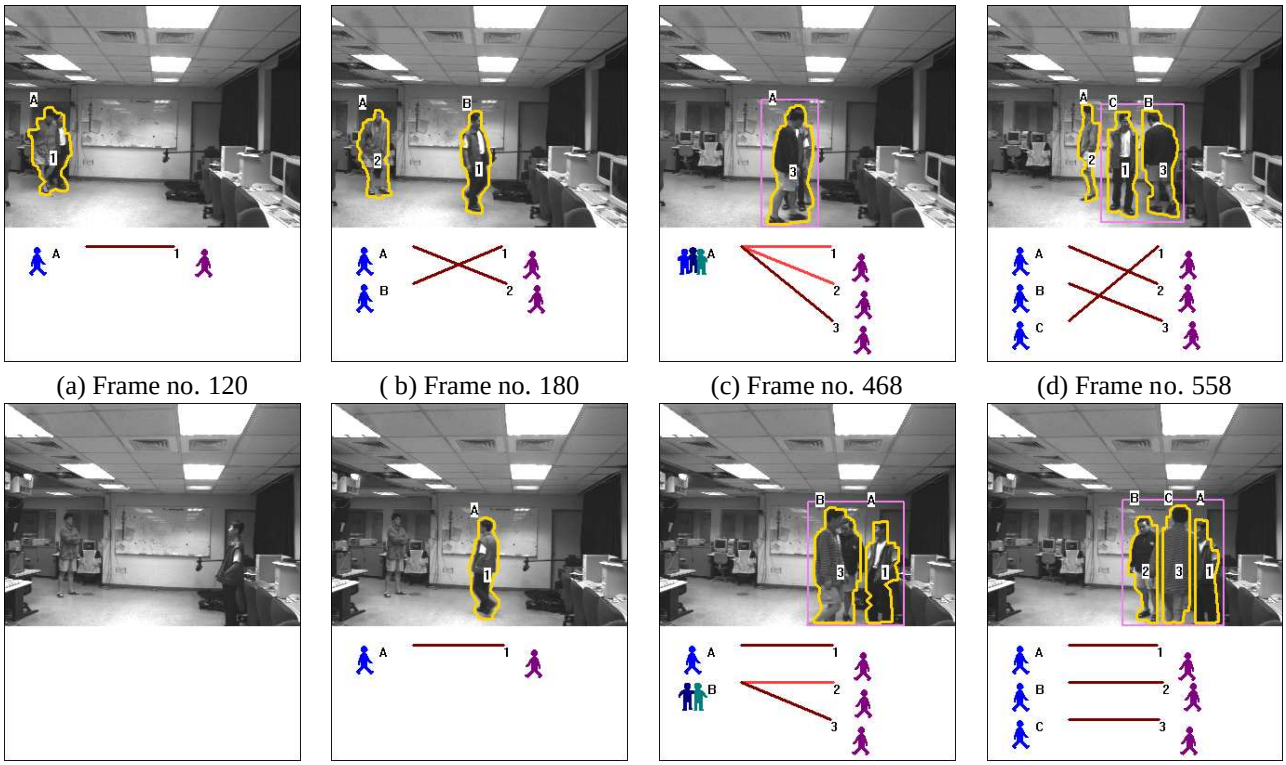
[4] R. Cutler and L. Davis, "Real-Time Periodic Motion Detection, Analysis, and Applications," *CVPR*, Vol. 2, pp. 326-332, Fort Collins, CO, 1999.

[5] I. Haritaoglu, R. Cutler, D. Harwood, and L. Davis, "Backpack: Detection of People Carrying Objects Using Silhouettes," *7th ICCV*, pp. 102-107, Corfu, Greece, 1999.

[6] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Who? When? Where? What? A Real Time System for Detecting and Tracking People," *AFGR*, Nara, Japan, 1998.

[7] M. Isard and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," *ECCV*, pp. 343-356, Cambridge, England, 1996.

[8] M. Isard and A. Blake, "CONDENSATION – Conditional Density Propagation for Visual Tracking," *Int. J. Computer Vision*, (29), 1, pp. 5-28, 1998.



(a) Frame no. 120 (b) Frame no. 180 (c) Frame no. 468 (d) Frame no. 558

(e) Frame no. 0 (f) Frame no. 43 (g) Frame no. 261 (h) Frame no. 282

Figure 5. (a)-(d) Tracking with interactions. (a) Two persons entering the scene together. (b) Separating into two objects. (c) Profile 1, 2, and 3 are matched to a 3-person object node A. (d) After interactions, each object is tracked correctly. (e)-(f) Background change adaption. (e) Initially there are two persons (almost stationary) appeared in the background. So they have been trained into the reference background. (f) One background object starts to move and is detected correctly. (g),(h) The two initially stationary persons interact with a third person.

[9] M. Isard and J. MacCormick, "BraMBLe: A Bayesian Multiple-Blob Tracker," *8th ICCV*, Vol. 2, pp. 34-41, 2001.

[10] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving Target Classification and Tracking from Real Time Video," *DARPA*, pp. 129-136, Monterey, CA, 1998.

[11] S. J. McKenna, S. Jabri, Z. Duric and H. Wechsler, "Tracking Interacting People," *AFGR*, Grenoble, France, 2000.

[12] N. Paragios and R. Deriche, "Geodesic Active Regions for Motion Estimation and Tracking," *7th ICCV*, Corfu, Greece, 1999.

[13] N. Paragios and R. Deriche, "Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects," *PAMI*(22), No. 3, pp. 266-280, March 2000.

[14] P. L. Rosin and T. Ellis, "Image difference threshold strategies and shadow detection," *Proceedings of the 6th British Machine Vision Conference*, pp. 347-356, BMVA Press, 1995.

[15] K. Toyama and A. Blake, "Probabilistic Tracking in a Metric Space," *8th ICCV*, Vol. 2, pp. 50-57, 2001.

[16] M. P. Wand and M. C. Jones, "Kernel Smoothing," Chapman and Hall, 1995.

[17] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *PAMI*, 19(7), pp. 780-785 July 1997.

[18] Y. Wu and T. S. Huang, "A Co-inference Approach to Robust Visual Tracking," *8th ICCV*, Vol. 2, pp. 26-33, 2001.