A Matching and Tracking Strategy for Independently Moving Objects

Larry S. Shapiro, Han Wang and J. Michael Brady

Robotics Research Group,
Department of Engineering Science,
19 Parks Road, Oxford University, OX1 3PJ, U.K.

Abstract

We present a robust and inherently parallel strategy for tracking "corner" features on independently moving (and possibly non–rigid) objects. The system operates over long, monocular image sequences and comprises two main parts. A *matcher* performs two–frame correspondence based on spatial proximity and similarity in local image structure, while a *tracker* maintains an image trajectory (and predictor) for every feature. The use of low–level features ensures an opportunistic and widely applicable algorithm. Moreover, the system copes with noisy data, predictor failure, and occlusion and disocclusion of scene structure. Motion and scene analysis modules can then be built onto this framework. The algorithm is aimed at applications with small inter–frame motion, such as videoconferencing.

1 Introduction

This paper addresses the problem of robustly extracting the image motion of independently moving (and possibly non–rigid) objects over long periods of time. The application motivating this research is model–based coding of facial image sequences [1], which requires fast, reliable motion estimation on lengthy monocular sequences. Here, we focus on the low–level "front–end" of such a system, and show that the use of general–purpose "corner" features enables us to cope with a wide range of facial variations and accessories (e.g., beards and glasses). The image trajectories that emerge are intended to drive "higher–level" modules, which will group features into coherently moving objects and estimate their 3D motion. Further details of the research described here can be found in [6].

We build on recent work at Oxford by Wang and Brady [11], who developed a "corner" finder that runs at 14Hz on T800 transputers. These "corners" are curvature extrema in the image intensity surface, and have already served successfully as features in a stereo–matching algorithm [10]. We extend this work to the general motion case, which introduces both benefits and complications: the former because temporal integration facilitates noise resistance and allows ambiguity to be resolved in time, and the latter because objects can change over time in ways they can't over space alone. As in the DROID system [3], we employ a single tracker and predictor per feature.

Our framework has two parts: a *matcher,* to perform two–frame correspondence, and a *tracker,* to maintain the trajectories and perform prediction. The utility of corners as correspondence tokens is demonstrated in Section 2, and their extraction mechanism described. The matcher and tracker subsystems are then discussed separately in Sections 3 and 4, and we conclude with directions for future research. Results on real imagery are given throughout.

BMVC 1992 doi:10.5244/C.6.32

2 Corner detection

In order to match different views of an object, one must first obtain a set of reliable features from each view. (The explicit extraction of "correspondence tokens" was supported by Ullman [8] and further justified by Verri and Poggio [9].) We employ the term "corners" to refer to distinctive feature points such as discontinuities, points of occlusion, and various intensity curvature maxima (e.g., surface markings). These appear in the image as loci of two–dimensional intensity change (i.e., *second–order features*), and impose more constraint on visual processes than edges (which encode only one–dimensional change) [2]. This is particularly true of visual motion; various authors (e.g., [4]) have shown that the *full* optic flow field μ is recoverable at corner points, whilst only the normal component $\mu^{\perp}$ can be recovered locally along an edge (owing to the aperture problem). Furthermore, corner tokens are discrete and distinguishable, so can be explicitly tracked over time; arbitrarily curving edges are difficult to describe and hence to track [3].

To date, corners have mainly been used where there is an abundance of physical corners arising from man–made objects (e.g., factory environments). We contend, however, that corner features are equally useful in many natural scenes. In a human face, for instance, there are no right–angles or sharp discontinuities; facial features are rounded and the skin surface is smooth. Nonetheless, our experiments on many images [6] have shown that there *are* numerous "corner points" in the image of a human face (Figure 1). Often they correspond to salient anatomical features (e.g., nostrils or corners of the eyes), but more importantly, they are *stable, robust* beacons which can be tracked as the head moves.

Furthermore, by using corner features we gain the important advantages of *generality* and *opportunism.* Videoconferencing researchers in particular have largely overlooked this low–level approach, aiming directly for high–level facial features. (An exception was So et al. [7] who used centres–of–gravity of iso–density contours.) Our tracking system can

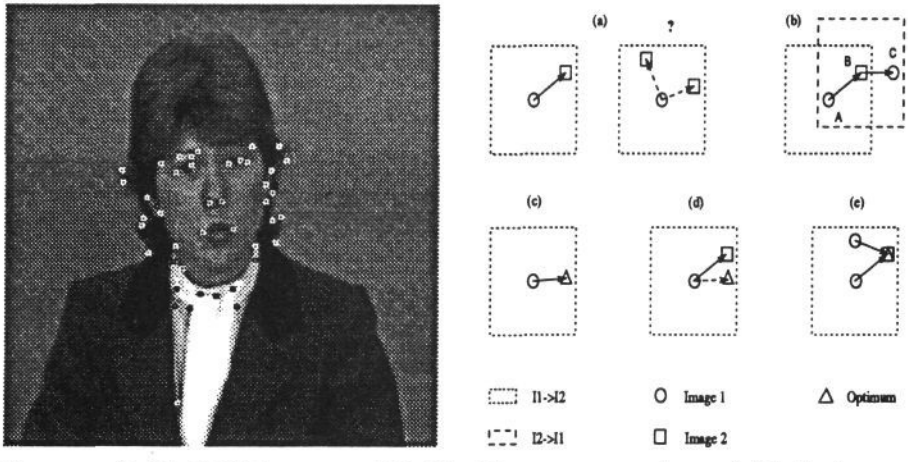

Figure 1: [Left] *CCITT sequence "Clair" with corners superimposed;* [a] *I_2 features (squares) falling in a search window centred on an I_1 feature (circle) are match candidates;* [b] *A "love triangle": feature A in I_1 chooses B in I_2, but B prefers C to A;* [c] *An unmatched I_1 feature finds a* ghost *match (triangle) by correlation search;* [d] *A nearby unclaimed feature in I_2 is accepted in a* compromise *match;* [e] *A ghost match coincides with a claimed corner, so the paths merge.*

cope with male and female subjects, glasses and eye–patches, plasters and nose–casts, and even facial hair (including beards, mustaches and sideburns). Moreover, it *uses* to its advantage any distinctive features that deviate from the norm, such as earrings, acne and facial scars. This opportunism applies both statically *and* dynamically; if the speaker dimples when smiling, or wrinkles his forehead when frowning, the system will utilise the corners while they are available.

Corners are detected using a *corner operator* [11], which involves only local operations and generates a response Γ which attains a local maximum at the corner location:

$$\Gamma = \left(\frac{\partial^2 I}{\partial t^2}\right)^2 - S\,|\nabla I|^2 \longrightarrow \max, \quad \Gamma > R, \quad |\nabla I|^2 > E. \tag{1}$$

There are four parameters to specify: scale S, edge strength E, corner response R and mask size M. Details appear in [11, 6], along with discussion on automatic parameter setting, the effects of Gaussian smoothing, and the problem of false corners (arising from profile edges, conjunctions of edges lying at different depths, shadow lines and specularities).

3 The matcher

The two–frame correspondence problem is familiar in computer vision. Seminal work in this area was done by Ullman [8], and numerous algorithms have since been proposed in an extensive body of literature. The demand for eventual frame–rate processing places constraints on our matcher: the algorithm must be parallel in nature and computationally cheap. The fact that inter–frame motion is small, however, ensures that the intensity pattern around a corner (its local *shape*) doesn't change much between frames.

The matcher receives as input two grey–scale images (I_1 and I_2) along with their respective sets of corners (having image coordinates $\mathbf{p}_{i,1}$ and $\mathbf{p}_{j,2}$ respectively). These corners are generated automatically as images enter sequentially in time. This section describes the two–frame matcher in the absence of predictions; the modifications when predictions are available are discussed in Section 4. An important feature of our matcher is that it leaves no corner in I_1 without a pairing, and uses only local operations.

3.1 Strong matches

Consider two sets of corners superimposed on the same system of image axes. For every corner in I_1, we construct a search window centred on it, and all corner points from I_2 lying in this window are candidates for the match (Figure 1(a)). We then perform a *local patch correlation* between the corner in I_1 (the "template") and each candidate corner in I_2 (the "patch"). The winning candidate is the one with the highest correlation value, provided it surpasses a certain minimum threshold; this is necessary since the "best" corner in the window need not be the correct one (e.g., the actual feature may have disappeared).

We then repeat the procedure, working back from the second image to the first. This widely–used technique (e.g., [3]) resolves conflicting attractions, where the preference of one feature for another is not reciprocated (Figure 1(b)). We only accept matches which concur in both directions, and discard the rest. A match that survives this pruning is a *strong* (or *natural*) match, with a confidence value c. The correlation metric we use is the standard *product moment coefficient*,

$$c = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{n}(t_i - \bar{t})^2}\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}}, \quad -1 \le c \le 1,$$

where $\{t_i\}$ and $\{p_i\}$ are the intensity values of the template and patch, $\bar{t}$ and $\bar{p}$ are their means, and we raster–scan pixels in the blocks of interest to give the two n–point data–sets. This metric is invariant to a linear change between the data–sets, i.e., $c = 1$ when $p_i = at_i + b$, with constants $a, b \geq 0$. Hence, c compares the *structure* of the patches, rather than their absolute intensities[1]. Only positive correlation values are considered; negative c indicates inversion of the intensity values. Since perfect correlation obtains when $c = 1$, c serves as a measure of confidence in the match.

3.2 Ghost and compromise matches

There are several reasons why there may still be unmatched corners:

1. A feature may permanently disappear from sight, due to occlusion. Hence, it will appear in I_1 but not in I_2 (a *ghost*).
2. A previously obscured feature may become visible as new structure sweeps into view (or previously–seen structure becomes visible once more). Hence, the feature will appear in I_2 but not in I_1 (an *intruder*).
3. A feature may appear intermittently ("flash" on and off) due to instability in the corner detector (e.g., the response Γ oscillates about the cutoff threshold). This can result in both ghosts and intruders.
4. A feature may appear once and then disappear, due to noise in the signal. This leads first to an intruder and then to a ghost.

Although it is impossible to distinguish these scenarios on the basis of only two frames (and it is precisely the last two problems which have made corner detection unattractive in the past), sustained observation of the features (coupled with prediction) makes this task simple.

It is, however, important that the matcher doesn't deprive the tracker of potentially useful information (which could always be overridden or discarded later). We therefore require the matcher to generate a best position estimate for *every remaining corner* in I_1. The assumption here is that the third scenario has happened, i.e., the corner has flashed off and will soon flash on again. If indeed this *has* happened, the "bridging" estimate will be good, since the feature is still visible and does have some second–order structure. If the assumption is incorrect and the feature has disappeared, the tracker will soon realise this (since the corner won't reappear and the ghost matches will be poor).

The best position estimate is computed via a correlation test over the whole search window, using every location as a candidate. The location with highest correlation (c_{max}) is accepted (Figure 1(c)). If there is a real corner close to the estimated position (with correlation c_{ok}), we accept it if it is sufficiently similar to c_{max} (Figure 1(d)). This is a *compromise* match, and its objective is to reduce the number of intruders and ghosts. If the ghost point coincides with an I_2 corner already claimed by a different point, we merge their paths, destroying the ghost and its trajectory (Figure 1(e)).

If there is no corner nearby the optimum correlation position, we settle for the *ghost* match. The new position assigned to the unmatched corner is thus decided purely by correlation, *with no actual corner being there*. On the next cycle ($I_2 \rightarrow I_3$), this ghost corner will be treated as a real corner in I_2, in the hope that it will find a strong match

[1] Assuming the albedo of the patch doesn't change over time, a accounts for automatic gain control of the camera, uniform changes in scene lighting and changes of object pose relative to a constant light source, while b accounts for a uniform intensity offset [6].

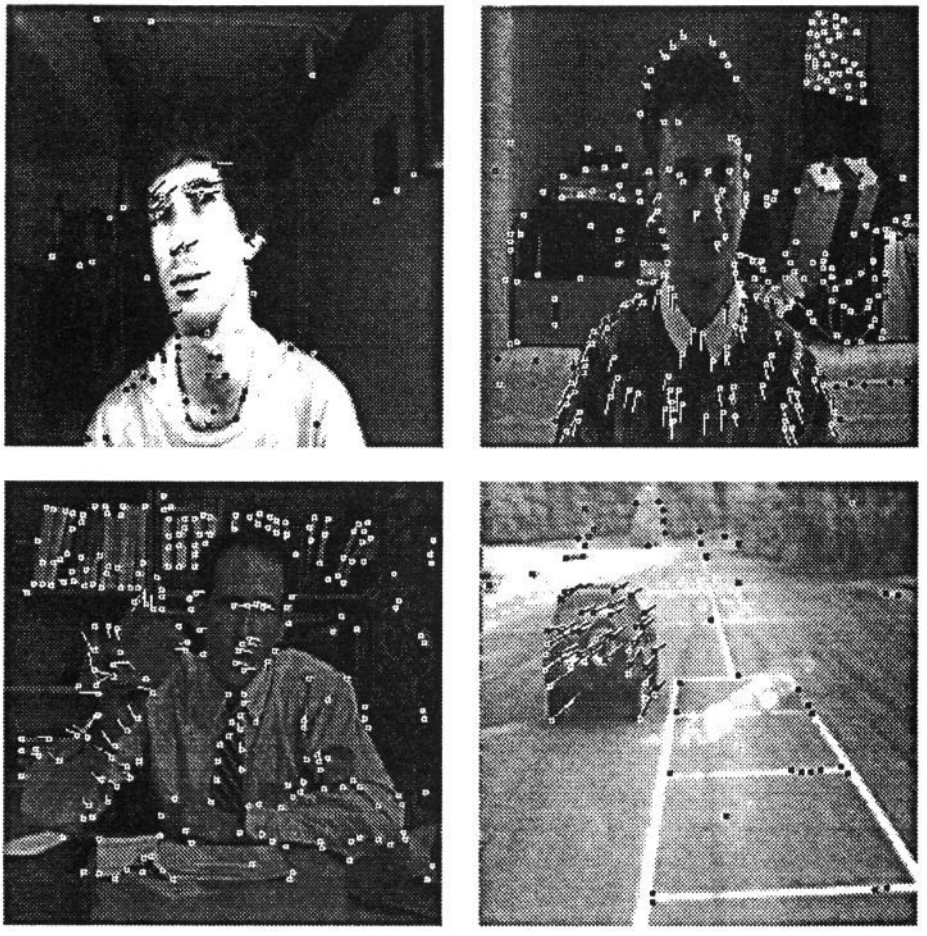

Figure 2: *Two-frame matches (markers indicate corners in the first frame, and vectors are drawn double their true length for clarity):* [top to bottom, left to right] *"Curl"; "Div"; "Salesman" (CCITT); "Car".*

in I_3 (signalling reappearance). We refer collectively to ghost, compromise and merge matches as *forced* matches.

This search procedure is expensive, since an $n \times n$ window yields n^2 possible positions. However, the operation can be done in parallel, and is performed only for the unmatched points. Moreover, n^2 is the worst-case scenario; when predictions are available, the search space is substantially reduced (see Section 4.2).

3.3 Results

The algorithm was implemented in sequential form in C and run on SUN SPARC-1 workstations. An 11 x 11 search window was used for initial candidates, and a 3 x 3 window for compromise matches(see [6]). We have tested this algorithm on a wide range of sequences, under different lighting conditions and facial poses.

Figure 2 shows several two–frame matches, combining the natural and forced pairings (camera is stationary). The "Curl" sequence shows a head (LSS) rotating about an axis parallel to the optic axis, while the "Div" sequence shows a subject looming towards the camera amidst a cluttered background (a diverging flow pattern). The CCITT "Salesman" sequence is a challenging one, for the speaker moves his arms, flexes his fingers, turns the object he is demonstrating and ripples his shirt. The corners accurately reflect this movement; his head moves left while his right arm moves right and upwards. Finally, we show a car accelerating forwards, indicating that the algorithm is transportable to other application domains. Similar results have been obtained when the camera moves as well as the scene (paper in preparation).

These results illustrate that the motion vectors give a clear indication of where in the image (and in what direction) movement occurs. This testifies both to the temporal consistency of the corners (strong and compromise matches), and to the suitability of corner locations for computing flow (ghost matches). The accuracy of the motion vectors despite the small motion indicates how well the corners are localised; this will prove a solid foundation for trajectories spanning multiple frames.

4 The tracker

The tracker has two responsibilities. Firstly, it maintains an image trajectory for each feature, charting its motion through successive frames. Secondly, it oversees the matcher, feeding it predictions and obtaining a set of matches in return. This also involves supervising the initial startup (*boot mode*) and securing the transition to normal operation (*run mode*).

4.1 Trajectory maintenance

Every feature has a record in the "world" database, describing its general details (e.g., when it first appeared) and its frame–specific information (e.g., position and velocity). Maintaining these spatio–temporal trajectories comprises various subtasks, e.g., *instantiate* new features, *retire* features which have disappeared, and *update* the records of tracked features. The number of tracked corners grows for the first few frames and then reaches an approximate equilibrium, once the instantiation of new points is offset by the retirement program.

4.2 Correspondence control strategy

When there are n frames in the sequence rather than just two, there are $n-1$ pairs of images to process in temporal order: I_1 and I_2, I_2 and I_3, etc. The algorithm in Section 3 operates until the predictor kicks in, whereafter the algorithm presented below is used. Hereafter, I_1 will refer to the "previous" image, and I_2 to the "current" image.

4.2.1 The prediction philosophy

Finding a suitable role for prediction in long–term tracking is a tricky problem. On the one hand, past behaviour can be a valuable indicator of future behaviour, since physical objects moving in the world build up inertia; to ignore these "motion trends" is therefore to discard useful information. On the other hand, predictors require a model of object motion: when the model is valid, the predictor works well, but when the model fails, do does the predictor (often badly). Typically, tracking systems match directly from predicted

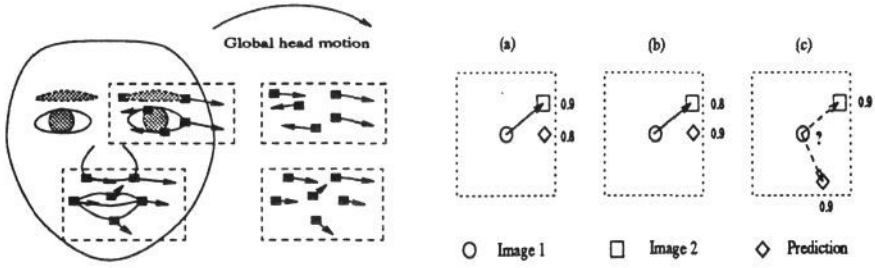

Figure 3: [left] *Valid facial motion patterns, which one might reject under the rigidity assumption (eyes move and mouth opens);* [right] *Strong matches versus predictions:* [a] *Match wins;* [b] *Prediction wins and is nearby, so we get a* compromise *match;* [c] *Prediction wins and is far away, so the corner is discarded (a* wimp*).*

positions to the data in the new frame. This approach only has merit when the prediction takes you *closer* to the true position; when it moves you further away, you are worse off than if you had used the *raw* data. Often, the problem of what to do when prediction fails is bypassed by using other cues [3], solved via adaptive filters [5], or even ignored.

However, the question is not *whether* to use the predictor, but *when* to use it; we need to distinguish between a predictor which is working well (and can be trusted) and a predictor which is failing (and should be restarted). Occasional predictor failure is inevitable; in videophony, for instance, heads can change direction rapidly (e.g., a nodding action). A mechanism for *graceful degradation* is thus of fundamental importance.

Our solution is to maintain a predictor (in image coordinates) for every corner, and use local patch correlation as an indicator of predictor success. When the predictor fails, we simply revert to the original image data. We use a simple model of feature motion, eschewing the use of (2D or 3D) global (or semi–global) motion models at this early stage of processing; such motion models can slot in at a higher stage. Thus, we avoid having to first segment the scene into differently moving objects; indeed, this is one purpose for which the trajectories are intended. Furthermore, we are able to track independent and non–rigid motions. Faces, for instance, are not only non–rigid themselves, but also *contain* non–rigid objects (e.g., the mouth). Consequently, the corners simply *don't* move in a locally consistent way (Figure 3).

Although our system has much in common with DROID [3] (which broke new ground in tracking corner features), we differ from it in several important ways. For example, DROID catered only for an observer moving through a static (hence rigid) world. A detailed comparison between the systems is provided in [6].

4.2.2 Strong matches

Predictions are initially held in abeyance while we obtain a set of mutually consenting matches from the raw data (Section 3.1). For every corner in I_1, we then compare its prediction against its strong match. If the strong match wins (Figure 3(a)), it is accepted as the correct solution. If the prediction is better than the match, then the course of action depends on how far apart these two locations are. If they are close together, we accept the corner but *downgrade* the match to a *compromise* (Figure 3(b)). If, they are far apart, we have an I_1 corner being strongly pulled by two very different image locations (Figure 3(c)). Such *wimp* corners arise in areas of uniform texture, caused either by a poor corner in a

region without much structure (e.g., a cheek), or a good corner in a highly (but similarly) textured area (e.g., chin stubble or hair). Either way, the corner is an unsuitable feature and is destroyed.

4.2.3 Compromise and ghost matches

Now the unmatched corners remain. If a prediction is available, we search a small region around the prediction and see whether the best correlation value there is good enough to be accepted. If not, we do a full correlation search starting from the original corner. Thus, when the predictor works, we save greatly on time, but when it fails, we revert to the full search method. The reason for not searching in a gradually expanding region around the prediction (e.g., radially) is that once the predictor has failed, a more accurate result obtains by ignoring it completely. Note that if the prediction correlates well but is incorrect, this becomes apparent in several frames' time when no match reappears, and the corner is retired.

A comparison (over a long sequence) between the cases with and without a predictor show interesting results. Firstly, the total number of corners being tracked is very similar in the two cases, and we also get almost identical matches; prediction simply *speeds up the process.* This differs from many prediction schemes where, in the absence of prediction, the number of unmatched points grows due to the large uncertainty. Because we force unmatched corners in I_1 to find "virtual" partners in I_2, we contain the uncertainty in the system. Secondly, the number of strong matches is fairly constant over time, suggesting that a fixed percentage of the corners are very robust [6].

4.2.4 Prediction strategies

We examined two fixed–coefficient predictors (constant velocity and constant acceleration). At best these model could only be approximate since they operate on *image* data, without modelling 3D motion (or camera projection). However, our goal here is *not* to deduce the world motion parameters, but rather to utilise trends in image motion to improve efficiency and reject unsuitable matches. For the constant velocity case (in finite difference notation), $\dot{x}(k+1) = \dot{x}(k)$, so $x(k+1) = 2x(k) - x(k-1)$. Two frames are needed for a prediction, and distance changes linearly in time. (We treat x and y coordinates independently.)

For the constant acceleration case, $\ddot{x}(k+1) = \ddot{x}(k)$, so $x(k+1) = 3x(k) - 3x(k-1) + x(k-2)$. Here, three frames are needed for prediction, and distance changes quadratically in time. Li et al. [5] also used these filters but favoured the adaptive coefficient forms. However, accurate tracking was crucial to their system since they matched from predictions to data; it is far less critical in our approach.

Experiments have shown that the "linear" predictor often outperforms the "quadratic" one, because the smallness of the inter–frame motion often leads to locally linear trajectories. Also, since the motion is small relative to the quantisation errors, the noise introduced by "second temporal derivatives" has a detrimental effect.

4.3 Results

Figure 4 shows image trajectories obtained over several frames. "Richard" (bespectacled) nods his head downwards, "Clair" moves her head round in a circular motion, and "Dave" (bearded) performs a "curl" motion while his body sways slightly in the opposite direction.

When the camera is still, it is simple to distinguish stationary from moving points on the basis of their velocity history. For each feature, we compute the mean ($\bar{s}$) and

standard deviation (σ_s) of its speed over several frames. Classification as a stationary point requires small $\bar{s}$ and small σ_s (with a minimum number of sightings). Figure 4 shows the segmentation for the "Dave" sequence, with the stationary points removed.

5 Conclusions

We have presented an algorithm to track moving objects in the image, using local operations. A key strength of this algorithm is the use of low-level "corner" features. These corners are stable and well-localised, making them suitable for tracking – even in applications where there aren't "physical" corners, such as human faces. Furthermore, being entirely image-driven, the corners differ from scene to scene, giving the powerful advantages of generality and opportunism. We further ensure robustness by means of temporal

Figure 4: *Image trajectories, each spanning an equal number of frames (true-length vectors, markers show corner positions in final frame):* [top to bottom, left to right] *"Richard" (8 frames); "Clair" (6 frames); "Dave" (6 frames); moving points for "Dave".*

integration, which overcomes the problems of "flashing" corners, noise, and occlusion and disocclusion of 3D structure.

Our matcher–tracker verifies matches by correlating local image structure, and limits uncertainty by assigning *ghost* matches to unmatched points. Use of a simple predictor (per point) speeds up the matching process significantly, though we take care to degrade gracefully when predictors fail (by reverting to the raw data). The image trajectories that finally emerge give a strong impression of "what motion occurs where".

The matcher–tracker forms only the first level of our motion analysis system, and there are several directions of research to pursue. Firstly, by forming clusters of corners having similar motion, we aim to segment the scene into coherently–moving objects and then compute 3D motion parameters. Secondly, as the "Salesman" sequence illustrates, edge information will be very useful for eliciting motion boundaries. We therefore plan to combine edge motion with corner motion. Thirdly, the usefulness of point features other than the ones we have described here (e.g., distinguished points from invariant theory) will be explored. Finally, we plan to implement the tracker in parallel.

Acknowledgments

LSS thanks Rami Guissin and Andrew Zisserman for useful discussions. Ian Reid, Dave Dyer and Richard Lewis willingly sat before our camera, and Bill Welsh (of British Telecom Research Labs) provided the CCITT sequences. LSS is supported by an ORS award (UK) and by the Foundation for Research Development (RSA).

References

[1] K. Aizawa, H. Harashima and T. Saito, "Model–based analysis synthesis image coding (MBASIC) system for a person's face", *Signal Processing: Image Communication*, Vol. 1, No. 2, Oct. 1989, pp. 139–152.

[2] J.M. Brady, "Seeds of perception", *Proc. 3rd Alvey Vision Conference*, Cambridge University, Sept 1987, pp. 259–265.

[3] D. Charnley, C. Harris, M. Pike, E. Sparks and M. Stephens, "The DROID 3D vision system: algorithms for geometric integration", Plessey Research, Roke Manor, Technical Note 72/88/N488U, Dec. 1988.

[4] L. Dreschler and H. Nagel, "Volumetric model and 3D trajectory of a moving car derived from monocular TV–frame sequence of a street scene", *Computer Vision, Graphics and Image Processing*, Vol. 20, No. 3, Nov. 1982, pp. 199–228.

[5] H. Li, P. Roivainen and R. Forchheimer, "3D motion estimation in model–based facial image coding", Report LITH–ISY–I–1278, Dept. Electrical Engineering, Linköping University, 1991.

[6] L.S. Shapiro, H. Wang and J.M. Brady, "A matching and tracking strategy applied to videophony", Report OUEL 1933/92, Dept. Engineering Science, Oxford, May 1992.

[7] I. So, O. Nakamura and T. Minami, "A study on a model–based coding system based on isodensity maps of facial images", *Picture Coding Symposium (PCS–91)*, 1991, pp. 299–302.

[8] S. Ullman, *The Interpretation of Visual Motion*, MIT Press, USA, 1979.

[9] A. Verri and T. Poggio, "Against quantitative optic flow", *Proceedings of the International Conference on Computer Vision (ICCV–1)*, London, UK, May 1987, pp. 171–180.

[10] H. Wang and J.M. Brady, "A structure–from–motion vision algorithm for robot guidance" in I. Masaki (ed.), *Proc. IEEE Symposium on Intelligent Vehicles*, Detroit, June 1992.

[11] H. Wang and J.M. Brady, "Corner detection with subpixel accuracy", Report OUEL 1925/92, Dept. Engineering Science, Oxford, 1992.