Multiple Object Tracking using a Neural Cost Function

James Humphreys [a] Andrew Hunter [b,*]

[a] *James Humphreys, Flat 7, Pembroke House, Station Road, Borehamwood, WD6 1DB, UK*

[b] *Centre for Visual Surveillance and Machine Perception, University of Lincoln, Brayford Pool, Lincoln, LN2 7TS, UK*

Abstract

This paper presents a new approach to the tracking of multiple objects in CCTV surveillance using a combination of simple neural cost functions based on Self-Organizing Maps, and a greedy assignment algorithm. Using a reference standard data set and an exhaustive search algorithm for benchmarking, we show that the cost function plays the most significant role in realizing high levels of performance. The neural cost function's context-sensitive treatment of appearance, change of appearance and trajectory yield better tracking than a simple, explicitly designed cost function. The algorithm matches 98.8% of objects to within 15 pixels.

Key words: Surveillance, Tracking, Background differencing, Self-organizing maps, Neural networks
PACS:

Automated video surveillance systems monitor CCTV systems and detect anomalous or suspicious behavior [3] [6] [4] [7] [2], with a view to alerting human operators who can take appropriate actions. The most popular approaches rely on motion detection algorithms to identify objects of interest. Two main classes of motion detection algorithms are used: optic flow algorithms based on tracking movement of some salient points [8], and background differencing-based systems, which assume a static camera, maintain a model of the background of the scene, and detect foreground objects by calculating the difference between the current scene and the background model [9] [6] [4].

Background differencing algorithms yield, at each frame, a binary image, the *silhouette map*, with silhouettes (connected components) corresponding to

* corresponding author, +44 1522 886456
Email address: ahunter@lincoln.ac.uk (Andrew Hunter).

moving objects. A good overview of the manifold problems inherent in the approach is given in the papers by Javed [10] and Toyama *et al* [11]. Surveillance systems based on background differencing use the silhouette map to detect objects of interest (typically people and/or vehicles) which are then tracked; the trajectory of the objects may then be used to highlight objects of interest. A typical system thus has three stages: background differencing, tracking, and event detection. Events of interest may be defined either explicitly (e.g. movement within a defined zone), or implicitly (e.g. anomalous or unusual object trajectories). The output of the tracking system is a number of object identities and trajectories, which may be used for a variety of purposes, including anomalous behavior detection [5] [12] [13]. Here, we discuss only the tracking problem.

This paper is concerned with the tracking phase in a background-differencing based system (further details are available in [14]). This phase must deal with a variety of common problems, including: occlusions (which may lead to objects disappearing temporarily from view), mutual occlusions (where multiple moving objects overlap – tracking objects in busy scenes is extremely challenging), appearance and disappearance (as objects move in and out of the frame), and inherent failings of the background differencing approach. The latter include: false positive silhouettes induced by reflection artefacts, shadows, camera jitter, and other problems; object fragmentation (where an object is broken into several silhouettes); object merging (where two or more objects create a single silhouette).

A number of approaches to tracking have been proposed in the context of background-differencing based systems. We are concerned with the approach where the algorithm assigns an object identity to newly detected silhouettes, and attempts to maintain this identity through successive frames, deleting objects when they ultimately disappear [1] [15] [9] [3].

The tracking component essentially solves a *correspondence* problem (assigning silhouettes to objects) – with the added complication that objects may be created and deleted. The difficulty of the problem relates to the number of objects in the scene; background differencing based systems are able to deal only with low to medium density traffic, with advances in tracking contributing to the ability to deal with higher densities. The correspondence problem is solved by exploiting consistencies in the objects between frames – typically of motion, position and/or appearance. Cost functions are often defined, implicitly or explicitly, to characterize the appropriateness of assigning particular silhouettes to particular objects, combined with algorithms that try out different assignment possibilities in the search for a low cost solution.

Simple approaches to the correspondence include matching simple features of the objects and silhouettes, such as bounding boxes, areas and appearance

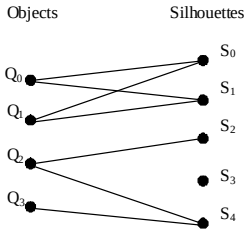
histograms [9]. A number of authors have investigated relatively sophisticated methods based on matching shape models, including splines [4], eigenshapes [1] and active contours [15]. These matching algorithms are typically integrated with simple search algorithms designed to make local searches for appropriate silhouettes to match.

In this paper we cast this correspondence problem explicitly as one of defining two components: an appropriate *cost function*, together with a *search algorithm* to identify an acceptable (and preferably optimal) correspondence. We use a detailed reference standard data set with correct correspondences defined, and an exhaustive search algorithm, as benchmarks. This allows us to explicitly separate the issues of cost function and search algorithm development. We demonstrate two effective cost functions: one using a simple, explicit appearance discrepancy measure; the other an implicit neural cost function using Self-Organizing Feature Maps. The latter demonstrates the improvements that can be made by taken into account position in the scene, and contextual information such as typical behavior, in solving the correspondence problem. For the search problem, we introduce a simple greedy algorithm with close to optimal performance. These experiments indicate that a relatively simple features are sufficient to allow robust tracking, provided that the cost function is sophisticated and makes full use of both appearance and motion invariances.

1 The correspondence problem

Background differencing maintains a background image, $\mathbf{B^t}$; as each new frame $\mathbf{F^t}$ arrives, it updates the background image, and produces a foreground map (a binary image), $\mathbf{S^t}$, by thresholding the difference image $\mathbf{S^t} = (\mathbf{F^t} - \mathbf{B^t}) > \theta$ (and possible applying some morphological clean-up). Connected components in the foreground map are called *silhouettes*, S_j^t. To track objects, the algorithm maintains a relationship between N object records, Q_i, and the M silhouettes, S_j; this is conveniently described using a *bipartite graph*, which may be represented by a binary $M \times N$ *match matrix* (see figure 1). The correspondence problem requires, on each time-step, the generation of a new match matrix, based on the previous time-step and current silhouette image.

Our approach is to define a *search algorithm* which produces *candidate* match matrices. Potential algorithms range from the simple, fast *greedy* algorithm which produces a single (possibly sub-optimal) match matrix, to the slow, reliable *exhaustive* algorithm which produces all possible match matrices within certain constraints. In both cases, the search algorithm makes use of a *cost function*, which is designed to identify good object-silhouette matches by comparing the current object position, motion and appearance with the previous frame.



	S_0	S_1	S_2	S_3	S_4
Q_0	1	1	0	0	0
Q_1	1	1	0	0	0
Q_2	0	0	1	0	0
Q_3	0	0	1	0	1

Fig. 1. A candidate match matrix illustrated as a bipartite graph

A number of complications arise in establishing a proper correspondence. First, multiple silhouettes may be assigned to a single object, which has broken up during detection. Second, multiple objects may correspond to a single silhouette, due to proximity. Third, there may be spurious silhouettes caused by noise, shadows, reflections, and other artefacts. Fourth, parts of objects may sometimes not be represented in silhouettes, due to occlusions or low contrast with the background. Fifth, silhouettes may sometimes include both parts of an object and spurious areas (most commonly, from shadows). Sixth, object appearance may change over time, for a variety of reasons, but most commonly changes of facing. Complex cases may involve several of these factors simultaneously.

The cost functions require that a one to one correspondence between objects and silhouettes be established. To deal with the first and second issues, we allow silhouettes to be *merged* or *partitioned* (resulting in new silhouettes) in a *conflict resolution* step. We also allow silhouettes to be left unmatched, corresponding to noise patches. This may create a greater or smaller number of silhouettes than the number of original connected components. We may also create new objects to correspond to silhouettes that are not assigned to existing objects. Each object is ultimately assigned a single silhouette (which may or may not have been created by merging and/or partitioning), and this silhouette is considered to define the position and appearance of the object.

The overall structure is therefore of a search algorithm that generates one or more candidate match matrices; a conflict resolution step that resolves such match matrices so that only one to one correspondences between objects and silhouettes exists; and a cost function which evaluates the resulting correspondence to drive the search algorithm.

2 An explicit cost function

In general, we expect that a moving object will retain approximately the same appearance from frame to frame. There may be some changes, including: scale changes due to distance from the camera, lighting conditions in different parts

of the scene, facing of object with respect to the camera, articulation of the object (e.g. human pose), and changes in occlusion. However, for a sufficiently rapid frame rate these changes are usually reasonably gradual. In addition, we expect that the position of an object will not change very much in a single frame, and will do so in a predictable fashion. Many trackers implicitly or explicitly construct a cost function based on differences between expected and actual measurements of appearance and/or position. For appearance correspondence, we may extract a number of features, including: size (in pixels), aspect ratio, bounding box, color or intensity histogram, etc. To ensure efficient merging, it is helpful if these features can be *composed* (i.e. given two silhouettes, it is possible to directly calculate the features of the union of the silhouettes from the features of the individual silhouettes); partitioning invariably involves recourse back to the pixel values.

Let S_j^t be the j^{th} silhouette at frame t, and Q_i^{t-1} be the i^{th} object record. Let $\mathbf{f}$ be a function mapping a silhouette or object record to a column vector of features (which may also involve sampling the pixels corresponding to the silhouette). Let $g(.)$ be a matching cost function.

The cost function g is easily defined for matching a particular object, Q_i^{t-1}, with a particular silhouette, S_j^t, based on disparity between the two; see equation 1. A typical choice for g is a weighted sum-square of the disparities; see equation 2.

$$E_{i,j}^t = g(\mathbf{f}(Q_i^{t-1}) - \mathbf{f}(S_j^t)) \tag{1}$$

$$g(\mathbf{x}) = \mathbf{x}'.\mathbf{x} \tag{2}$$

A global cost function, for a particular configuration matching all objects and silhouettes, is derived by summing the costs of individual matches. However, the issue is complicated by the need to handle multiple matches, object creation and object deletion. We therefore introduce a *conflict resolution* step, which takes a given match matrix $\mathbf{M}$, and produces a new match matrix $\mathbf{C}(\mathbf{M})$ with a new set of silhouettes and objects such that each object is matched to exactly one silhouette, and each silhouette to exactly one object (a binary monomial matrix). New objects are created and existing objects removed as necessary during the conflict resolution stage. We may then define the global cost function as in equation 3, where δ is an object creation cost, u is the number of newly-created objects, γ is an object removal cost function, and c the number of removed objects. These costs prevent excessive object removal and addition.

$$E^t = \sum_{i,j \in M} E_{i,j}^t + c\delta + u\gamma \tag{3}$$

We represent silhouettes and objects using a feature vector $\mathbf{f} = (x, y, a, h, w, \mathbf{g})$, where (x, y) is the centroid position of the pixels, a the area in pixels, h and w the height and width respectively of the bounding box, and $\mathbf{g} = (g_k)$ is the 16-bin normalized histogram of the pixel intensities ($\sum_k g_k = 1$). Let $\mathbf{f}_i$ and $\mathbf{f}_j$ represent the feature vectors for the i^{th} object and j^{th} silhouette respectively. Then we define the matching cost using normalized disparities in the size and appearance histogram, following a cost function implicitly defined by Owens [9]; see equation 4.

$$E_{i,j} = pA + pH + pW + dH = \frac{a_i - a_s}{a_i} + \frac{h_i - h_s}{h_i} + \frac{w_i - w_s}{w_i} + \|\mathbf{g}^i - \mathbf{g}^j\| \quad (4)$$

3 A Neural Approach to Cost Functions

Self-Organizing Feature Maps (SOMs), are frequently used to characterize normal and abnormal features. For example, in surveillance they may be used to identify anomalous trajectories [5] [12] [13]. In such systems, a large number of trajectories are tracked, and the SOM is trained by examples to identify normal and unusual activity. Such systems can pick up surprisingly subtle events, and have the potential benefit of context-sensitivity (e.g. particular events may be more common in particular parts of the scene).

In this section we propose the use of SOMs to provide the matching cost function, by learning based on hand-marked reference matches. SOMs are designed to produce a novelty signal, which is monotonically related to the *a posteriori* probability of observing the given activity, assuming a "normal" event (i.e. one which is consistent with the events present in the training set). Matching cost functions require precisely such signals, with lower probability matches being accorded higher costs. The use of a SOM allows us to capture subtleties, and context-specific issues, that are very hard to build into a "hand-coded" cost function. The price paid for this improved performance is the need to hand-label a reference match set.

The cost function is provided by three SOMs, which characterise correct matches from objects to silhouettes: the *Motion SOM*, *Comparative SOM*, and *Appearance SOM*. In operation, a proposed match of object Q_i to silhouette S_j is costed by assuming that the match is made, and then summing the output of the SOMs (the novelty signal, the Euclidian distance from the input feature vector to the prototype vector of the winning neuron). Each 40×40 SOM is trained using a classic two-phase approach: 100 iterations with learning rate $\alpha = 0.1 \rightarrow 0.02$ and neighbourhood $w = 3 \rightarrow 1$, then 1000 iterations with $a = 0.1 \rightarrow 0.02$, $w = 0$.

The Motion SOM is similar to that described by Owens et. al. [5] to perform

motion analysis. It is trained to detect whether the combination of current and recent positions is a usual combination. In its original role, it indicates whether a detected object trajectory is unusual (and therefore worthy of operator attention) or not, and can be deployed to this intent in addition to being used as part of the matching cost function. The SOM has eight inputs: $(x, y, dx, dy, w(x), w(y), w(dx), w(dy))$ where (x, y) is the current position, the motion vector (dx, dy) is given by $(dx, dy) = (x_t - x_{t-1}, y_t - y_{t-1})$, and $w(.)$ is a time-smoothed average function given by equation 5, where $n = 5$ is the window size.

$$w_t(x) = \frac{1}{n} x_t + \frac{n-1}{n} w_{t-1} \tag{5}$$

This SOM learns to identify normal motion patterns, which are locale-specific. For example, in an area where north-to-south motion is normal it will generate a high cost for south-to-north motion. It is worth contrasting this approach with the use of prediction-corrective tracking (e.g. Kalman filtering), which is typically not locale-specific, and therefore gives preference to conservation of movement [16] [17]. Consider a location with a sharply-turning path; the Motion SOM will learn to treat a rapid change of direction to follow the path as normal, while motion directly ahead and leaving the path may be unusual; in contrast, the Kalman filter is likely to predict that the next location will be straight ahead and off the path. This location-sensitivity gives the Motion SOM advantages in resolving uncertainties in match-conflicts. The effect is to favor matches which produce normal movement patterns; for example, if two objects taking different paths come together and separate, this element of the cost function will tend to select the match which produces the most usual pair of resulting trajectories.

The Comparative SOM plays a similar role to the Owen's cost function presented in the previous section. It has eight inputs: $(x, y, dx, dy, pA, pH, pW, dH)$, where (x, y) is the centroid position, (dx, dy) the motion vector, and pA, pW, pH and dH are the four terms in the Owen's cost function; see equation 4. The last four terms allow the Comparative SOM to assign costs to *changes* in the basic appearance of object. The first four terms provide the "context," allowing the system to estimate the cost differently according to position and velocity. For example, an object entering the edge of the screen tends to grow in size rapidly as it becomes visible. Similarly, pedestrians exiting their vehicles may be partially occluded by their own or other cars; they then apparently grow in size as they emerge from occlusion. The Comparative SOM learns to model these localized effects, becoming tolerant of various context-specific changes in object appearance.

The Appearance SOM is the third element of the system. Its role is to assess whether the appearance of the object is normal – again, in a location-specific

way. In contrast with the Comparative SOM, it assesses the absolute appearance rather than change in appearance. The inputs are (x, y, a, ar, h, w), with (x, y) the centroid location, a the area, (ar, h, w) the aspect-ratio, height and width of the bounding box. Critically, the object size typically varies with y (since high-mounted cameras tilted downwards show more distant objects higher on the y axis). More subtle location-specific variations may also be captured, including again differences due to partial occlusions and appearance/disappearance zones.

Arguably, the three SOMs could be combined into a single SOM with an input vector including the features of all three; however, our experiments indicate inferior performance, no doubt due to the higher dimensionality. We have also experimented with the use of a single set of SOMs for all objects, versus two sets of SOMs – one for pedestrians, and one for vehicles. Again, although in principle a single SOM should suffice, we achieved higher performance with separate versions for pedestrians and vehicles.

In order to apply separate SOMs for pedestrians and vehicles, it is necessary to classify objects. We experimented with two classifiers; a simple Bayesian classifier based solely on object area, which correctly classified 95% of objects; and a Multilayer Perception with input vector $(a, w, h, ar, \max(s), y)$ (where s is the inter-frame centroid speed in pixels), and four hidden units, which correctly classified 99.2%. The rare object classification failures of the MLP are invariably due to occlusion events. Further details are available in reference [14].

4 Conflict Resolution

The conflict resolution stage processes a match matrix with conflicts (multiple objects matched to a single silhouette and/or multiple silhouettes matched to a single object), and removes these conflicts by *merging* and *splitting* silhouettes. In merging, multiple silhouettes are combined into a single silhouette; in splitting, a silhouette is divided into several sub-silhouettes (one per matching object).

In merging, the new silhouette is effectively created from the statistics of the union of the pixels in the merged silhouettes. However, the use of composable features allows us to generate a new feature vector efficiently. Given the silhouette feature vectors $\mathbf{f}_1 = (x_1, y_1, a_1, h_1, w_1, \mathbf{g}_1)$ and $\mathbf{f}_2 = (x_2, y_2, a_2, h_2, w_2, \mathbf{g}_2)$, and the bounding box definitions $((t_i, b_i, l_i, r_i)$, $h_i = t_i - b_i$, $w_i = r_i - l_i$, $i \in 1, 2)$ the merged vector is calculated as below.

$$a_m = a_1 + a_2 \tag{6}$$

$$x_m = \frac{x_1 a_1 + x_2 a_2}{a_m} \tag{7}$$

$$y_m = \frac{y_1 a_1 + y_2 a_2}{a_m} \tag{8}$$

$$g_m^i = \frac{g_1^i a_1 + g_2^i a_2}{a_m} \tag{9}$$

$$l_m = \min(l_1, l_2); r_m = \max(r_1, r_2); t_m = \min(t_1, t_2); b_m = \max(b_1, b_2) \tag{10}$$

Splitting is more complex. Our approach is to split a silhouette that is matched to multiple objects along a single direction, in proportion to the size of the source objects. Our first approach was to calculate the principal component of the pixel coordinates of the silhouette, and to split along a line orthogonal to this. This approach relies on the assumption that the principal axis is likely to run between objects, and that the level of mutual occlusion is not too high. The first of these assumptions is routinely violated by pedestrians walking side by side, a problem which may be corrected by adjusting the covariance matrix in the PCA calculations using the typical pedestrian aspect ratio (2.5); see figure 2.

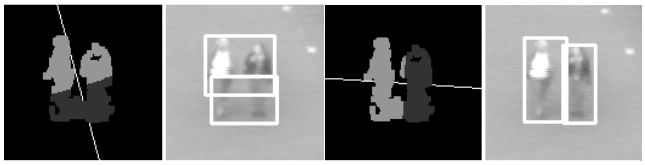


Fig. 2. The PCA-based partitioning of two merged pedestrians. The merged silhouette is taller than it is wide, causing partitioning to fail. Scaling corrects the problem.

A more computationally expensive approach to line partitioning is to search through a number of splits at different angles, choosing that with the lowest cost (we take 30 divisions at every 15 degrees); see figure 3. This algorithm has excellent performance providing its inherent assumptions are not violated: the objects have been tracked correctly, there is little mutual occlusion, division along a line is possible, and the visible area has not changed significantly, and we use it throughout this work. Figure 3 also shows the cost versus angle for this case, where the cost is estimated using the SOMs on the silhouettes arising from each tested angle.

The conflict resolution algorithm applies merges and splitting efficiently as follows. The (binary) candidate match matrix can also be represented using a bipartite graph; see figure 1. The match matrix is augmented with two

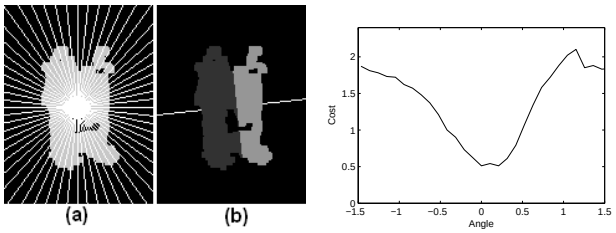


Fig. 3. Angle-search method – (a) 30 different angles are assessed, and (b) the best angle is chosen

additional working states, as follows:

$$M_c(q,s) = \begin{cases} 0 \text{ No link between silhouette and object} \\ 1 \text{ Link between object and silhouette that needs resolving} \\ 2 \text{ Secure link between object and silhouette} \\ 3 \text{ Silhouette-object match no longer possible} \end{cases}$$

All matches are initialized to state 1. Splitting and merging operations both result in extension of the match matrix, with new silhouettes and/or objects added, and some original silhouettes marked as no longer used (state 3). Ultimately, all matches resolve to state 2 (secure links). The algorithm has three steps, with the first two repeated until there are no conflicts; the final step consolidates results; see figure 4. In step one, fully connected subgraphs are resolved. Such a sub-graph contains silhouettes and objects that have been mutually assigned to each other and to no others. Trivial cases include one to one matches (which are accepted as is), a single object matching multiple silhouettes (which are merged), and a single silhouette which matches multiple objects (which are split). Many to many fully connected subgraphs typically occur when a group of objects are moving closely together and the joint silhouette breaks up due to noise effects. Such a subgraph is resolved by merging all the silhouettes and then splitting using the algorithms discussed above. Subgraph identification is performed efficiently by sorting objects by valency (since members of a fully connected subgraph must have the same valency).

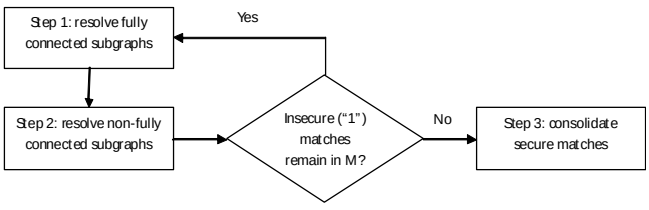


Fig. 4. An overview of the steps taken by the conflict resolution module

Step two reduces the number of insecure matches, yielding further fully connected subgraphs that can be resolved by iteration through step one. This step

deals with the unsatisfactory situation where we have non-fully connected subgraphs, where it is not so clear how to merge and/or partition. One option is to enforce full connection within the subgraph and treat as above. However, this leads to a less powerful search algorithm than the method described below. Instead, we find the lowest valency silhouette, and make the match secure for this silhouette, splitting it in the case that the valency is greater than one. The areas of the matched object(s) are temporarily adjusted by subtraction of the silhouette area, so that any further splitting is appropriate. The most common case here is that a single silhouette (valency one) is firmly matched to an object, and iteration through step one leads to further merges and splits to resolve the remaining areas of that object.

Step three resolves any remaining matches of multiple silhouettes to single objects – these are firm matches arising when step two has confirmed a match and further iterations of step one have assigned other silhouettes to the same object. This stage ensures that all remaining matches are one to one. With a one to one match established, the cost function can be used to assess how acceptable the assignment is.

5 The Search Algorithm

The object correspondence is solved by a search algorithm, which considers some or all of the possible match matrices arising from the current set of silhouettes and objects. Naively, The number of possible match matrices is 2^{M*N}. We can reduce the size of the problem trivially by specifying a maximum match radius between object and silhouette centroids; this is used to initialize a binary *valid match matrix*, with unit entries only for "in-range" object-silhouette matches. The number of possible match matrices is then 2^V, where V is the number of unit entries in the valid match matrix.

A *global* search algorithm finds the lowest cost of all possible match matrices. This is particularly useful during development, as it allows us to analyze the performance of the cost function separately from the search algorithm. Any inadequate matches generated while using a global search algorithm are the fault of the cost function. When testing any other search algorithm, we can benchmark its performance against the global search algorithm. The simplest global search is the *exhaustive* search algorithm, which considers all possible match matrices. However, this has high computational expenses, and so is not suitable for real-time use.

We introduce a greedy algorithm that is designed to yield an acceptable (but possibly sub-optimal) match matrix rapidly, and is capable of real-time execution. It has four stages:

In stage one, each object in turn is matched to the single valid silhouette with the lowest cost match. Objects are left unmatched only if there are no valid (in range) silhouettes; some silhouettes may be matched to multiple objects. An analysis of the reference standard shows that 98.7% of silhouettes are assigned by this phase to an object that is at least part of that silhouette.

In stage two, we try to verify if potential merges of objects are likely to be valid. Such potential merges are represented by multiple objects assigned to the same silhouette by stage one. We create a temporary "macro object" by merging the object records using the technique described for silhouette merging above, with the modification that the object positions are projected forward one time-step in space using the previous time-step velocity. The cost of matching the macro object to the silhouette is compared with the cost of matching the single best-matching object to the silhouette; if the macro object matches best, it is assumed that a "merge event" has occurred.

If it is concluded that a merge event has not occurred, only the best-matching correspondence is maintained. The other objects are reassigned to their next-best match. This may in turn cause further match conflicts, and the process is repeated until these have all been removed.

In stage three, unmatched silhouettes are considered. Each such silhouette is considered against each valid object; the silhouette is assigned to the object where this yields the lowest cost, provided the resulting cost is less than the cost of leaving the silhouette unassigned (bearing in mind that leaving the silhouette unassigned imposes an "unmatched silhouette" cost). The assignment costs are calculated by using the conflict resolution and global cost function steps.

Step four removes poor object matches. It unmatches each object in turn from its silhouettes, and calculates if this lowers the global cost. Such matches may arise, for example, when an object leaves the scene and the record is matched to a noise silhouette within the valid range.

This greedy algorithm is extremely simple, with complexity $O(MN)$, and has performance close to optimal, as discussed in the next section.

6 Evaluation

A key part of the work reported in this paper is the use of a reference standard data set. In this data set, the correct object-silhouette correspondence is manually identified. Given the complexity of the algorithm, generating such a reference standard offers some challenges, which are discussed below. The

benefits include the ability to develop the cost function and search algorithm independently, and to have a powerful mechanism to evaluate both. This has allowed us to develop algorithms which are both simpler and more powerful than our previously-published versions – including discarding features which intuitively seemed helpful, but in reality were shown to have no impact on performance.

In the context of object tracking, a large number of video frames are required to provide a reasonably diverse set of activities. We used three sequences of 1.5 hours each, sampled at 4 frames per second; there were approximately 15000 frames with some activity. The scene chosen exhibits a mixture of pedestrian and car traffic, with low to medium levels of crowding (up to twenty or so visible objects, but usually much lower).

An ideal reference standard for this problem would have each object labelled, pixel by pixel, for every frame. This is clearly impracticable for such a large number of frames. Given that our aim is to investigate effective algorithms for generating match matrices, we instead use a reference standard which specifies the optimal match matrix. To generate the reference standard, a special version of the tracking programme was developed with a simple greedy assignment algorithm, and the ability to walk through the sequence frame by frame. The user interface displays object identities and types against silhouettes, and allows these to be altered where the greedy algorithm makes mistakes.

In evaluation of a tracking algorithm against the reference standard, it is necessary to "reset" the tracking algorithm objects to those found in the reference standard on each frame – if the tracking algorithm is run continuously, object identities may be permuted with respect to the reference standard, and so match statistics cannot be easily and automatically generated. A disadvantage of this approach is that our statistics do not address the capability of a tracking algorithm to recover from errors.

In both reference standard and tracker, each object from the previous frame may either be matched (assigned to a silhouette), or unmatched. This yields four possible results: if the object is matched in both, we can assess to what extent the match is consistent; if matched in neither, then the object has been correctly removed. If the reference standard has a match and the tracker none, then the object is defined as *lost*. *Hanging* objects are created when the reference standard removes an object, but the tracker retains it (e.g. by spuriously assigning it to a noise patch). In addition, errors sometimes cause the tracker to create entirely new spurious *extra* objects (e.g. as a result of camera jitter). When the object has been matched in both reference standard and tracker, we can assess the correctness of the match in a variety of ways. Perhaps the most useful is the distance between the reference and tracker centroids, which we place into five pixel bins for simplicity of analysis. A

second measure is the number of "flips" - disparities in the match-matrix for a given object (i.e. a count of the number of silhouettes missing or added). This gives us a good picture of the goodness of fit at the match matrix level, although it does not distinguish between the effect of small and large silhouette errors.

The use of the reference standard has several deficiencies. It accepts the background differencing results as fact, although this stage in the processing sometimes produces serious errors. Figure 5 illustrates one extreme case, where a pedestrian is exiting a car and his or her shadow has caused very poor segmentation. Any match matrix will give questionable results here. To avoid distorting effects, we have excluded a small number of frames from the reference standard, where such issues are particularly severe.

Fig. 5. A poorly segmented pedestrian. Whichever match matrix is chosen for the reference standard, the result will always be unsatisfactory

The exhaustive algorithm, although very useful, is extremely slow for frames with large numbers of objects. Consequently, we removed a small proportion of very busy frames (1.3%; the removal of the worst three alone reduced the processing time by a third).

The reference standard data set consists of three sequences of 1.5 hours each. The first sequence was used to *train* the SOMs for the neural cost function. The second sequence, the *selection* set, was used during the development of the system to identify effective algorithms, SOM configuration, and control settings. The third sequence, the *test* sequence, is used to generate unbiased performance statistics, reported below. The test set contains 5785 frames. Figures 6 shows the performance of four system configurations (combining greedy and exhaustive algorithms with Owen's and SOM cost functions) on the test set.

It is apparent that the greatest factor in performance is the cost function, with the SOM neural cost function out-performing the explicit Owens cost function. There is surprisingly little differentiation between the exhaustive and greedy algorithms when the neural cost function is used, indicating that the greedy algorithm is more than adequate to the search task. Using the Owens cost function, the exhaustive algorithm matches 99.72% of objects to within 15 pixels, whereas the greedy algorithm manages only 98.83% (roughly three times the error rate). As the cost function improves, the search algorithm becomes less relevant – perhaps because the first stage match in the greedy

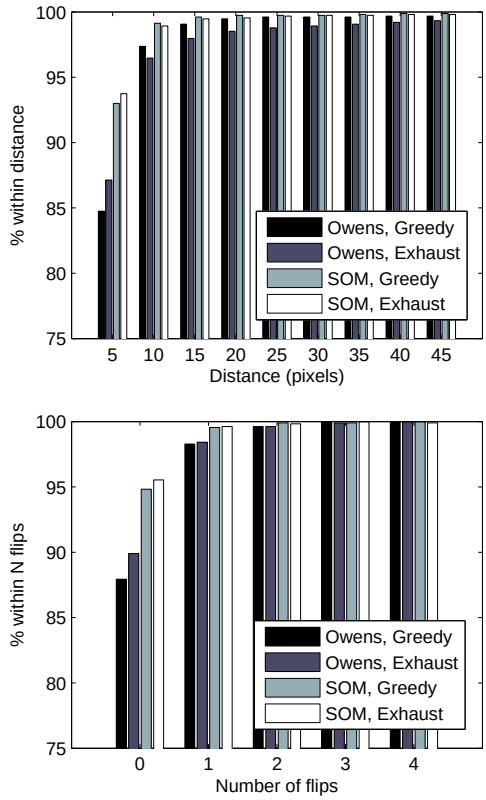


Fig. 6. Percentage of object matches within given distance and flips of the reference standard, using four combinations of cost and search function

algorithm is more likely to be correct. The initial step of the neural cost function finds a silhouette which is part of the object 98.7% of the time, compared with 96.6% for the Owen's cost function.

The number of lost, hanging and extra objects is shown in table 1. The picture is rather more ambiguous here, with the SOM and exhaustive approaches showing overall advantages over the Owen's and greedy approaches, but with some differences in performance. The exhaustive algorithms tend to do worse on hanging objects, as they search more thoroughly for a matching silhouette when an object has gone out of sight, and are consequently more prone to falsely assign the object to a noise silhouette. A substantial number of extra objects are created in the test sequence – these are transients due to camera judder in high winds, and are an inevitable occurrence. However, they do not damage the quality of the tracking of true objects, and are rapidly eliminated by further stages of processing [14].

To illustrate how and why the SOM-based cost function out-performs the Owen's cost function, we conducted a number of sensitivity analysis experiments. The Motion SOM is designed to respond to unusual paths. Figure 7 shows the effect of modifying two normal silhouettes by artificially displacing their positions a short distance at a variety of angles; object A was sampled

Table 1
Lost, hanging and extra objects

Algorithm	Lost	Hanging	Extra
Owens, Greedy	18	14	425
Owens, Exhaust	11	17	416
SOM, Greedy	12	2	411
SOM, Exhaust	1	4	410

at a position where movement up and down the screen are usual, whereas object B was sampled at a position where only downward motion is normal. The graph illustrates that the Motion SOM is extremely sensitive to context-dependent direction of motion.

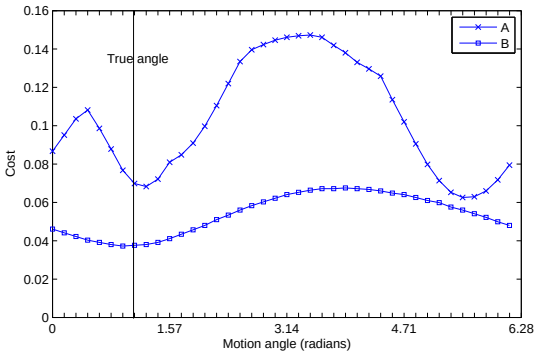


Fig. 7. Performance of the motion SOM, given a specific point and speed but different angles of motion.

Figure 8 illustrates context-sensitive performance of the Comparative SOM. In this experiment, object A is a car ready to park near the centre of the scene; object B is another car entering the camera view. The figure illustrates sensitivity to changes of the object area. For object A, the lowest cost occurs where the area does not change, as expected. However, for object B the Comparative SOM assigns the lowest cost when the area increases at the normal rate for a car entering the scene in this location.

Figure 9 shows the performance of the Appearance SOM, under manipulation of the aspect ratio. Pedestrian A is getting into a car parked amid a row of cars, and consequently is partially occluded, which is usual in this location; pedestrian B is in an uncluttered part of the scene. The SOM demonstrates a clear locale-dependent preference for particular aspect ratios.

It is this ability of the SOM cost function to model context-dependent cost relationships (including position and velocity dependent costs) which makes it so effective. Further sensitivity experiments reveal similar responses to other input variables of the cost function. This is well-illustrated by the performance

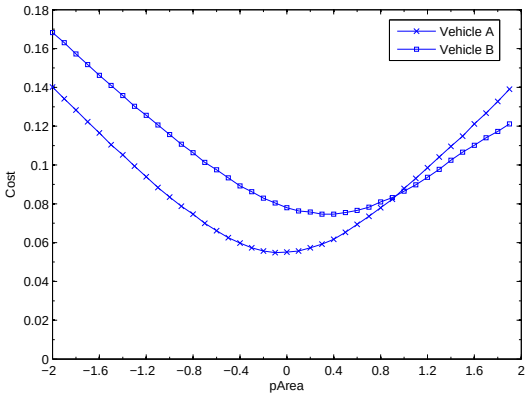


Fig. 8. Performance of the comparative SOM, testing the effect of changing pArea on the output cost. Vehicle A is in the centre of the scene, about to park. Vehicle B is just entering the scene.

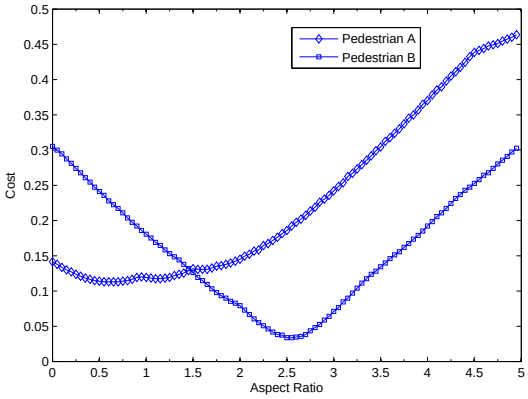


Fig. 9. Performance of the appearance SOM, testing the effect of changing aspect ratio on the output cost. Pedestrian A has just exited his/her vehicle. Pedestrian B is in an unobstructed area of the scene.

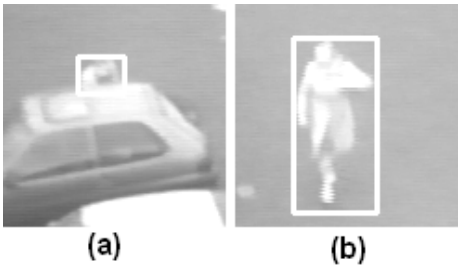


Fig. 10. The two pedestrians used to capture the data for figure 9

of the neural cost function on objects touching the edge of the image, where object appearance and disappearance is most common. Contrary to our initial expectations, the greedy neural system matches 99.55% of such objects to within five pixels, as opposed to 99.1% of non-edge objects, thus indicating that the context-sensitivity of the SOMs is able to handle such appearance

and disappearance zones perfectly adequately, without the need to explicitly model them.

7 Conclusion

We have introduced a new algorithm for the tracking of multiple objects in a background-differencing based system using object to silhouette matching. The algorithm uses a neural cost function based on three SOMs that learn normal patterns of motion, change and appearance of objects in the scene in a context-sensitive fashion. This cost function is combined with a simple, highly effective greedy algorithm to allow object identification consistent with normal patterns of behavior in the scene. We have used a reference standard data set, an explicitly designed cost function and an exhaustive search algorithm to benchmark the new algorithm. This study shows that the context-sensitive performance of the neural cost function is key in achieving good tracking performance, allowing us to use a relatively simple approach to assigning the best match. We have thus established that the use of SOMs in constructing cost functions for configuration problems is viable.

The most significant drawback of the approach is the need for a reference-standard marked-up data set to train the SOMs. The mark-up is a time-consuming procedure, and is clearly not viable for practical use. However, it should be possible to automatically produce a training reference set, by gathering data for a period of time, and inserting into the reference set only data that can be unambiguously identified automatically – that is, where there are no other objects within the vicinity, and a "clean track" (with no break-up of the silhouette) is achieved throughout. This will be the subject of future work.

References

[1] A. Baumberg and D. Hogg. An adaptive eigenshape model. In *Proc of the 6th British Machine Vision Conference, Vol 1, pp 87-96*, 1995.

[2] O. Javed and M. Shah. Tracking and object classification for automated surveillance. In *7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31*, 2002.

[3] A. Hunter, J. Owens, and M. Carpenter. A neural system for automated cctv surveillance. In *IEE Symposium on Intelligent Distributed Surveillance Systems, ed. S. Velastin, 26 Feb. 2003, IEE Savoy Place, London, IEE London, ISSN 0963-3308*, 2003.

[4] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *The third European conference on Computer vision (vol. 1), Stockholm, Sweden*, pages 189–196, 1994.

[5] J. Owens and A. Hunter. Application of the self-organising map to trajectory classification. In *Proc. 3rd IEEE International Workshop on Visual Surveillance*, pages 77–83. Dublin, 2000.

[6] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. In *IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.19, number 7*, pages 780–785, 1997.

[7] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *2nd European Workshop on Advanced Video Based Surveillance Systems, AVBS01. Sept*, 2001.

[8] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2000.

[9] J. Owens, A. Hunter, and E. Fletcher. A fast model-free morphology-based object tracking algorithm. In *British Machine Vision Conference*, volume 2, pages 767–776, 2002.

[10] O. Javed, S. Khurram, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *IEEE Workshop on Motion and Video Computing, Orlando, Dec 5-6*, 2002.

[11] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Seventh International Conference on Computer Vision, Kerkyra, Greece*, pages 255–261, 1999.

[12] J. Owens, A. Hunter, and E. Fletcher. Novelty detection in video surveillance using hierarchical neural networks. In *Proc. International Conference on Artificial Neural Networks (ICANN 2002)*, volume 2, pages 1249–1254. Madrid, 2002.

[13] N. Johnson and D.C. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996.

[14] J.A. Humphreys. The automated tracking of vehicles and pedestrians in cctv for use in the detection of novel behaviour. Master's thesis, University of Durham, Durham, United Kingdom, 2004.

[15] I. Haritaoglu, D. Harwood, and L.S. Davis. w_4:who? when? where? what? a real time system for detecting and tracking people. In *International Conference on Face and Gesture Recognition, Nara, Japan, April 14-16*, 1998.

[16] J.M. Ferryman, A.D. Worral, G.D. Sullivan, and K.D. Baker. Visual surveillance using deformable models of vehicles. *Robotics and Autonomous Systems*, 19(3-4), 1997.

[17] P. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A. Worral, and K. Baker. An integrated traffic and pedestrian model-based vision system. In *British Machine Vision Conference*, volume 2, 1997.

Statement regarding amendments

The reviewer made two comments:

That the model requires training using hand-marked data, and this is time-consuming. As the reviewer notes, we have drawn attention to this, and consider it a subject for future work. It does not invalidate the scientific contribution of the paper, and I think the reviewer has recognised this, and has not actually asked for any change to this part of the paper. A future paper will consider the impact of automatic data gathering for this purpose.

Eqn. four looks like there should be some normalization. The reviewer is correct to recognise a problem. In fact the dH term is formed from a *normalized* histogram, which we had not described properly in the previous paragraph. We have therefore reworded the definition of $\mathbf{g}$ to make this clear. With this definition, the histogram cost is correctly scaled to be composed with the other costs by addition without need for further normalization.