Metric Mixtures for Mutual Information (M^3I) Tracking

Nicholas Dowson and Richard Bowden
Centre for Vision, Speech and Signal Processing
University of Surrey
Guildford
GU2-7XH, UK
{n.dowson;r.bowden}@surrey.ac.uk

Abstract

A new method for updating the template in a feature tracking application is presented, which has minimal memory and processing overhead. The proposed method is an expectation maximisation inspired approach based on modelling the variable appearance of a template using a Gaussian mixture model in a discrete metric space, termed the M^3I tracker for short. The proposed technique is compared to various other techniques in several experiments, where it performs robustly. Several comparison methods are outperformed. In addition to robust template tracking it has wider applications to advanced techniques such as AAMs and deformable templates.

1. Introduction

The tracking of features as they vary in appearance during a video sequence is an important and difficult problem in machine vision, as the large amount of research in this area implies. Template matching is one method of tracking features and involves the movement of a template or kernel over a region in the image and locating the area in the image which best matches that kernel.

Lucas and Kanade made one of the earliest attempts to locate a representative feature (template) in another image [5]. They limited the processing by using a Newton-Raphson method to traverse the search space. Although translations were considered, they mentioned how the method could easily be adapted to optimise other transforms such as rotation. Indeed, later research extended this method to consider rotation, affine translation and warping [1]. In addition new minimisation methods have been applied, which have faster convergence and greater likelihood of finding the global minimum, such as the Levenberg Marquardt method [2].

It is natural to extend feature detection to tracking a feature over a motion sequence. This leads to a further problem: how and when to update the template. If the template is never updated, tracking will only work as long the template closely represents the current appearance of the feature. This assumption is generally safe for several frames after the one from which the feature was extracted. Eventually however, the template does not represent the feature sufficiently well and a catastrophic failure ensues: i.e. the error suddenly becomes very large. One alternative to this is to update the template after every frame. However, sub-pixel errors inherent to each match are stored in each update. These drift errors gradually accumulate and the template drifts off the original feature.

Two recent approaches to overcome the problem of drift, include those by Matthews et. al. in [6], and Kaneko and Hori in [4]. Matthews approach is a simple but effective extension of the naive update algorithm, where the first template from the frame is retained and used to correct location errors made by the updated template. If the size of the correction is too large, the algorithm acts conservatively by not updating the template from the current frame. Kaneko on the other hand trades off between accumulated drift error and misrepresentation error. In Kaneko's algorithm the template is updated just before a catastrophic error occurs. Each of these errors is modelled by estimating the boundary of the maximum error for each possible template.

Although strictly speaking it was not intended in this context (without AAM), the single drawback to Matthews approach, is that the appearance of the feature could eventually change enough that the distance between the two matches will always differ by a large amount. This will result in no further updates being made, and failure. Kaneko's approach attempts to minimise the number of updates, but eventually sub-pixel errors accumulate enough for drift to occur. Both of the afore-mentioned errors also rely on the first template being a good representation of the feature.

This paper presents a new statistical method for updating a template that is based on building up a model of the feature appearance. As tracking proceeds a library of templates is built up, to which a Gaussian Mixture Model is fitted similar to the approach used by KaewTrakulPong and Bowden in [3]. This online or incremental maximisation inspired approach attempts to select the templates that match the feature the best. Toyama and Blake use an offline exemplar based approach to learn appearance and dynamics from a training set of examples in [8]. They term this the Metric Mixture (M^2) approach. We use a similar metric space to build a probability density of appearance on the fly without any *a priori* knowledge. Given sufficient samples, the feature appearance will be as close to optimal as the available data allows. The Gaussians in the mixture model are weighted according to how many recent matches have been made and obsolete portions of the model are aged out. This approach does not count on the first template being a good representation of the feature. Instead, it uses an initial template to seed the model, from which more reliable templates are later extracted. Drift error is limited by the fact that templates with accumulated sub-pixel errors will generally not be close to a Gaussian's mean. For clarity, the proposed method is presented in the context of a brute force method of traversing the solution space. Also, translation is the only transformation considered.

The remainder of the paper is organised as follows. Section 2 gives some background to tracking and correlation methods. The proposed method for choosing template appearance is discussed in Section 3, followed by the experiments and results in Section 4. The paper is concluded in Section 5.

2. Template Tracking

We begin by formalising the problem of tracking a feature in an image using a template. Given a motion sequence of N images, where $S_n(x,y)$ represents the pixel intensity at position (x,y) in frame S_n, we wish to find the transformation T that minimises the distance function D between some kernel or template s_n such that

$$\arg\min_{\forall T} D\left(S_n, T(s_n)\right) \tag{1}$$

where $D(.)$ is some correlation measurement such as Sum of Squared Distances (SSD), and s_n is a a small $s_X \times s_Y$ image patch selected from an earlier image. As a search over the entire image is costly and unnecessary it is restricted to a small region surrounding the last known position of the feature. For a non-updating template, three assumptions are necessary for the algorithm to work:

- The feature does not change significantly in appearance compared to the kernel extracted from S_1.
- The transformation T is sufficient to accommodate any breach in the first assumption.
- The feature is always visible and is always within the search window.

Our approach makes no such assumptions about the appearance of the model but instead attempts to build to build a statistical model that fully describes a feature's possible appearances. A detailed explanation of the model is left for Section 3, since a description of the distance function is required first.

2.1. Mutual Information

Several distance functions were considered, including SSD, renormalised SSD and Mutual Information. Mutual information was eventually chosen as the favoured approach due to it's robustness to environmental lighting conditions, more pronounced minima and accounting for the representativeness of the template [9].

Mutual Information quantifies the information shared between two sets of data. The definition as defined in [7] in terms of entropy:

$$J(z_i, z_j) = H(z_i) + H(z_j) - H(z_i, z_j) \tag{2}$$

where z_i and z_j are sets of data being compared, and $H(.)$ is the entropy of a random variable and is defined as

$$H(z_i) = -\int p(z_i(x,y)) \ln p\big(z_i(x,y)\big) \mathrm{d}x\mathrm{d}y \tag{3}$$

where $p(z_i)$ is the probability of z. In this context equation (3) is approximated to its discrete equivalent, where $z(i,j)$ represents the probability of intensity i and j appearing at the same pixel location in the template and the comparison image region, respectively.

3. The Metric Mixture Model

As previously mentioned, our approach does not make any assumptions about the appearance of the feature. Instead a probabilistic model is constructed, which explains the variation in a feature's appearance $P(s_n|S_n)$. Nothing about S is known before operation, except for a single example s_0, which is not sufficient to build a model. Simplistic updates only serve to move the position of the model in space rather than expand it to properly describe s's occupation. To overcome this our approach attempts to incrementally construct $P(s|S)$ in a similar fashion to an online expectation maximisation (EM) algorithm [3]. We represent the model as a mixture of K Gaussians

$$p(s_n) = \sum_{k=1}^{K} w_k \eta_k(s_n; \mu_k, \Sigma_k) \tag{4}$$

where w_k is the weight parameter of the kth Gaussian component and satisfies the condition $\sum_{\forall k} w_k = 1$. In the Gaussian distribution of the kth component: $\eta(s_n; \mu_k, \Sigma_k)$, μ_k is the mean kernel. The distance from the mean is calculated by the distance function, in this case the MI function $J(.)$. The standard deviation Σ_k is assumed to be spherical.

Considering kernels as points in some appearance probability space, kernels of similar appearances will cluster together. These clusters are what each Gaussian represents. Overly large Gaussians could result in misrepresentation of the feature, so some discrimination between inliers and outliers is required. The membership of a particular template s_n to a Gaussian η_k is determined by its proximity to the Gaussian's mean μ_k.

$$t_k = \begin{cases} 0 & J'(s_n, \mu_k) < \tau_k \Sigma_k \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

where t is the threshold. Gaussians that are sufficiently close together may overlap, so a template may belong to more than one Gaussian. $J'(s_n, \mu_k)$ is the normalised value of J. For clarity, the discussion of the normalisation is relegated to the end of the section. For now, please treat J and J' as if they are equivalent. K will generally be some small number greater than one, since clusters of representative kernels are not necessarily of spherical distribution. Neither are clusters necessarily in close proximity to each other, since feature appearances may vary a great deal if they are included or if lighting conditions change.

The relationships between each sample template in a Gaussian are stored in a correlation matrix C,

$$C = \begin{pmatrix} J(s_1, s_1) & J(s_1, s_2) & \dots & J(s_1, s_M) \\ \vdots & & & \vdots \\ J(s_M, s_1) & J(s_M, s_2) & \dots & J(s_M, s_M) \end{pmatrix} \quad (6)$$

The number of kernels that are stored is limited to M to ensure that the memory and processing requirements remain low. On the other hand M should large enough that each Gaussian is representative of the local cluster of kernels.

In general, newly created Gaussians are less reliable than older ones, since they have fewer samples and the variation in appearance may be due to transient environmental effects. To model the effect of a particular Gaussian gaining "trustworthiness", it's weight is increased each time a region in the image is located that is an inlier. The increase is at the expense of the other Gaussian components:

$$w_k^{(n+1)} = \begin{cases} w_k^{(n)} \frac{\alpha}{1+\alpha} & k = K' \\ w_k^{(n)} \frac{1}{1+\alpha} & \text{otherwise} \end{cases} \quad (7)$$

where K' is the index of the successfully matched Gaussian and α is the learning rate such that $\alpha \in (0;1)$. $1/\alpha$ defines the time constant which determines change. The learning parameter trades off between outmoded representations of the feature gradually being removed, and new erroneous representations suddenly taking over and causing failure.

Using all K Gaussians to search for the feature each time is time-consuming and unnecessary. In practice, a representation that is within the threshold t_k of the current Gaussian is adequate. So, a greedy algorithm is employed that uses the Gaussians in descending order of weight w_k. If a successful match is made, no further Gaussians are checked and the new template is incorporated into the successfully matching Gaussian.

To start with, only one Gaussian exists. The first outlier s_n to this Gaussian is used to seed a new Gaussian, along with the closest matching template within the first Gaussian. The best matching template is assumed to be the template from the previous frame s_{n-1}.

Since the two samples will generally be close together, a tolerance t based on the standard deviation will be very narrow making any other possible samples outliers. A newly created Gaussian will does not have enough samples to be a good representation of a feature's appearance. Hence until three or more samples are available an initial tolerance value is used. The initial tolerance value is calculated by perturbing the template by one pixel in each direction and calculating the MI value. The initial threshold is set to the best match from the 8 border positions:

$$t_k^{(1)} = t_k^{(2)} = \min_{\Delta y; \Delta x \in [-1;1]} \{J'(s_n(x,y), s_n(x+\Delta x, y+\Delta y))\} \quad (8)$$

Eventually all K Gaussians are populated. It is still conceivable in this case that the feature can take on an appearance outside of any the existing models. In this case, the lowest weighted Gaussian is destroyed and a new one is created in its place. In this way obsolete representations can be aged out. This event is flagged as a possible tracking failure. Three or more successive failures to reach the threshold of any of the existing Gaussians, indicate that either a feature is changing appearance very rapidly or that it has been obscured. In either case, tracking is impossible, so the location position of the feature is not updated again until a successful match is made.

μ_k represents the median of the samples rather than the mean, since a mean in image space generally implies blurring and a loss of feature representation. The median is calculated by finding the template that most closely resembles its fellow Gaussian members. This is the template with the lowest overall mutual information value, i.e. the lowest column sum in the correlation matrix:

$$i = \arg\min_i \sum_{\forall j} C_{i,j} \quad (9)$$

This median template is used as the exemplar template for locating the feature in search window. A difficult problem now occurs, because the probability distribution function of MI values does not have a defined maximum. As a result, the set of MI values do not fit a Gaussian distribution cleanly. Hence the collection of MI values need to be "Gaussianised" in terms of the median template:

$$J'(S_n, s_k^{(n)}) = 1^* - \frac{J(S_n, s_n)}{J(s_n, s_n)} \qquad (10)$$

where 1^* is a matrix of ones with the same dimensions as the numerator. The denominator is a scalar.

This particular normalisation is chosen for two reasons. Firstly, it is a more convenient value, since the best possible match gives a zero value like SSD. Secondly, and more importantly, the value of J' is always a value between zero and one, which allows a usable threshold to be set using (5). Σ and t must be also be recalculated with the new J' values. It is important that Σ is set using the standard deviation of values in the median column only, since the normalisation makes the kernel space non-metric across the columns of C. However, within a single column the kernel space *is* metric. Hence J values are stored in C, rather than J' values, and the median is selected before Σ and t are calculated.

4. Experiments and Results

To test the proposed method a number of experiments were undertaken, in which five algorithms were compared: no update $s = s_0$, naive update $s = s_{n-1}$, strategic update (Matthews approach), M^3I with τ=1 , and M^3I with τ=2. The first two algorithms were used for illustrative purposes only, since they were expected to fail. A number of features were selected in three video clips and the tracked position of these was observed for each of the five algorithms. A feature was selected and hand labelled with a ground truth position. The feature was chosen for the high likelihood of tracking failure due to its variations in appearance. Due to lack of space, the results for one sequence only are shown: that of the newscaster shown in Figure 1. However, these results are indicative of algorithm performance in many cases.

M^3I was limited to 5 Gaussians, each consisting of 60 templates. The kernel size was 11*11 pixels, while the search window was defined as a 5*5 pixel border around the kernel.

Figure 1 shows four images from the motion sequence with the estimated positions of the template using each of the five algorithms. Frame 0 is the beginning of the sequence. Frame 95 shows the newscasters head has turned to the side and the chosen feature has been obscured. At Frame 120, the newscasters head has turned back to face the camera revealing the feature once again. Frame 150 is the last frame in the sequence.

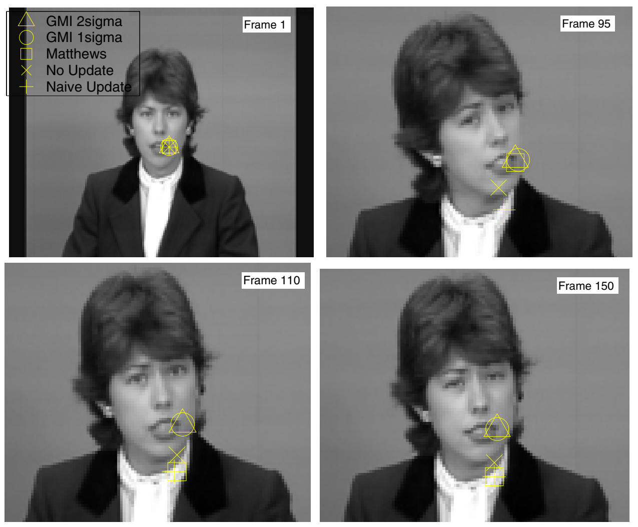

Figure 1. Feature Positions at Different Stages of Motion Video

The results of these tests are shown more clearly in Figure 2, where the error of the five algorithms is plotted against the frame number. The no-update and naive-update algorithms failed in a predictable fashion. Unexpectedly, the strategic algorithm failed not long after the no-update algorithm failed. Figure 1 shows the estimated feature position for the strategic update as it starts to fail. In further tests where the search window was increased after failure, the strategic update algorithm recovered.

Both of the M^3I trackers, temporarily drift off the feature at frame 97. The τ=1 algorithm, with its tighter threshold immediately creates a new Gaussian, since the new appearance of the feature is so different from its existing models. This allows the algorithm to quickly adapt and locate the feature despite its changed appearance.

The τ=2 algorithm, with its looser constraints, incorporates the feature into its existing models, where their influence is somewhat diluted and takes somewhat longer to adapt to the changed appearance of the feature. The algorithm recovers immediately the feature resumes its old appearance. Worth noting is that during the τ=2 test, the tolerance was high enough that only one Gaussian was ever used. Even so, the resulting error was still low, proving that the approach of using a median is sound.

Unlike the comparison method, M^3I successfully tracked the selected features in both cases. This was without resorting to methods such as increasing the search window size, which is computationally expensive.

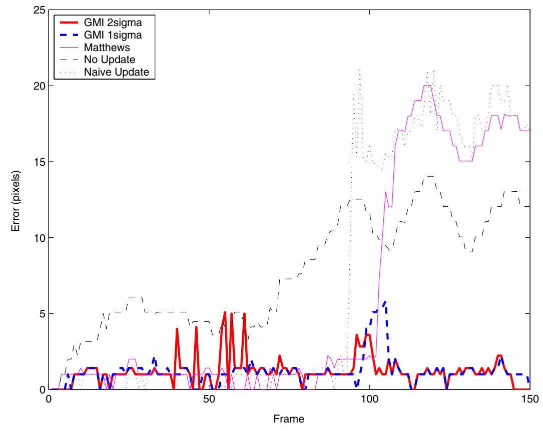

Figure 2. Error Rates for Different Tracking Algorithms

4.1. Note on Memory and Processing Overhead

The additional memory required for storing the mixture models is insignificant compared to the size of the video clip. For example, assuming a standard YUV image clip of a 1000 frames, the memory required is $N \cdot X_{\max} \cdot Y_{\max} = 1000 \cdot 176 \cdot 144 = 25344kB$. The mixture model on the other hand requires space for K Gaussians and a maximum of C samples per Gaussian per kernel. Assuming the values used in the experiments: $K \cdot I_{\max} \cdot X_{\max} \cdot Y_{\max} = 5 \cdot 60 \cdot 11 \cdot 11 = 36kB$.

The processing overhead of the model is primarily in the calculation of a new row in the correlation matrix. The calculation of MI is $O(W^2 \cdot N^2)$ per frame, assuming a brute force search for optimal translation. W is the size of the window over and above that of the kernel size N. The cost of updating the correlation matrix every frame is $O(M \cdot N^2)$, so the processing overhead is not large.

5. Conclusion

A new statistical framework for the template update problem has been proposed. The proposed method involves the storage of templates within a Gaussian Mixture Model. A correlation matrix is used to quantify the relationship between each sample template in the M^3I tracker, based on their Mutual Information value. The median template and threshold value, used to detect outliers, are updated intelligently as the algorithm progresses. The method places a minimal overhead on the existing correlation operations in terms of memory and processing cycles.

The proposed method has a more generic philosophy than existing methods. It is a more robust representation of a feature than algorithms using a single updating template. Most existing methods are designed to minimise template updates and thereby stave off drift error, rather than overcoming it directly. Currently, the proposed method does not completely eliminate drift errors, but is more successful at minimising them, since it stores and optimally reuses old templates. However, the proposed method could be adapted to optimally reselect templates from previous images from the sequence based on *a-posteriori* data, which will eradicate the problem of drift entirely. This is left for further work.

The M^3I method was compared to another recently proposed update method. Test results confirmed the benefit of the method proposed in this paper. Currently, the Mixture Model method is being extended to use the Lucas-Kanade type techniques so that other transformations such as rotation and affine transforms may be used.

Acknowledgements

This work has been made possible by funding provided from CVSSP at Surrey University and an ORS scholarship provided by the Department for Education and Skills, UK.

References

[1] S. Baker, R. Gross, I. Matthews, and T. Ishikawa. Lucas-kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[2] A. Fitzgibbon. Robust registration of 2D and 3D point sets. In *Proceedings of the British Machine Vision Conference*, Sep 2001.

[3] P. KaewTrakulPong and R. Bowden. A real-time adaptive surveillance system for tracking low-resolution colour targets in dynamically changing scenes. *Image and Vision Computing*, 21(10):913–929, 2003.

[4] T. Kaneko and O. Hori. Template update criterion for template matching of image sequences. In *Computer Vision and Pattern Recognition, 2003*, volume 1, pages 1–5, June 2003.

[5] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, 1981.

[6] I. Matthews, T. Ishikawa, and S. Baker. The template update problem. In *Proceedings of the British Machine Vision Conference*, Sept. 2003.

[7] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Inc., third edition, 1991.

[8] K. Toyama and A. Blake. Probabilistic tracking in a metric space. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume 2, July 2001.

[9] P. Viola and W. Wells. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.