Computing 62, 321–338 (1999)

Computing
© Springer-Verlag 1999
Printed in Austria

Feature Point Tracking for Incomplete Trajectories

D. Chetverikov and J. Verestóy, Budapest

Received June 8, 1998; revised November 18, 1998

Abstract

A new algorithm is presented for feature point based motion tracking in long image sequences. Dynamic scenes with multiple, independently moving objects are considered in which feature points may temporarily disappear, enter and leave the view field. This situation is typical for surveillance and scene monitoring applications.

Most of the existing approaches to feature point tracking have limited capabilities in handling incomplete trajectories, especially when the number of points and their speeds are large, and trajectory ambiguities are frequent. The proposed algorithm was designed to efficiently resolve these ambiguities. Correspondences between moving points are established in a competitive linking process that develops as the trajectories grow. Appearing and disappearing points are treated in a natural way as the points that do not link.

The proposed algorithm compares favorably to efficient alternative algorithms selected and tested in a performance evaluation study.

Key Words: Motion tracking, feature points, algorithms, validation.

1. Introduction

Feature point tracking is a standard task of computer vision with numerous applications in navigation, motion understanding, surveillance, scene monitoring, and video database management. In an image sequence, moving objects are represented by their feature points detected prior to tracking or during tracking. Feature points may have local image properties assigned to them. However, in dynamic scenes these properties are often unstable. Instead of identifying a (physical) point by its neighborhood pattern, in the traditional statement of the feature point tracking problem [4, 6–8] the points are treated as indistinguishable, and kinematic constraints are solely used to establish the correspondences.

In real point-tracking tasks, the kinematic constraints are imposed by the particular application considered. The strongest and most commonly used constraint is the trajectory smoothness one, which stems from the inertia law of physics and limits the accelerations of the moving objects. When more complex motion is tracked, the smoothness constraint is partially relaxed. Other constraints, e.g., spatial proximity

[6], are added.

Alternative approaches to local motion estimation are based on optical flow and patch matching. The reader is referred to [1] for a discussion of these approaches. This paper only deals with feature point based motion tracking.

In many applications, e.g., surveillance and scene monitoring [3], objects may temporarily disappear, group, enter or leave the view field. In some tasks, such events are of particular interest, while in others they are admissible but disturbing. Several tracking methods are available, but neither of them performs well for dense feature point sets with occlusions and motion across image borders. In addition, the existing algorithms have limited scopes of admissible events. We present here a new algorithm called *IPAN Tracker* whose scope includes all events handled by the alternative algorithms. (IPAN stands for Image and Pattern Analysis group.) The new algorithm performs reasonably well even in hard conditions, when the motion is fast and the number of trajectories is large.

Recently, we have initiated a comprehensive performance evaluation study of feature point tracking techniques. A detailed report on this ongoing study is published elsewhere [9, 10]. In the current paper, we use some of the performance evaluation results to demonstrate that the IPAN Tracker compares favorably to the existing tracking techniques.

The paper is organized as follows. Section 2 formulates the general feature point tracking problem considered. Previous work on feature point tracking is discussed in Section 3. The proposed algorithm is presented in Section 4. Section 5 compares the performance of the IPAN Tracker to that of a number of the existing algorithms. The evaluation tests are based on a motion model which generates independently moving points, with frequent point groupings, controllable speed, number of points and occlusion. An application of feature point tracking to estimation of blood flow velocity is discussed. Finally, conclusions are drawn.

2. Problem Statement

Consider a motion sequence of multiple objects viewed by a static camera. Assuming no particular 3D model of motion, the problem is restricted to 2D projection of real 3D motion. Let F_k, $k = 1, 2, \dots, M$, be the kth frame of the sequence, where M is the total number of frames. Assume feature points $P_{k,i}$ have been detected in each F_k prior to tracking. The number of points in the kth frame, N_k, may vary from frame to frame. Denote the total number of distinct points that appear in the sequence by N. This number is equal to the total number of distinct trajectories T. An occluded trajectory counts as one although it consists of two or more pieces.

When a point enters or leaves the view field in any frame $k \neq 1, M$, the trajectory is called *partial*. A trajectory is *broken* if the point temporarily disappears within the view field, and later reappears again. In this case, we speak of (temporary) *occlusion*. If a trajectory is broken, partial, or both, it is called *incomplete*. Entries, exits and temporal occlusions are called *events*.

The feature point tracking problem is a motion correspondence problem under the general assumptions of: (1) indistinguishable points; (2) smooth motion; (3) limited speeds; (4) short occlusions.

Assumption 2 means limited accelerations, i.e., limited changes in motion directions and speeds. The speeds themselves are also limited so that a point is observed a sufficient number of times as it crosses the view field. However, small inter-frame displacements are not assumed. Assumption 4 implies directional continuity of broken trajectories, which makes smoothness applicable to occluded paths as well.

In addition to the general assumptions 1–4, most tracking algorithms use specific assumptions concerning the admissible events. These assumptions are discussed in the next section.

3. Previous Work

Most of the feature point tracking techniques define a smoothness-based cost function for three points from three consecutive frames; typical cost functions will be discussed in Section 4. Different linking strategies are applied to find the correspondences and optimize the trajectories.

Sethi and Jain [8] developed an iterative greedy exchange algorithm to find a suboptimal solution to the motion correspondence problem. This algorithm will be referred to as **SJ87**. The average local smoothness of the trajectory set is maximized, penalizing changes in velocity magnitude and direction. In each new frame the trajectories are extended by first establishing correspondences based on the nearest neighbor relationship, then exchanging those correspondence pairs which yield the largest gain in smoothness. This procedure is repeatedly applied in forward and backward directions until an equilibrium state is reached.

The computational load of SJ87 becomes very high as T grows. The iterative procedure converges slowly. As pointed out in [6], it may occasionally not converge at all. SJ87 assumes that every trajectory extends through the whole sequence: entry and exit are not considered. Occlusion is only indicated in F_k when $N_{k-1} > N_k$. The $N_{k-1} - N_k$ occluded points of F_{k-1} are then identified, and their positions in F_k are found by extrapolation.

The assumption of no entry and exit is restrictive. Based on the idea of [8], Salari and Sethi proposed later an improved algorithm [7], denoted here by **SS90**, in which partial trajectories are allowed for. SS90 uses a speed limit and a cost limit. A trajectory is split if the cost limit is exceeded. An occlusion also splits a trajectory into two partial ones: no occlusion processing is done. The iterative SS90 is again slow for large T, although it is somewhat faster than the original SJ87.

Hwang [4] proposed a heuristic search algorithm denoted here by **HW89**. The algorithm extends the trajectory of a point by predicting its location in the new frame and examining the points lying in the vicinity of this location. A point may belong to several trajectories. For each point, the algorithm keeps track of the best few plausible trajectories. The valid trajectory is then selected based on heuristic reasoning.

HW89 uses a speed limit and indicates that a point has disappeared if a vicinity of the predicted location is empty. The vicinity is defined by the speed limit. This is sufficient but not necessary condition for occlusion, especially in dense point sets. The resulting trajectory set may contain superfluous trajectories, while some points may be assigned no trajectory at all. Exit is a non-admissible event, although entry is allowed.

The non-iterative tracking algorithm by Rangarajan and Shah [6], denoted by **RS91**, uses the same assumptions as SJ87, that is, entry and exit are not admissible. In addition, it is claimed that the initial correspondences between F_1 and F_2 must be given, as smoothness alone is not sufficient. (In [9], we show that RS91 is not robust with respect to its initial conditions: an initialization error may propagate to distant trajectories.)

Given the initial links, RS91 maximizes the proximal uniformity of the trajectories, that is, prefers smooth trajectories and small displacements. Each individual correspondence is optimized for proximal uniformity while trying to keep the overall cost value close to the minimum as possible. The occlusion handling of RS91 is similar to that of SJ87.

Courtney [3] developed an automatic video indexing system for surveillance and scene monitoring. After simple motion-based segmentation, the moving regions are represented by their centroids. These feature points are tracked and the resulting trajectories analyzed to detect and classify the events of interest. Only a few easily detectable, well-separated moving objects are considered. This allows the author to concentrate on event reasoning rather than sophisticated tracking.

A comprehensive way of coping with incomplete trajectories was proposed by Chetverikov and Lerch [2]. Two consecutive frames are matched *given the expected direction of motion* for each point. In some tasks, the direction estimates can be obtained from the shapes of the moving objects. The magnitudes of the velocities are unknown. The algorithm uses directional coherence and competitive linking based on hypothesis testing. This approach has the advantage of uniform handling of events.

The formulation of the tracking problem adopted in [2] differs from the standard one discussed in Section 2. In the proposed algorithm, the idea of [2] is extended to long motion sequences. The assumption of the expected directions being given is now dropped and the standard feature point tracking problem is addressed.

Table 1 summarizes the scopes of five feature point tracking algorithms, including the proposed one (**IP97**). The first column indicates the capability to operate without the prior knowledge of the initial correspondences, called *self-initialization*. The options 'Occlusion', 'Exit' and 'Entry' refer to the capabilities to handle the corresponding events. In the table,'$+$' denotes the presence, '$-$' the absence of an option, while '$\pm$' means a limited capability.

Table 1. Capabilities of feature point tracking algorithms

Algorithm	Self-init.	Occlusion	Exit	Entry
Sethi & Jain SJ87	$+$	$+$	$-$	$-$
Salari & Sethi SS90	$+$	$-$	$+$	$+$
Rang. & Shah RS91	$-$	$+$	$-$	$-$
Hwang HW89	$+$	$\pm$	$-$	$+$
IPAN Tracker IP97	$+$	$\pm$	$+$	$+$

4. The New Tracking Algorithm

Similarly to most of the existing tracking algorithms, the IPAN Tracker assumes that a point can only belong to a single trajectory. An important additional assumption is that an *upper estimate for the maximum speed*, v_{max}, is given. Currently, the duration of an occlusion is limited to 2 frames, but this restriction can be lifted.

The algorithm is based on the idea of competing trajectories. Correspondences to the previous and the subsequent frames are established as the result of a competitive hypothesis testing process similar to the one applied in the original two-frame matching procedure [2]. The basic difference is that three consecutive frames are now used to obtain the two hypothetic displacement vectors needed to form a smoothness-based cost function.

The matching algorithm has the following three steps: the initialization, the processing of the subsequent frames, and the post-processing. The initialization procedure operates on the first three frames and induces the tracking process. The points in F_2 are linked to the corresponding points in F_1 and F_3. Starting from F_3, a different matching procedure is applied. When all frames have been matched, a post-processing procedure is used to reconsider the points that temporarily disappeared and later re-appeared. This procedure attempts connecting the corresponding endpoints of the broken trajectories. The three steps are described below in more detail.

4.1. Initialization

The initializing step of the algorithm is illustrated in Fig. 1. Consider an arbitrary point $P_{2,i}$ in the frame F_2 and try to find the corresponding points in F_1 and F_3. Using the maximum speed constraint, project the location of $P_{2,i}$ onto F_1 and F_3 and determine the *search areas* of $P_{2,i}$ in F_1 and F_3 as those regions that may contain the candidate matching points for $P_{2,i}$. Denote by $S^-_{k,i}$ and $S^+_{k,i}$ the search areas of $P_{k,i}$ in F_{k-1} and F_{k+1}, respectively. The radius of these circular areas is defined by the maximum speed v_{max}.

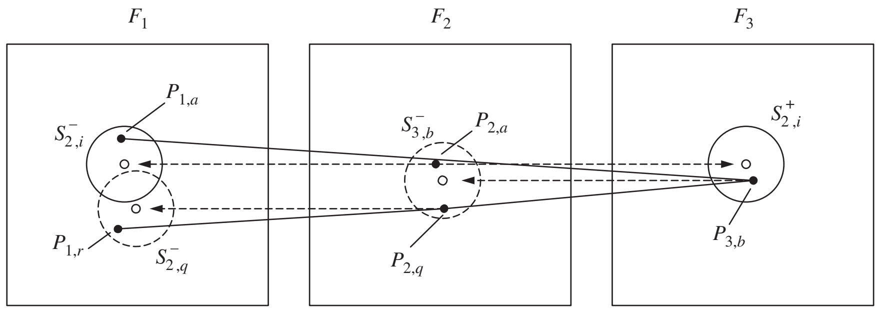

Figure 1. Initializing the tracking procedure. The filled circles are the moving points, the empty circles their locations projected to the neighboring frames. The solid lines show the competing trajectories. The dashed lines illustrate the formation of the search areas

Then consider all possible triplets of points $(P_{1,n}, P_{2,i}, P_{3,m})$, $P_{1,n} \in S^-_{2,i}$, $P_{3,m} \in S^+_{2,i}$, that contain the point $P_{2,i}$. Find the triplet $(P_{1,a}, P_{2,i}, P_{3,b})$ that minimizes for $k = 2$ the *cost function* $\delta(P_{k-1,n}, P_{k,i}, P_{k+1,m})$.

A cost function evaluates the local deviation from smoothness and penalizes changes in both direction and magnitude of the velocity vector. Its value is normalized so as to span the range [0, 1], with 0 meaning no change at all. Different cost functions implemented and tested with the proposed algorithm are defined in Section 4.4. The IPAN Tracker uses the *cost limit* $\delta_{max} \in [0, 1]$ as a smoothness threshold to discard obviously wrong links. (The setting of δ_{max} is discussed in Section 5.)

Select the triplet $(P_{1,a}, P_{2,i}, P_{3,b})$ as the initial hypothesis and rank the remaining triplets according to their cost values. (The triplets with $\delta < \delta_{max}$ are only considered.) Test the initial hypothesis by scanning $S^-_{3,b}$, the search area of $P_{3,b}$ in F_2. Let $P_{2,q}$ be a point in $S^-_{3,b}$ (see Fig. 1). If $S^-_{2,q}$ has a point $P_{1,r}$ such that $\delta(P_{1,r}, P_{2,q}, P_{3,b}) < \delta(P_{1,a}, P_{2,i}, P_{3,b})$, then the initial hypothesis is rejected and the second ranking hypothesis is considered and tested. Otherwise, the testing of the initial hypothesis proceeds with checking $P_{1,a}$ in the same way. If this check is also successful, $(P_{1,a}, P_{2,i}, P_{3,b})$ is output as the initial part of the trajectory of $P_{1,a}$. The correspondences established are not altered during further processing.

If all the hypotheses for $P_{2,i}$ have been rejected, this point is not linked to F_1 and F_3. It is still possible that $P_{2,i}$ will be linked to F_3 when the latter is processed. In such case $P_{2,i}$ *appears* and opens a trajectory.

When all points of F_2 have been processed, some points in the neighboring frames may remain unmatched. A point in F_1 that is not linked to any point in F_2 *disappears,* that is, either leaves the view field or temporarily disappears within the image, e.g., due to occlusion. An unmatched point in F_3 may later open a trajectory.

In the above description of the hypothesis testing procedure, the original hypothesis is rejected immediately when a 'stronger' triple $(P_{1,r}, P_{2,q}, P_{3,b})$ is found. However, $(P_{1,r}, P_{2,q}, P_{3,b})$ can itself be overruled by still another triple if the verification proceeds by testing $(P_{1,r}, P_{2,q}, P_{3,b})$ in a similar way, that is, by switching back to F_2 and F_3. The original hypothesis can possibly be restored. This deeper verification implies a longer sequence of simultaneous events whose probability decreases rapidly with the *verification depth* N_{ver}. The version described above is $N_{ver} = 1$. Currently, we use $N_{ver} = 2$ which is sufficient for point sets of reasonable density. (See Section 4.5 for a discussion.)

4.2. Processing the Subsequent Frames

In each frame, a point may have at most two links, a 'forward' one and a 'backward' one. A link indicates that the point is connected to a neighboring frame. Each link has a displacement vector assigned to it.

The matching procedure for $k > 2$ also operates with 3 consecutive frames. Consider the current frame $F_k, k > 2$. The previous frame F_{k-1} has just been processed. In F_{k-1}, the zero-link points (Z-points) are the appearing points whose potential correspondences to F_k are to be established. All the single-link points are connected to F_{k-2}, that is, they are backward-linked points abbreviated as B-points. These points have been marked as disappeared. They are only considered in the post-processing step. Most points usually have both links indicating continuous trajectories. In F_k, a point can have one backward link or no link at all. In F_{k+1}, all points are free (see Fig. 2).

The feature points in F_k are processed similarly to the initializing step described above. The only difference is that the already established correspondences are used when available. They are not modified. Consequently, during the hypothesis testing the B-points of F_k supply their previous displacements, while the Z-points are projected backwards onto F_{k-1} to find their candidate displacements. In F_{k-1}, the Z-points are only considered. These points may get linked to F_k and become

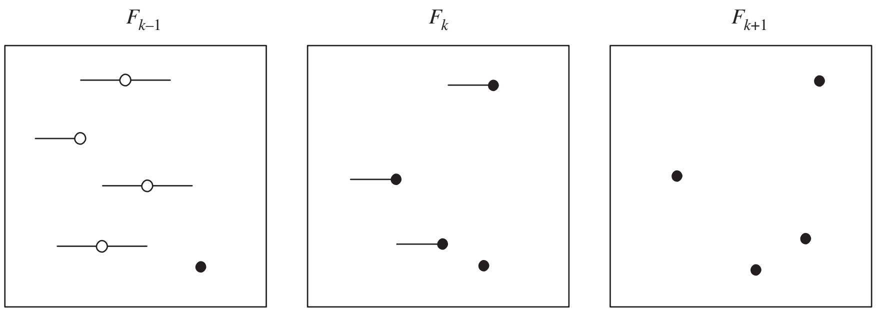

Figure 2. Processing the frame $F_k, k > 2$. The lines show the symbolic backward and forward links. Only the dark points are considered in the hypothesis testing

forward-linked (F-) points.

This procedure provides a natural way to handle appearing and disappearing points, including the motion across the image border. The moving points establish their links in a competitive process that develops as the trajectories grow. When the final frame has been processed, the double-link points form the continuous trajectories. The B-points are the disappearing, the F-points the appearing points.

4.3. Post-Processing of the Broken Trajectories

This procedure attempts connecting the broken trajectories. In Fig. 3, a broken trajectory is shown along with two possible continuations and a separate, continuous trajectory. Consider a B-point P_e with the incoming velocity $\vec{v_e}$ and an F-point P_s with the outgoing velocity $\vec{v_s}$. A candidate occluded point is searched in the intersection of the two search areas S_e^+ and S_s^-.

The search areas are basically defined by the cost function, the cost limit and the speed limit. In addition, the procedure makes use of the fact that *simultaneous* occlusion and drastic turn are rare, since both are relatively rare events. To ensure directional continuity of broken trajectories, $S_e^+ \cap S_s^-$ is more constrained than it is prescribed by the cost limit alone. The search areas for the particular cost function used in the IPAN Tracker are given in Section 4.4.

If $S_e^+ \cap S_s^-$ is empty, the trajectory remains broken. Otherwise, the point is found that minimizes the cost for the interpolated trajectory. This is done by exhaustive search of $S_e^+ \cap S_s^-$ with a suitable spatial resolution. To account for possible two-frame occlusions, each point of S_e^+ (S_s^-) is expanded in the same way into a search area in the next (previous) frame, and the average cost is minimized.

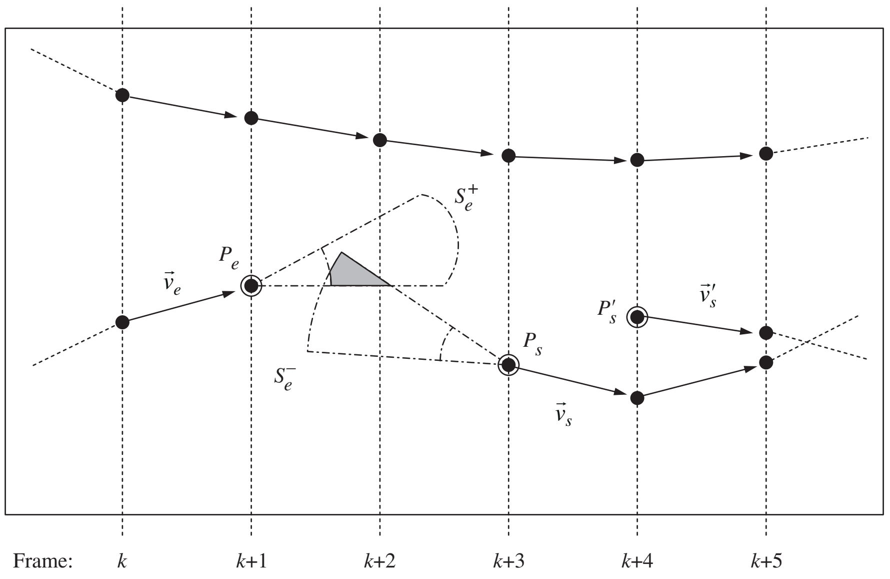

Figure 3. Post-processing of broken trajectories. A broken trajectory is shown along with two alternative continuations and a separate, continuous trajectory. The occluded point is searched in the filled region

4.4. Cost Functions

The IPAN Tracker uses the cost function introduced by Sethi and Jain in [8]:

$$\delta_o(P_{k-1,n}, P_{k,i}, P_{k+1,m}) = w_1 \left(1 - \frac{\overline{P_{k-1,n}P_{k,i}} \cdot \overline{P_{k,i}P_{k+1,m}}}{\left\|\overline{P_{k-1,n}P_{k,i}}\right\| \cdot \left\|\overline{P_{k,i}P_{k+1,m}}\right\|}\right) + w_2 \left(1 - \frac{2\left[\left\|\overline{P_{k-1,n}P_{k,i}}\right\| \cdot \left\|\overline{P_{k,i}P_{k+1,m}}\right\|\right]^{\frac{1}{2}}}{\left\|\overline{P_{k-1,n}P_{k,i}}\right\| + \left\|\overline{P_{k,i}P_{k+1,m}}\right\|}\right) \quad (1)$$

Here $\overline{P_{k-1,n}P_{k,i}}$ and $\overline{P_{k,i}P_{k+1,m}}$ denote the vectors pointing from $P_{k-1,n}$ to $P_{k,i}$ and from $P_{k,i}$ to $P_{k+1,m}$, respectively.

The first term in (1) penalizes changes in the direction, the second in the magnitude of the speed vector. By default, the first term is set to zero if any of the two vectors is zero.

The weights w_1 and w_2, $w_1 + w_2 = 1$, are used to balance the two components of the cost function. Salari and Sethi [7] and Hwang [4] also use similar cost functions. In IP97, SJ87 and SS90, $w_1 = 0.1$ and $w_2 = 0.9$, while in HW89 a different setting is used: $w_1 = 0.4$, $w_2 = 0.6$, and the cumulative cost over five frames is considered. We experienced that the original weights are slightly better; however, this decision does not seem to be critical.

The cost function by Rangarajan and Shah [6] was also tried out. This sophisticated function accounts for the distributions of accelerations and displacements. It can only be applied in the special case of known initial correspondences:

$$\delta(P_{k-1,n}, P_{k,i}, P_{k+1,m}) = \frac{\left\| \overline{P_{k-1,n}P_{k,i}} - \overline{P_{k,i}P_{k+1,m}} \right\|}{\sum\limits_{P_p \in F_{k-1}} \sum\limits_{P_q \in F_{k+1}} \left\| \overline{P_{k-1,p}P_{k,\phi_{k-1}(p)}} - \overline{P_{k,\phi_{k-1}(p)}P_{k+1,q}} \right\|} + \frac{\left\| \overline{P_{k,i}P_{k+1,m}} \right\|}{\sum\limits_{P_p \in F_{k-1}} \sum\limits_{P_q \in F_{k+1}} \left\| \overline{P_{k,\phi_{k-1}(p)}P_{k+1,q}} \right\|}, \tag{2}$$

where $\phi_k : F_k \rightarrow F_{k+1}$ is the mapping function that links the two frames.

While testing δ_R , we observed that it performed consistently better when its second term was discarded. The explanation is that the second term prefers small displacements typical for the tests done in [6]. This preference is not used by other authors as it is not realistic in many potential applications. Our test dataset has a wide range of displacements, including the large ones for which δ_R is not favorable. In the experiments, we used the modified version of (2) with the second term removed.

Once the cost function (1) has been selected, the post-processing search area S_e^+ introduced in Section 4.3 (Fig. 3) is obtained by separately applying the cost limit δ_{max} to the two terms of the cost function. The first term then limits the direction as $\theta \in [\theta_1, \theta_2]$; the second term, combined with the speed limit v_{max}, constrains the speed as $v \in [v_1, v_2] \cap [0, v_{max}]$.

As discussed in Section 4.3, less deviation from smoothness is allowed for a hypothetical occluded point than for an actually observed point. Mathematically, directional continuity of broken trajectories is enforced by limiting in (1) the *unweighted* rather than weighted terms. Under these constraints, it is easy to derive that

$$\theta_{1,2} = \theta_e \mp \arccos(1 - \delta_{max}), \tag{3}$$

$$v_{1,2} = \frac{v_e \left(1 \mp \sqrt{\delta_{max}(2 - \delta_{max})}\right)^2}{(1 - \delta_{max})^2}. \tag{4}$$

Here θ_e is the direction, v_e the magnitude of the incoming velocity vector $\vec{v_e}$.

From (3), the maximum allowed direction change for broken trajectories is $\pi/2$, when $\delta_{max} = 1$. When $\delta_{max} = 0.5$, the turn limit is $\pi/3$. $0 \leq v_1 \leq v_e$, while $v_2 \geq v_e$. When δ_{max} is set close to 1, $v_1 \approx 0$ and $v_2 \gg v_e$.

Expressions for S_s^- are obtained in the same way.

4.5. Parameter Settings

The IPAN Tracker has three basic parameters to be set: the speed limit v_{max}, the verification depth N_{ver}, and the cost limit δ_{max}.

The upper limit of the speed v_{max} is a critical parameter: the closer the estimate the better the performance, in both tracking quality and processing time. If v_{max} is smaller than the actual maximum speed, some links will be lost, as they fall outside the search area. Setting v_{max} to a 'safe' large value is not a good solution either, as the search area looses its selective role: the processing time will grow, together with the probability of false links.

As it was experienced earlier for the original algorithm [2], the verification depth $N_{ver} = 2$ brings a significant improvement in the performance compared to $N_{ver} = 1$. Further increase of N_{ver} has only minor influence on the tracking accuracy, while the computational load increases. Verification depths $N_{ver} > 2$ should only be considered in the case of extremely dense point sets.

The optimal value of the cost limit δ_{max} depends on the character of motion. For smooth motion, δ_{max} can be set relatively low; to allow for more drastic turns or speed changes, it should to be set closer to 1. Currently, we use $\delta_{max} = 0.6$; however, this setting does not seem to be too critical.

Two additional, 'hidden' parameters specify the spatial resolution of the search for occluded points in the post-processing procedure (see Section 4.3 and Eqs. (3) and (4)). We use $\Delta\theta = 10^\circ$ and $\Delta v = 1$ pixel as fixed default values.

5. Tests

A systematic experimental study [9, 10] is in progress aimed at comparative and quantitative performance evaluation of feature point tracking algorithms. Since the alternative algorithms assume different admissible events, they cannot be compared directly one to another. The IPAN Tracker which handles all events serves as the reference to which the other algorithms are compared under their own conditions. The comparison is done for a variety of cost functions embedded in the proposed algorithm and for different performance evaluation criteria. For a detailed presentation of the current evaluation results of this ongoing research the reader is referred to [9, 10]. In this paper, we only present some of these results to demonstrate the efficiency of the IPAN Tracker under different motion conditions.

Two efficient algorithms were selected from the four alternative techniques being considered in the systematic study. The algorithm by Rangarajan and Shah [6], RS91, is the most restrictive in admissible events (see Table 1). No entry and exit are allowed, and the initial correspondences must be given. The technique by Salari and Sethi [7], SS90, is the least restrictive among the alternative algorithms, as only occlusion is not handled. The most general case of all admissible events is also considered.

Results of comparison of IP97 with SJ87 are also briefly discussed, as SJ87 offers very good trajectory-based performance, within limited scope and at high computational cost. Comparison with HW89 is not discussed here because this algorithm appears to be definitely inferior to the other four.

Beyond the comparative performance evaluation, we seek to answer a more general question: how justified is the claim [6] that incomplete trajectories can be reliably tracked only if the initial correspondences are given?

5.1. Experimental Setup and Evaluation Criteria

A motion model called Point Set Motion Generator (PSMG) was used to generate synthetic motion sequences for testing the tracking algorithms. Let us briefly overview and justify the PSMG whose detailed description is given in [10].

The randomly generated points move independently. The advantage of this solution is that different types of events and trajectory ambiguities appear with statistically controllable frequencies. In fact, the PSMG is a generator of these events and ambiguities as *disturbances* which efficiently 'test' the algorithms.

Later, simulations of assorted coherent motions will be added to test the capability to cope with correlated trajectories. Real motion sequences and detected feature points will also be considered. We believe that the disturbances that make the algorithms err will essentially remain the same. However, we understand that the occurrence probabilities of various disturbances depend on the test data. Since the algorithms may be sensitive to different disturbances in different ways, one should be cautious when interpreting the error rates presented in alternative experimental studies, including ours. Keeping this in mind, we still believe it is possible to assess the performances of the tracking techniques using the PSMG.

Each test sequence has $M = 20$ frames. The size of a frame is 200×200. In F_1, initial points are located randomly; random initial motion directions and Gaussian-distributed speeds with the mean v are assigned to them. In the subsequent frames, a limited Gaussian perturbation of the velocity vectors is introduced. The speeds are limited by $v_{max} = 2v$. The probability and the maximum duration of occlusions are specified. (The results shown below were obtained for single-frame occlusions.) Motion across the image borders is generated by applying the PSMG to a large area enclosing the view field. Figure 4 shows sample trajectories generated by the PSMG for different speeds and numbers of points.

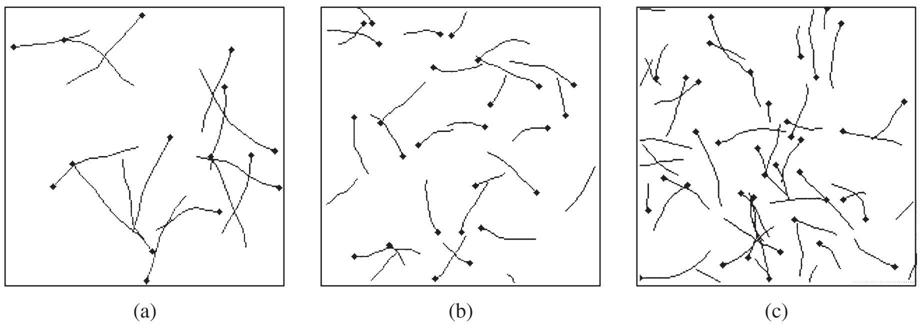

Figure 4. Examples of trajectories generated by the PSMG in 7 frames. The spots are the positions in the last frame. **a** $N = 15$, $v = 12$; **b** $N = 30$, $v = 6$; **c** $N = 50$, $v = 6$

The performance of the algorithms is evaluated by repeatedly generating trajectories with PSMG and comparing the tracking results to the ground truth. The varying parameters are the total number of trajectories T and the mean speed v. (Recall that T is equal to the total number of points in a sequence.) The generation of a trial data set is repeated until the desired T is obtained. For each setting of the varying parameters, 100 trials are done with independently generated data.

Three *merits of tracking performance* are used in our systematic study. The strict trajectory-based merit, whose experimental values are presented in Section 5.2, only accepts perfect trajectories and is defined as

$$M_{traj} = \frac{T_{corr}}{T}, \tag{5}$$

where T_{corr} is the number of perfectly tracked trajectories, T the total number of trajectories generated in the sequence.

The relaxed trajectory-based merit allows for local deviations from the ideal trajectory if the last point is connected to the correct initial point. The link-based merit accounts for the partially tracked trajectories. (A correct link is a vector that connects the same two points of the two consecutive frames as the ideal trajectory.) These two merits are normally consistent with M_{traj}, although the figures are different. The reader is referred to [9, 10] for tracking results based on these merits.

5.2. Test Results

Tables2 and 3 summarize the tracking results obtained by IP97, RS91 and SS90 for the strict trajectory-based merit defined in (5). Each row shows M_{traj} for varying total number of trajectories and two different mean speeds, low and high, which are 3 and 12 pixels per frame, respectively.

The first three columns of the tables specify the algorithm and the conditions of the evaluation, in the way consistent with Table 1. In particular, '$-$' in the 'Si.' column means that no self-initialization was done, that is, the initial correspondences were given.

Two essentially different data sets were used in the evaluation. The sequences used to obtain Table 2 contain no entry and exit, while the sequences generated for Table 3 contain both. It is important to emphasize that the results for these two data sets should not be compared directly. The reason is that the same number of trajectories in the two data sets does not mean the same average point density: when the points are allowed to exit and enter, many of the trajectories are short. When no entry and exit are allowed, a generated trajectory is only accepted if it extends through the whole sequence, with a few possible occlusions. Higher average point density leads to more frequent groupings and ambiguities, especially in the middle of the image. This difference in density for the same T grows with the mean speed.

Table 2. Tracking results for IP97 and RS91, without entry / exit

Alg.	Si.	Occl.	Low speed			High speed		
			$T = 20$	$T = 40$	$T = 60$	$T = 20$	$T = 40$	$T = 60$
IP97	$+$	$+$	95.35	92.07	89.98	82.70	67.55	54.56
IP97	$+$	$-$	95.60	92.92	91.20	85.70	74.02	62.45
IP97	$-$	$+$	97.50	94.10	92.28	83.95	72.05	61.26
RS91	$-$	$+$	98.50	95.52	94.58	81.15	59.47	16.61

Table 3. Tracking results for IP97 and SS90, without entry / exit

Alg.	Si.	Occl.	Low speed			High speed		
			$T = 20$	$T = 40$	$T = 60$	$T = 20$	$T = 40$	$T = 60$
IP97	$+$	$+$	95.02	94.18	90.51	79.45	68.78	58.82
IP97	$+$	$-$	95.77	95.64	92.16	87.92	79.06	71.17
SS90	$+$	$-$	95.87	95.39	92.75	85.35	72.33	57.20

The first two rows of Table 2 refer to IP97 in its standard self-initializing mode. One can see that the negative impact of occlusion on the tracking performance grows with v and T. In the third row, the proposed algorithm is given the initial correspondences. Apparently, the initialization can compensate for the negative effect of occlusion, since the corresponding values in the second and the third rows are quite close. Note that the differences between the first and the third rows only become significant for high speeds and many trajectories. Otherwise, self-initialization is possible and sufficient.

The third and fourth rows show that the performance of RS91 is close to, or even slightly better than, that of IP97 for low speeds and/or sparse point sets. However, RS91 performs poorly at large speeds and medium-to-high densities.

Table 3 demonstrates that the proposed algorithm performs reasonably well in presence of all events considered. Occlusion has the same impact here as in the previous case. Again, the alternative technique (SS90) can only compete with the IPAN Tracker until the motion conditions become severe.

Figure 5 compares the IPAN Tracker to SJ87. In this case, no entry and exit were allowed, as the algorithm SJ87 cannot handle these events. Both strict and relaxed trajectory error plots are shown. (For better visibility, error rates are given here instead of tracking merits.) The performance of SJ87 is somewhat better, with the difference decreasing with the number of trajectories. The price of this difference is the extremely high computational cost of the iterative SJ87, which grows dramatically and becomes prohibitive when the number of trajectories exceeds 45.

The processing speeds of the non-iterative algorithms HW89, RS91 and IP97 are similar, while the iterative SS90 is approximately 10 to 100 times slower, depending on the number of trajectories.

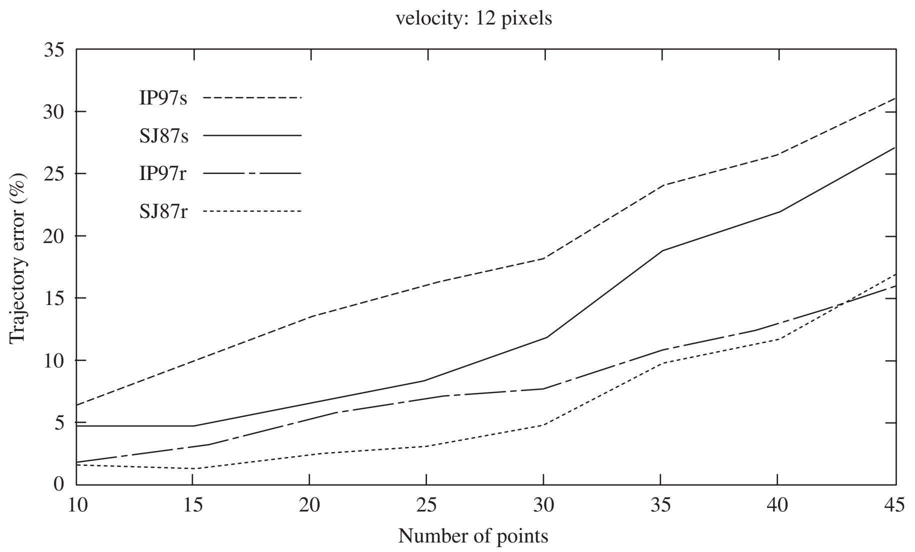

Figure 5. Error plots for IP97 and SJ87 for high speeds, without entry/exit. ‘s’ is the strict, ‘r’ the relaxed trajectory tracking merit

5.3. Application to Particle Velocimetry

An application of the proposed tracking algorithm is demonstrated in Fig. 6. Particles in a blood flow were tracked with the purpose of measuring their velocities. In particle velocimetry [5], a velocity field should be computed to adequately represent motion in all parts of flow. There are hundreds of particles in a sequence. Due to varying visibility, they often disappear and appear again. Entries and exits are also frequent. Partial trajectories can only be tracked, which is sufficient to obtain a velocity field. Here, the proper merit of tracking performance is the link-based one, as discussed in Section 5.1.

From Table 1, it is obvious that the tracking algorithms SJ87 and RS91 are unsuitable for this task, since they cannot handle flow across image border. SS90 cannot handle occlusions, which is of no particular importance, as partial trajectories are sufficient. However, SS90 cannot be applied to such a large number of points because of the prohibitive computational cost of this iterative algorithm. Therefore, HW89 and IP97 remain the only two candidates for this velocimetry application.

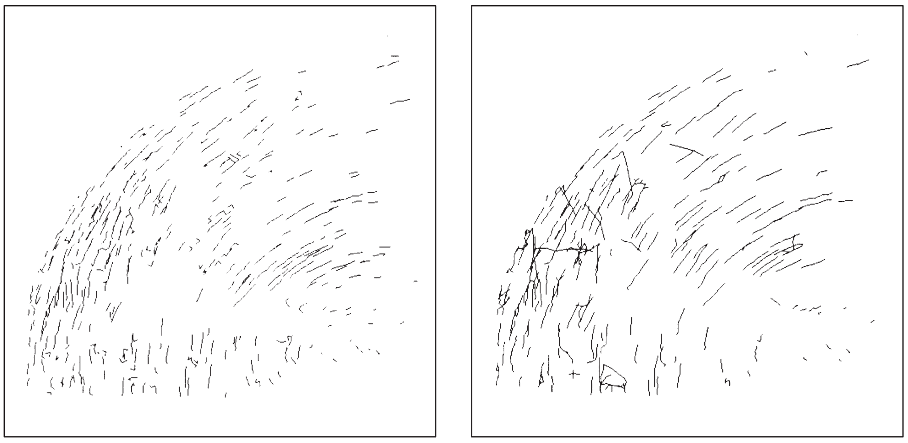

Figure 6. Blood flow tracking results by IP97 (left) and HW89 (right)

Blood flow is *coherent motion*, in which spatially close particles are likely to have similar velocity vectors. To enhance the velocity estimation, coherence based post-processing was applied to the tracking results of the two selected algorithms. The post-processing discards those velocity vectors whose orientation is incompatible with the dominant orientation of the surrounding vectors. The dominant orientation is the median in a window centered on the origin of the vector being considered. Incompatible outliers are detected as falling outside the limits specified by the median and the double variance.

Figure 7 shows the results of the coherence filtering of the original trajectories displayed in Fig. 6. The image size is 512 $\times$ 512, the window size 100 $\times$ 100 pixels. After coherence filtering, very short trajectories (less than 3 frames) were discarded. The IPAN Tracker yields more trajectories, and these are more coherent than those obtained by HW89. The latter provides longer connected trajectories, since this algorithm can cope with longer occlusions than IP97.

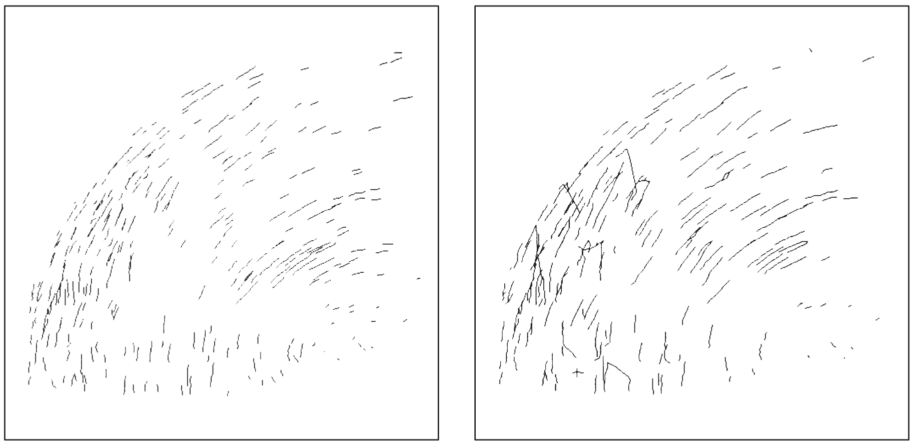

Figure 7. Enhanced results by IP97 (left) and HW89 (right) after coherence filtering. (Compare to Fig. 6)

6. Conclusion

A new feature point tracking algorithm has been presented. The algorithm performs reasonably well for dense feature point sets and large displacements. It efficiently resolves ambiguities arising from point groupings, temporary occlusions and motion across the border of the view field. The IPAN Tracker has been validated by systematic experiments at different motion conditions. Comparative performance evaluation of the new algorithm and several alternative techniques indicates that the IPAN Tracker is a good choice for uncorrelated trajectories, high speeds and high point densities.

The feature point motion tracking algorithms described in this paper are available for testing over the Internet at the following URL:

http://visual.ipan.sztaki.hu/psmweb/psmweb.html

Two testing options are provided. One can generate a synthetic sequence by the Point Set Motion Generator described in Section 5.1 and run a tracking algorithm on this sequence. Alternatively, one can submit his own motion file and obtain the tracking results. This gives a potential user a possibility to select an algorithm that fits his application.

At the same web site, results of our comparative performance evaluation study are presented in detail.

Acknowledgements

This work is supported by grants OTKA T026592 and INCO-COPERNICUS CT96-0742. The authors are grateful to Gábor Pécsy, László Szűcs, Attila Rácz, Krisztina Viktor, András Erdei and Márton Fülöp for their help in implementing the tracking algorithms.

References

[1] Jähne, B.: Digital image processing. Berlin Heidelberg New York Tokyo: Springer, 1997.
[2] Chetverikov, D., Lerch, A.: A matching algorithm for motion analysis of dense populations. Pattern Rec. Lett. *11*, 743–749 (1990).
[3] Courtney, J. D.: Automatic video indexing via object motion analysis. Pattern Rec. *30*, 607–625 (1997).
[4] Hwang, V.: Tracking feature points in time-varying images using an opportunistic selection approach. Pattern Rec. *22*, 247–256 (1989).
[5] Okamoto, K., Nishio, S., Saga, T., Kobayashi, T.: Standard images for particle imaging velocimetry. http://sap.gen.u-tokyo.ac.jp/image-e.html.
[6] Rangarajan, K., Shah, M.: Establishing motion correspondence. CVGIP: Image Understand. *54*, 56–73 (1991).
[7] Salari, V., Sethi, I. K.: Feature point correspondence in the presence of occlusion. IEEE Trans. Pattern Anal. Mach. Intelligence *12*, 87–91 (1990).
[8] Sethi, I. K., Jain, R.: Finding trajectories of feature points in a monocular image sequence. IEEE Trans. Pattern Anal. Mach. Intell. *9*, 56–73 (1987).
[9] Verestóy, J., Chetverikov, D.: Comparative performance evaluation of four feature tracking techniques. In: Proc. 22nd Workshop of the Austrian Pattern Recognition Group, pp. 255–263. Österreichische Computer Gesellschaft, 1998.
[10] Verestóy, J., Chetverikov, D.: Experimental comparative evaluation of feature point tracking algorithms. In: Proc. Workshop on Evaluation and Validation of Computer Vision Algorithms. Kluwer series in Computational Imaging and Vision, 1998 (submitted).

Dmitry Chetverikov, Judit Verestóy
Image and Pattern Analysis Group
Computer and Automation Research Institute
Budapest, Kende u.13-17, H-1111 Hungary
{csetverikov,verestoy}@sztaki.hu