Algorithmic Fusion for More Robust Feature Tracking

Brendan McCane, Ben Galvin and Kevin Novins

Department of Computer Science
University of Otago
Dunedin, New Zealand

September 3, 2002

Abstract

We present a framework for merging the results of independent feature-based motion trackers using a classification based approach. We demonstrate the efficacy of the framework using corner trackers as an example. The major problem with such systems is generating ground truth data for training. We show how synthetic data can be used effectively to overcome this problem. Our combined system performs better in both dropouts and errors than a correspondence tracker, and had less than half the dropouts at the cost of moderate increase in error compared to a relaxation tracker.

1 Introduction

Image motion contains an extraordinary wealth of information: subtle variations in shape are mirrored by subtle variations in image motion, discontinuities in surface direction and depth are evidenced by equivalent discontinuities in the motion field, and camera ego-motion induces global motion in the image. Yet

despite the large body of literature on motion analysis, this potential remains largely unrealized and many useful algorithms remain confined to the laboratory. This is at least partly due to the difficulties in computing reliable motion estimates.

Errors in any motion-tracking system can arise from two different sources: measurement error and algorithmic failure. Measurement errors occur when a feature is tracked properly, but its new position cannot be determined with perfect precision. Measurement errors are well behaved in the sense that they usually can be modeled as having zero mean and normal distribution. They can therefore be largely accounted for by employing filtering techniques and propagating confidence values throughout the system (with the Kalman filter for example).

Algorithmic failures occur when the system generates an incorrect trajectory for a feature; correspondence mistakes are a typical example. These errors will usually have much more serious consequences on any subsequent processing because they are unbounded. Recovery is impossible using only local filtering. Errors produced by algorithmic failures are a major obstacle to the design of robust vision systems.

The easiest way of reducing algorithmic errors is to reject a larger portion of unlikely potential matches. This can be done by raising a threshold, for example. However this has the undesirable effect of increasing dropouts, thus limiting the number of frames across which any particular feature can be expected to be tracked. Algorithmic errors can also be reduced by improving individual tracking algorithms, but it is unlikely that any one algorithm will be successful in all situations.

Given the wealth of tracking algorithms already available, a more promising solution seems to be to combine the results of multiple tracking systems - a type of algorithmic fusion. Such an idea is not new. It has long been recognized that the human visual system uses sensor fusion to make sense of the world (Murphy, 1996), and combining the results of multiple classifiers has received considerable attention (Foggia, Sansone, Tortorella and Vento, 1999; Ho,

Hull and Srihari, 1994; Xu, Kryzyzak and Suen, 1992). What we propose in this paper are methods for applying both classification techniques and algorithmic fusion techniques to the problem of reliably tracking features in a video sequence. A major problem with such classification techniques is generating ground truth data for training the classifier. We demonstrate how easily generated synthetic data can be used to effectively overcome this problem. Our combined system performs better in both dropouts and errors than a correspondence tracker, and had less than half the dropouts at the cost of moderate increase in error compared to a relaxation tracker.

2 Previous Work

Existing tracking techniques fall into two major categories: global correspondence and local search. Smith and Brady (1995) employ global correspondence in their motion segmentation system, ASSET-2. All tracked corners from previous frames are compared to all candidate corners in the current frame using image color and spatial image gradients. Tracked corners are then assigned to candidate corners using a greedy algorithm that preserves a one-to-one correspondence between the tracked corners and candidate corners. The greedy algorithm terminates when the best remaining correspondence fails to exceed a user selected threshold. Users are forced to choose between being able to track a small number of very obvious features, or making a significant number of correspondence errors. When these errors do occur, they tend to be large and persistent.

Tomasi and Kanade's feature tracking algorithm is a typical example of a local search technique (Tomasi and Kanade, 1991). Their technique uses an exhaustive search in a neighbourhood to find the best correlation match which can be thresholded to reduce errors (and produce dropouts). Shi and Tomasi (1994) have modified this algorithm to allow for affine distortions of each patch, and Kang, Szeliski and Shum (1997) describe a similar correlation based technique that requires the patches to tile the image and allows bilinear distortions of each

patch. However, such correlation style trackers cannot perform well in areas of low contrast or in areas of occlusion.

Global search has distinct advantages over local search in some situations, and vice-versa. In order to combine the strengths of these two approaches, we need a robust method for determining if a tracker is reporting a correct match. However, any such scheme is likely to involve several parameters. It is our belief that arbitrarily setting such parameters is too complex and error-prone for practical application.

There has been other work on combining multiple sources of information. These can be broadly classified into four types of techniques depending on the multiplicity of data source and algorithmic method:

1. single data source and single algorithm,
2. single data source and multiple algorithms,
3. multiple data sources and single algorithm,
4. multiple data sources and multiple algorithms.

Category 1 includes the traditional single mode method of performing a task such as the trackers discussed above. Category 2 includes methods such as the multiple hypothesis tracker (MHT) (Cox and Hingorani, 1996), interacting multiple models (IMM) methods (Mazor, Averbuch, Bar-Shalom and Dayan, 1998), and techniques involving multi-classifiers (Foggia et al., 1999; Ho et al., 1994; Xu et al., 1992). As far as the authors are aware, no current techniques fall into Category 3, and Category 4 includes the work on sensor fusion (Murphy, 1996).

Our work falls into Category 2 where we have a single data source (the raw image) and multiple trackers that we want to combine in a sensible way. The key difference between our work and the MHT or IMM methods is that we use multiple low level algorithms rather than multiple higher level algorithms. In both the MHT and IMM methods a single low level tracking algorithm is used, but the higher level filtering models (typically Kalman filters) are distinct. For

the MHT, tracks are split if there is any ambiguity and followed independently until they can be discarded. For the IMM approach, multiple motion models for each track are simultaneously maintained and the result is a weighted sum of the prediction from each model. In our method we use multiple and fundamentally different low level trackers - in this case, a local relaxation based tracker and a global correspondence tracker. Our method is more closely related to that of multiple classifiers. However, a key assumption in much multiple classifier work (Foggia et al., 1999) is that classifiers are independent. This assumption is highly suspect when the techniques utilise the same raw data and thus we reject it. In fact, our method relies on the notion that the techniques are not completely independent.

3 Overall System Architecture

The architecture of our system is shown in Figure 1. First, corners are extracted from an input image using the Harris and Stephens corner detector (Harris and Stephens, 1988). These corners then initialise new tracks in each of the corner trackers (in our case, there are only two trackers). Each tracker tracks any current corner to the next image and passes its position to each of the classifiers at the next level. The classifiers use each corner position and several other attributes to determine if the tracker has tracked correctly. If only one tracker is deemed "correct" or if the results of all "correct" trackers are in agreement, the corner is tracked to the new position, otherwise a dropout is generated. The new corner position is used to update a Kalman filter which then predicts a corner position for the next frame using a linear motion model, and this information is passed to each tracker to process the next frame.

4 The Trackers

We use the well known Harris and Stephens (1988) corner detector for initialising our corner tracks. Details are given in Appendix A. The corners are

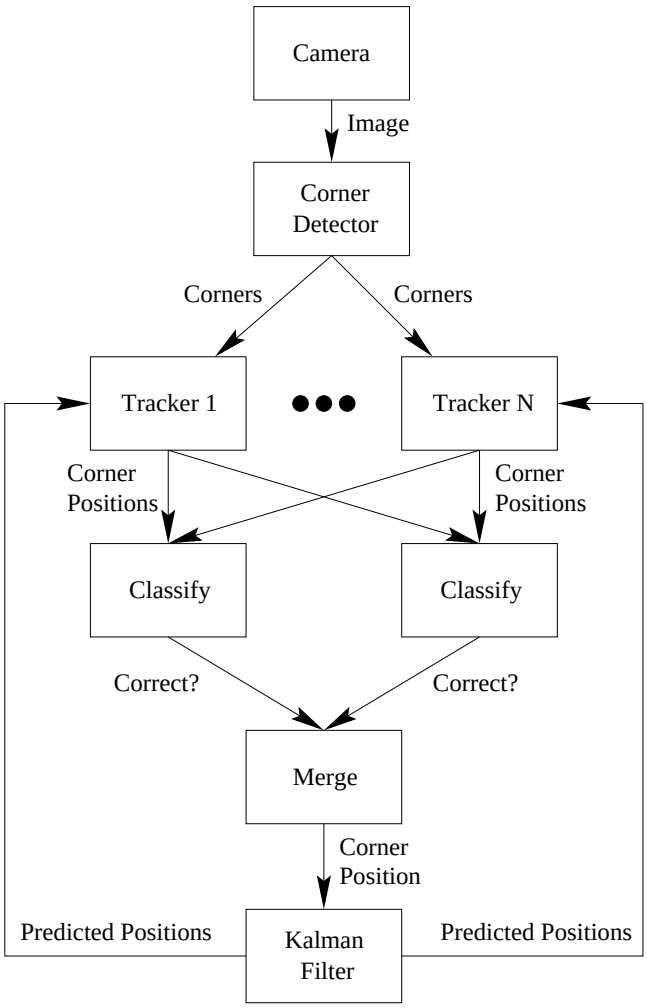


Figure 1: The overall system architecture

tracked using a Kalman filter based on a constant velocity motion model. We also maintain some corner attributes within the filter as these are useful for other aspects of the system. In particular, we observe the following quantities for each corner tracked in the system:

$$\mathbf{x} = (\mathbf{p}, v(\mathbf{p}), \mathbf{g}(\mathbf{p}))^T \tag{1}$$

where T denotes transpose, $\mathbf{p}$ denotes the location of a corner, $v(\mathbf{p})$ denotes the gray value at corner location $\mathbf{p}$, and $\mathbf{g}(\mathbf{p})$ denotes the gradient vector at corner location $\mathbf{p}$. Our system variables for the Kalman filter are simply $\mathbf{x}$ enhanced with the estimated velocity for the corner.

We implemented a corner tracking system with two component trackers: a global correspondence tracker and a local relaxation-based tracker. The component trackers, described below, could easily be modified to track any feature type for which it is possible to calculate how 'feature like' a given pixel in the image is.

4.1 The Global Correspondence Tracker

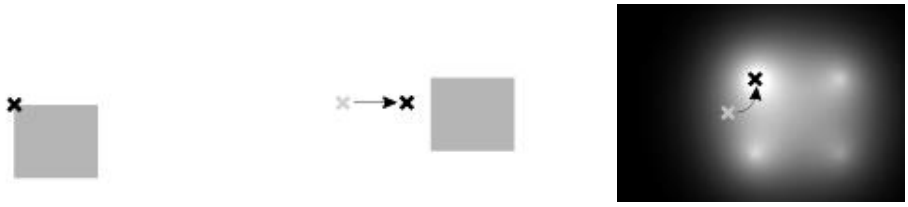
We implemented a global correspondence tracker that is based on the work of Smith and Brady (1995). Input corners are matched to candidate corners using local image intensities, local image gradients, and predicted position based on past motion. We define a difference function, $d(i, \mathbf{p})$ between the Kalman filter predicted values of tracked corner i and a candidate corner at image location $\mathbf{p}$, based on the Mahalanobis distance. The details are given in Appendix B. For each tracked corner, i, we add every candidate corner to the list of possible matches. We define the confidence that candidate corner j corresponds to the tracked corner i as

$$c_c(i, j) = \frac{1}{1 + d(i, \mathbf{p}_j(i))}. \tag{2}$$

The corners extracted by the Harris and Stephens detector that are not used are initialized as 'new corners' to be tracked through subsequent frames. This allows the system to continually adapt in a dynamic environment.

4.2 Local Relaxation Tracking

Our second tracker uses relaxation to perform a local search for a matching corner (Galvin, McCane and Novins, 1999). The process is analogous to the relaxation method used in the active contour framework of Kass, Witkin and Terzopoulos (1988). Corner relaxation works by locating a local minimum of an error function using a simple downhill search. The error function has two components: the similarity between the original corner i and the new location $\mathbf{p}$, and how 'corner-like' the new location is. For details, see Appendix C. Figure 2 shows pictorially how the relaxation tracker functions.



(a) A corner in the previous frame.

(b) The corner's position is updated using the current velocity estimate.

(c) The corner moves 'downhill' on its energy surface.

Figure 2: The corner relaxation algorithm.

To evaluate the confidence in a match, we assess the stability of convergence by running the relaxation procedure from N more locations in the neighborhood of the predicted position of the corner. In our system N was 4 and the locations were four pixels from the expected corner position along the coordinate axes. If all of the relaxation procedures converge to the same position we can be very confident that we have the correct result. We quantify this by expressing the confidence that candidate corner j matches tracked corner i as

$$c_r(i,j) = \frac{1}{1 + d(i, \mathbf{p}_j(i))}(n/N)^2, \tag{3}$$

where n is the number of relaxation procedures which converged to $\mathbf{p}$. Each

unique relaxation result is then inserted into the list of possible matches for the current corner.

5 Reliability Attributes

In our system, each component tracker outputs at most one match in the current frame for each feature detected in the previous frame. The first step in fusing the results for a particular input feature is to determine the reliability of each tracker's best match. We assume that the attribution process is run after all trackers have generated all possible matches for all features in the previous frame and ranked each of them according to a confidence value in the range $[0, 1]$. A restricted range is used so that confidence levels are broadly comparable between trackers. There is no assumption that these confidence values are probabilities – they need not sum to one. While only the highest confidence match for each tracker undergoes attribution, the confidence values of other matches are used in the calculations.

We present four simple attributes that encapsulate intuitive notions of a 'good match', $(q_0, ..., q_3)$. The first (q_0) is simply the confidence value of each tracker in a particular match. It is this attribute that is thresholded in typical tracking systems. The second (q_1) is a measure of the uniqueness of a match. This is calculated as the ratio of the best confidence to the sum of the best to the second best. The justification here is that matches that score much more highly than other candidates are more likely to be correct. The third attribute (q_2) is the one-to-one correspondence attribute. This attribute is to prevent several tracks attaching to the same candidate corner. The final attribute (q_3) is the consistency attribute. Here we test if all trackers produce similar results. In this way, we use the dependence of each of the trackers to either boost or inhibit the evidence from the other. For details of the attributes we use, see Appendix D.

6 Classification and Merging

At this stage in the processing, each tracker has output its best match, along with an evaluation of the match in terms of the four attributes. We now need to fuse the results so that the system outputs at most a single output match. We chose to implement a two-stage process. First, each tracker's match is classified as "correct" or "incorrect". Next, the correct matches are merged to produce an output match.

6.1 Classification Boundaries

Although low attribute values are broadly indicative of a correct match, determining correctness with a simple threshold on just one attribute will be prone to error. This can be seen in Figure 3, which shows the results of a global correspondence tracker on a synthetic sequence. Using ground truth data, each match was classified for correctness and plotted along with its value for each attribute. No set of vertical cuts would perfectly separate correct matches from an incorrect one.

We have experimented with a number of classification boundaries, and have had the most success with a double threshold for each attribute. In order to be deemed correct, we require that the candidate match has a low value for all attributes and a very low value for at least one attribute. More formally, we define eight thresholds and accept a match $BestJ(i)$ if and only if

$$(q_0 < U_0 \ \wedge \ q_1 < U_1 \ \wedge \ q_2 < U_2 \ \wedge \ q_3 < U_3) \ \wedge$$
$$(q_0 < L_0 \ \vee \ q_1 < L_1 \ \vee \ q_2 < L_2 \ \vee \ q_3 < L_3).$$

This classification boundary is shown graphically in 2D in Figure 4.

6.2 Training

The boundary parameters U and L are found by training the system on ground-truth data. This is done by minimizing a weighted sum of the number of

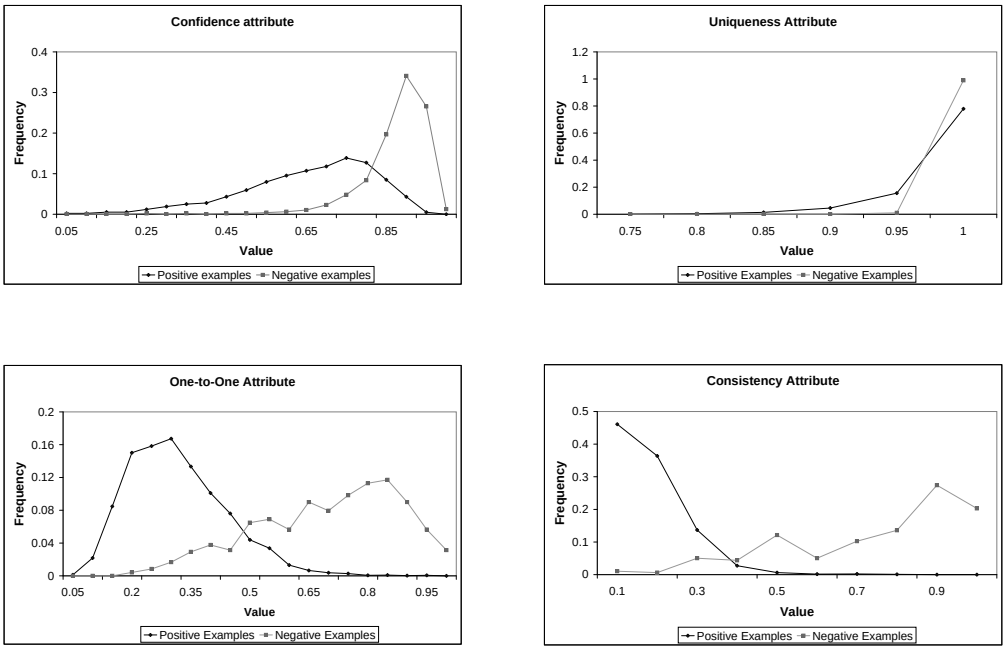


Figure 3: Relative Frequency histograms of positive and negative examples along each dimension.

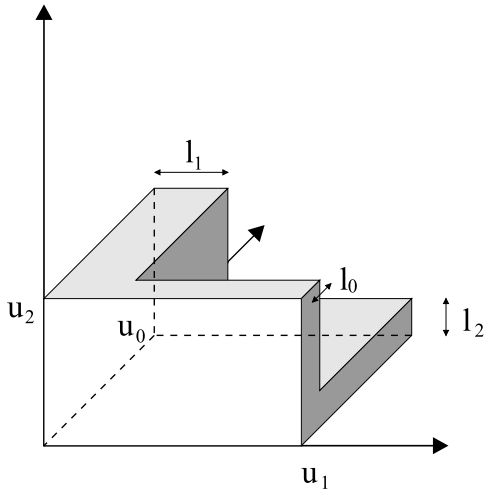


Figure 4: An example classification boundary.

dropouts and the number of errors with respect to U and L:

$$f(\mathbf{g}) = n_d + w_e n_e \tag{4}$$

where $f(\mathbf{g})$ is the error function we are trying to minimise, $\mathbf{g}$ contains the 8 thresholds $U_0 ... U_3, L_0 ... L_3$, n_d is the number of dropouts for a given $\mathbf{g}$ and n_e is the number of errors. In our experiments reported here, $w_e = 2.0$. The training method we use is an hierarchical bracketed search akin to a golden section search (Press, Teukolsky, Vetterling and Flannery, 1992). To avoid minimizing over 8 dimensions simultaneously, we minimize with respect to U first, then fix U and minimize with respect to L. The method is described in Algorithm 1. We also experimented with Powell's method (Press et al., 1992) but found that the results were not as good and the training took considerably longer.

Algorithm 1 (Training the Classifier)

```
proc train(Vector min, Vector max, int steps, Vector best)
  δ ← (max − min)/(steps − 1)
  new_best ← best
  /* search at fixed intervals */
  for current ← min; current <= max;
      current+ = δ do
      /* get the minimum score */
      if f(current) < f(new_best)
        then
              new_best ← current
      fi
  end
  /* bracket the minimum */
  min ← new_best − δ
  max ← new_best + δ
  /* check improvement */
  if (f(best) − f(new_best))/(f(best) + f(new_best)) > threshold
    then
          /* hierarchical search of new interval */
          best ← new_best
          train(min, max, steps, best)
  fi
```

.

6.3 Merging the Predictions

The procedure outlined in the previous section classifies the correctness of the best match of an individual tracker. Used in isolation, it could be used to reduce the algorithmic error rate at the expense of an increased dropout rate. By using multiple trackers in concert, we can hope to reduce the error rate without a corresponding increase in dropout rate. This approach will work as long as

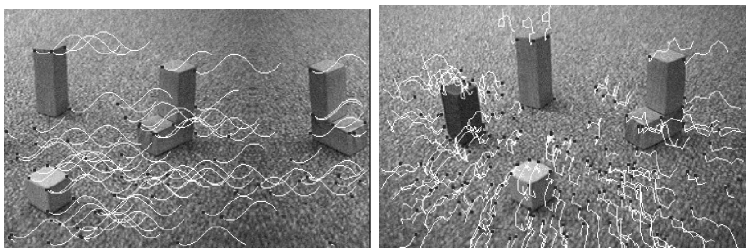
the trackers are independent enough so that it is unlikely that all trackers will fail in exactly the same situations. This technique, called multiclassification, has already proved successful in the area of handwriting recognition (Foggia et al., 1999; Ho et al., 1994; Xu et al., 1992). The two most common methods of combining multiple classifiers are the majority voting method and Bayesian analysis (Foggia et al., 1999; Xu et al., 1992).

A problem with Bayesian analysis is that, in practice, it is often used with the assumption of complete classifier independence (Foggia et al., 1999; Xu et al., 1992). This assumption is highly suspect when the classifiers use the same raw input data. Nevertheless, we have maintained the spirit of Bayesian analysis, if not the detail. Our method for using the output of each tracker as the input for determining the correctness of the other trackers is akin to estimating the conditional prior probabilities needed for Bayesian combination.

During the merge process, we immediately remove from consideration all trackers that are classified as having an incorrect match. If none appear to be correct, we have to accept a dropout. Otherwise, there is at least one tracker classified as being correct. In principle, since classification isn't perfect, if more than one tracker is classified as correct, they may not agree on the position of the match. In practice this is unlikely since the distance between each tracker's output is one of the classification attributes. However, if the difference in positions is large (greater than 2.5 pixels for our experiments) then a dropout is reported.

7 Training Data

The biggest hurdle to overcome in the proposed scheme is that of what training data to use. Obviously, the closer the training data reflects the real data which will be used, the better the results will be. However, it is extremely difficult to obtain ground truth data for most computer vision applications and this application is no different. The only practical form of ground truth data seems to be synthetic data but this introduces the problem of using unrealistic training



(a) Sine wave translation (b) Noise modulated zooming

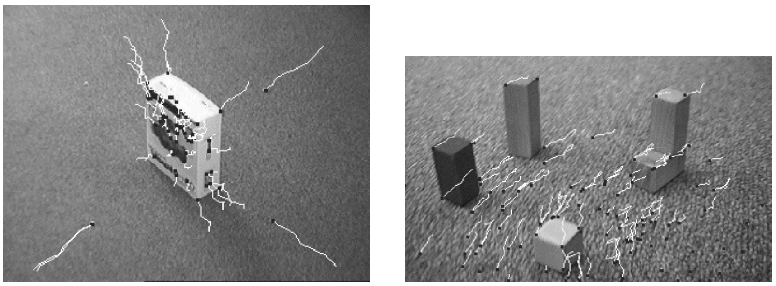
Figure 5: The motion used for the synthetic training sequences.

data. We have attempted to overcome this problem by basing our synthetic data on real images.

In the results presented below we have used two synthetically generated sequences to train the classifier for all tests. For each sequence, a single frame from a real sequence was used as the base frame and synthetic motion was applied to this frame to produce a motion sequence. The sequences differed across experiments only in the base frame used to generate the sequence - the generated motion was identical in all experiments. We propose that training the classifier on a small set of synthetic yet realistic sequences can improve performance of the tracker. However, the training sequences must be complex enough to result in both correct and incorrect matches for the training method to produce useful results. An example of the first training sequence is shown in Figure 5(a) and uses simple translation on a sine curve. The second sequence involves a noise modulated zooming motion and an example is shown in Figure 5(b). Details of the exact motions are discussed in Appendix E.

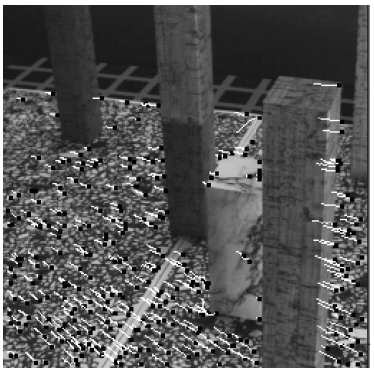
8 Results

Our test sequences consist of three real sequences for which we have ground truth data. The first two sequences (ZoomBox and MoreBlocks) were produced at the Computer Graphics and Vision Laboratory at the University of Otago, and the



(a) ZoomBox Sequence

(b) MoreBlocks Sequence



(c) TrBsb

Figure 6: The three real sequences used for testing.

method for obtaining the ground truth data is described elsewhere (McCane, Novins, Crannitch and Galvin, 2001). The third sequence (TrBsb) was obtained from Institut fur Algorithmen und Kognitive Systeme, Kognitive Systeme (Otte and Nagel, 1995). We use Frames 25 to 39 of this sequence. Figure 6 shows the last frame used of each of the sequences overlaid with corners and the tracks of those corners throughout the sequence.

Table 1 displays the results we obtained for our technique on the above three sequences. Each column represents the results of testing the system on a particular sequence. Each row represents the results of testing for a particular

synthetic training sequence. The last row indicates the best possible results using our two base trackers assuming there is an oracle that can tell us whether a given tracked corner is correct or not. The diagonal elements of the table represent the case where the given sequence was trained on a synthetic sequence generated from the first image in the test sequence. This case typically produces the best results and, except for the MoreBlocks sequence which seems to be quite difficult to track correctly, the diagonal elements are comparable with the optimal results. The off diagonal elements of the table indicate the importance (or lack of) of the training sequence. Although it is clear from the results that using a base image close to the real sequence produces the best results, it is also true that using different base images does not degrade the results significantly. This is a promising result as it implies that the choice of base image for the training sequences is not critical, and that the technique generalises well.

To check that any improvement we achieved was due to algorithmic fusion rather than simply our classification technique, we also produced results showing how each tracker performed individually using the classification technique on the first three attributes $q0 - q2$. These results are displayed in Table 2. In this case, our training sequences consist of the synthetic sequences for which the first image of the test sequence is the base image. It is clear from these results that our method of algorithmic fusion does offer a significant improvement. The advantages of the approach are highlighted by the relaxation tracker results. Although the error is reasonably low for the relaxation tracker, it produces these low errors at the cost of a very high percentage of dropouts. Generally, our combined tracker produces comparable error results to the relaxation tracker but with significantly fewer dropouts. The exception is the MoreBlocks sequence, but the error for the combined tracker could be reduced by modifying our optimisation function at the expense of increased numbers of dropouts.

Training Image	Results		
	ZoomBox	MoreBlocks	TrBsb
	(%dropouts, %errors)	(%dropouts, %errors)	(%dropouts, %errors)
ZoomBox	(8.2, 4.0)	(15.3, 13.3)	(6.9, 0.6)
MoreBlocks	(10.0, 7.2)	(14.8, 16.2)	(7.2, 1.6)
TrBsb	(8.5, 10.8)	(14.3, 15.4)	(6.1, 1.1)
Optimal	(1.9, 0.0)	(14.4, 0.0)	(2.4, 0.0)

Table 1: Results for each of the real scenes. The classifier was trained on simulated motion from a single frame of each scene and tested on the real sequence.

Tracker	ZoomBox	MoreBlocks	TrBsb
	(%dropouts, %errors)	(%dropouts, %errors)	(%dropouts, %errors)
Correspondence	(11.2, 6.9)	(14.6, 20.9)	(11.1, 1.6)
Relaxation	(20.0, 2.0)	(44.0, 6.2)	(19.2, 0.2)
Fusion	(8.2, 4.0)	(14.8, 16.2)	(6.1, 1.1)

Table 2: Results for each of the real scenes with and without algorithmic fusion. For the individual trackers the classifier was trained on three-dimensions ($q0 - q2$) on simulated motion from a single frame of the same scene.

9 Conclusion

We have presented a simple technique for improving the performance of existing feature tracking systems via algorithmic fusion. We use a classification technique to detect tracking errors, and merge the results of all matches classified as correct. We showed that the system can be adequately trained on synthetic sequences and that the choice of such sequences is not crucial to the success of the method.

The global correspondence tracker and the local relaxation tracker seem to work very well in tandem. In the future, we would like to experiment with combining other trackers under this framework.

Some of our reliability attributes must be computed using global information. This results in a system that is not suitable for real-time applications. We would like to determine if similar results to our current system could be attained with a larger set of simpler attributes.

Algorithmic fusion is a promising area. In general, our system performed better in both error and dropout rates than a global correspondence tracker. Also, the system performed significantly better in terms of dropout rates compared to a novel local relaxation tracker at the cost of a moderate increase in error rates. Such a system has the ability to produce two substantial benefits for motion tracking applications: i) we can track features for potentially more frames - substantially reducing errors in motion prediction, and (ii) we can track more features than before, giving a more complete description of the scene motion.

Appendix A Corner Detector

The corner detector we use is the well known Harris and Stephens (1988) corner detector. This detector defines a corner as one for which $h(x, y) > C$ where C is a cornerness threshold, and $h(x, y)$ is the Harris and Stephens cornerness measure:

$$h(x,y) = det(A) - 0.04 * trace(A), \tag{5}$$

where

$$A(u,v) = \begin{bmatrix} \sum \frac{\partial I}{\partial x}\frac{\partial I}{\partial x} & \sum \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum \frac{\partial I}{\partial y}\frac{\partial I}{\partial x} & \sum \frac{\partial I}{\partial y}\frac{\partial I}{\partial y}, \end{bmatrix} \tag{6}$$

and $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ are the spatial image derivatives. $\sum$ indicates the sum of values over a local region. We used 5×5 region in our calculations.

Appendix B The Global Correspondence Tracker

Using the predicted values for corner i from the Kalman filter, $\mathbf{X}(i) = (\mathbf{P}(i), V(i), \mathbf{G}(i))$, we define the 'difference', $d(i, \mathbf{p})$ between a tracked corner i and a candidate corner at image location $\mathbf{p}$ in the current frame as a weighted sum of the differences of each component of X. We use the Mahalanobis distance (Cox, 1993) rather than the Euclidean distance to account for uncertainty levels in the predicted values. The distance $d(i, \mathbf{p})$ is defined as:

$$d(i,\mathbf{p}) = w_g \frac{\mathcal{M}(\mathbf{G}(i) - \mathbf{g}(\mathbf{p}))}{(\|\mathbf{G}(i)\| + \|\mathbf{g}(\mathbf{p})\| + 1)^{1/2}} + w_v \frac{\mathcal{M}(V(i) - v(\mathbf{p}))}{\|\mathbf{G}\|} + w_p \mathcal{M}(\mathbf{P}(i) - \mathbf{p}) \tag{7}$$

where $w_g = 1.5$,$w_v = 7.03$,$w_p = 0.23$ are empirically derived constants and $\mathcal{M}(\mathbf{x})$ is the Mahalanobis distance where $\mathbf{x}$ is the innovation and the covariance matrices are the relevant sub-matrices extracted from the Kalman filter. Capitalised variables refer to predicted values and lower case to measured values. Of course, we could just use the Mahalanobis distance calculated from the entire five-dimensional observation vector and related covariance matrix. However, finding the inverse of such a matrix as required in the Mahalanobis distance is considerably more expensive than the above scheme and also removes the ability of weighting the individual measures.

The first term in Equation 7 represents the scaled relative difference between the gradient vectors at the original and candidate locations. The second term

represents the difference in image gray-value between the original and candidate locations, divided by the image gradient magnitude. Dividing by the image gradient normalizes this term's response to varying corner strengths. Without this, strong corners may generate a very large response for very small positional errors. The final term is the difference between the candidate corner location and predicted corner location.

Appendix C The Local Relaxation Tracker

We define the error caused by placing predicted corner i at image position $\mathbf{p}$ as:

$$e_i(\mathbf{p}) = \frac{d(i, \mathbf{p})}{h(\mathbf{p})^\alpha}, \tag{8}$$

where $d(i, \mathbf{p})$ is defined in Equation 7 and $h(\mathbf{p})$ is the Harris and Stephens cornerness measure. The parameter α is used to weight the response of the cornerness measure relative to the distance measure. For our experiments, we found that $\alpha = 3$ gave good results. An example energy field is shown in Figure 2(c).

The minimization process is shown graphically in Figure 2. The first step is to place a corner in its predicted position $\mathbf{p}$. We then find the 8-neighbor of $\mathbf{p}$ with the minimum $e_i(\mathbf{p})$ value, set $\mathbf{p}$ to this neighbor, and repeat until we reach a local minimum. If too many iterations (15 for these experiments) occur before a local minimum is found, the corner tracker reports failure, and a drop-out occurs.

Appendix D Reliability Attributes

In the description of each attribute below, we denote the position of the j^{th} candidate feature generated by a tracker for corner i by $\mathbf{p}_j(i)$. The confidence that i matches j, is represented by $c(i, j) \in [0, 1]$. We represent the highest confidence match by

$$BestJ(i) = \arg\max_j c(i,j). \tag{9}$$

D.1 The Confidence Attribute

The first attribute is simply the confidence value in a particular match. This is the value that is normally thresholded by the trackers when used in isolation. While it is not sufficient for determining the correctness of a match, it does provide useful information. We define:

$$q_0 = 1.0 - c(i, BestJ(i)). \tag{10}$$

D.2 The Uniqueness Attribute

The second property favors candidates that have significantly higher confidence than other matches from the same tracker. This is an intuitive notion of uniqueness. We use the ratio of the confidence of the best candidate to the confidence of the second best candidate, scaled appropriately:

$$\begin{aligned} c_1' &= c\left(i, BestJ(i)\right) && (11)\\ c_1'' &= \max\left\{c(i,j'), j' \neq BestJ(i)\right\} && (12)\\ q_1 &= 1.0 - c_1'/(c_1'' + c_1'). && (13) \end{aligned}$$

A q_1 of 0.0 indicates a unique solution, a q_1 close to 0.5 indicates that there was another candidate feature that was almost as good as the best.

Alternatively, we could have defined q_1 by considering all possible matches instead of only the top two. However, in practice this did not improve performance, so we decided to use the simpler measure.

D.3 The One-to-One Correspondence Attribute

If a tracker predicts that more than one input feature matches the same candidate output feature, it is unlikely that all matches are correct. We therefore

penalize the match if the tracker predicts that another input feature matches the position of the best candidate, $\mathbf{p}_{BestJ(i)}(i)$, with high confidence:

$$c_2' = c\,(i, BestJ\,(i)) \tag{14}$$

$$c_2'' = \max\{c(i', j'), \mathbf{p}_{j'}(i') = \mathbf{p}_{BestJ(i)}(i)\} \tag{15}$$

$$q_2 = 1.0 - c_2'/(c_2'' + c_2'). \tag{16}$$

D.4 The Consistency Attribute

Although the confidence values returned by the individual trackers are good indications of the correctness of the match, they are only as good as the individual tracker. One of the strongest indications of a correct match is that many trackers produce the same result. We define $\mathbf{p}^*(i)$ to be the average of the best match positions, $\mathbf{p}_{BestJ(i)}(i)$, from each tracker. Our final attribute measures the deviation of $\mathbf{p}_{BestJ(i)}(i)$ from the average:

$$D = \|\mathbf{p}_{BestJ(i)}(i) - \mathbf{p}^*(i)\| \tag{17}$$

$$q_3 = 1.0 - k_3/(D + k_3). \tag{18}$$

Here $k_3 = 5.0$ and was empirically determined to improve the spread of the attribute in the range of 0 to 1.0. There is an implicit assumption that if $D = 5.0$, then we are 50% confident that the tracker is correct. So in this sense, k is an indication of how large a discrepency between the trackers we are willing to accept still indicates a consistent result.

Appendix E Training Sequences

The first training sequence involves translation on a simple sine curve:

$$x(t) = x(t-1) - \frac{40.0\pi t}{t_{max}} \tag{19}$$

$$y(t) = y(t-1) - 10.0 \sin\left(\frac{4.0\pi t}{t_{max}}\right) \tag{20}$$

where $\mathbf{p}(t) = (x(t), y(t))$ are the new pixel positions at time t, and t_{max} is the maxiumum time of the sequence. Figure 5(a) displays the last frame in the translation training sequence for the MoreBlocks base frame with the current corners and the tracks for each of the corners overlaid.

The second training sequence involves a noise modulated zooming motion which is defined as follows:

$$\mathbf{p}(t) = ((1.0 - \frac{t}{t_{max}})z_s + z_f * \frac{t}{t_{max}})\mathbf{p}(t-1) + \mathcal{N}(0, \sigma) \tag{21}$$

where $z_f = 1.0$ is the final zoom factor, $z_s = 1.7$ is the initial zoom factor, and $\mathcal{N}(0, \sigma)$ is a normally distributed random variable with $\sigma = 4.0$ for our experiments. Figure 5(b) shows the last frame in the synthetic zooming sequence using the MoreBlocks base image.

References

Cox, I. J. (1993). A review of statistical data association techniques for motion correspondence, *International Journal of Computer Vision* **10**(1): 53–66.

Cox, I. J. and Hingorani, S. L. (1996). An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18**(2): 138–150.

Foggia, P., Sansone, C., Tortorella, F. and Vento, M. (1999). Multiclassification: reject criteria for the Bayesian combiner, *Pattern Recognition* **32**: 1435–1447.

Galvin, B., McCane, B. and Novins, K. (1999). Oscar: Object segmentation using correspondence and relaxation, *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling (3DIM) '99*, Ottawa, Canada.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector, *Proceedings 4th Alvey Vision Conference (AVC88)*, pp. 147–151.

Ho, T. K., Hull, J. J. and Srihari, S. N. (1994). Decision combination in multiple classifier systems, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(1): 66–75.

Kang, S., Szeliski, R. and Shum, H. (1997). A parallel feature tracker for extended image sequences, *Computer Vision and Image Understanding* **67**(3): 296–310.

Kass, M., Witkin, A. and Terzopoulos, D. (1988). Snakes: Active contour models, *International Journal of Computer Vision* pp. 321–331.

Mazor, E., Averbuch, A., Bar-Shalom, Y. and Dayan, J. (1998). Interacting multiple model methods in target tracking: a survey, *IEEE Tracnsactions on Aerospace and Electronic Systems* **34**(1): 103–123.

McCane, B., Novins, K., Crannitch, D. and Galvin, B. (2001). On benchmarking optical flow, *Computer Vision and Image Understanding* . To appear.

Murphy, R. R. (1996). Biological and cognitive foundations of intelligent sensor fusion, *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans* **26**(1): 42–51.

Otte, M. and Nagel, H.-H. (1995). Estimation of optical flow based on higher order spatiotemporal derivative in interlaced and non-interlaced image sequences, *Artificial Intelligence* **78**: 5–43.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992). *Numerical Recipes in C*, Cambridge University Press.

Shi, J. and Tomasi, C. (1994). Good features to track, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR94)*, pp. 593–600.

Smith, S. and Brady, J. (1995). Asset-2: Real-time motion segmentation and shape tracking, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(8): 814–820.

Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features, *Technical Report CMU-CS-91-132*, Carnegie Mellon University Technical Report.

Xu, L., Kryzyzak, A. and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition, *IEEE Transactions on Systems, Man and Cybernetics* **22**(3): 418–435.