Using colour features for video-based tracking of people in a multi-camera environment

Mathew Price, Fred Nicolls, Gerhard de Jager

Department of Electrical Engineering
University of Cape Town
Rondebosch 7701, South Africa
mathew@dip.ee.uct.ac.za

Abstract

This paper provides a brief overview of ongoing work towards the realisation of a multi-camera video-based person tracking system. The developing system is orientated towards a distributed architecture for sharing visual information between multiple cameras. A hierarchical method for modelling people in scenes using colour features is proposed, however, future development will also look at including other useful measures such as texture. The method uses a kernel-based classification implementation to track a base set of visual components which can then be grouped to form high-level object descriptors. Some results are shown for a Kernel Adatron implementation of the classifier.

1. Introduction

Computer aided surveillance has been a developing area in computer vision circles for some time. With the increase in computer processing speeds and memory sizes, much past research is starting to have useful applications now that near real-time implementations are realisable.

This project has been primarily aimed at the global problem of tracking several people, simultaneously, through a multi-camera environment. Thus, in addition to dealing with single-view occlusions and scene changes, the system needs to be able to detect object movement between multiple camera views, which may, or may not, be overlapped. The problem faced is one that requires a system capable of coordinating data among several distributed resources in a manner that can be tied in with current computer vision methods.

Focus has been aimed at using colour as the chief reference to distinguish people and objects as they traverse the environment. It is envisioned that further features may prove useful in addition to colour, such as texture and shape descriptors[7][5]. However, for simplicity and speed in the initial implementation, these have been omitted.

There are two ways of looking at a visual tracking system: Estimation and Classification. If we have a near real-time stream of image samples from an initialised target, estimation can be used to track the target's position by exploiting statistical relationships of the target's motion. This process works as long as the measurement data is reliable and the object can be clearly located at each step. In the case where this condition is not met, the estimator's accuracy will degrade until the target has been lost, thereby requiring a full re-initialisation of the system.

This drawback leads to the idea of creating a complementary system, using a classification scheme, to deal with these unstable cases. In this way, it was hoped that a symbioses could be developed where the estimator (while tracking correctly) would provide a stable set of measurements which could be used to generate a set of online features for training a neural network. Conversely, should the estimator reach an unstable state, a classifier could then determine the correct position of the target and thus automatically re-initialise the estimator's model parameters. Ultimately, this would enable the system to deal with events such as occlusion, video transmission breaks and transitions between multiple camera views.

The following sections describe a distributed classification system capable of tracking colour components belonging to moving objects and persons. Operation proceeds by breaking a scene into unimodal colour components which are then classified into temporary object classes using an online kernel adatron. The final stage involves refining the allocation of classified colours to their respective objects based on group and spatial dependencies, though this stage has not been completely implemented as yet.

2. System Overview

A basic premise of the system is that data should be hierarchically divided so that different levels of information can be processed appropriately (Figure 1). In this application, our lowest data level is the video input from the camera network.

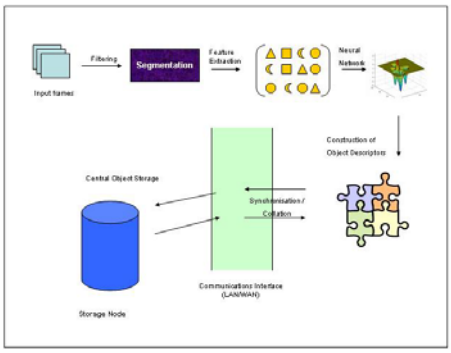

Figure 1: *System Data Flow Diagram.*

The first level of processing involves repetitive, autonomous tasks such as noise filtering, colour transformations and smoothing. Following this, the data enters the low-level processing level, where visual feature detection takes place (i.e. colour/texture). These visual features are extracted using colour

segmentation, and tracked using a neural network classification technique.

Once reliable visual features are available, construction of high-level descriptors of foreground objects is possible. These descriptors consist of combinations of the low-level visual features and can then be shared through the distributed processing system.

3. Feature Extraction

Extracting good features is probably the most critical task. While it is relatively simple to take a variety of measurements from images, it is imperative that one ensures that the feature measurements are both repeatable and comparable (i.e. the feature space is consistent and uniform).

Since the thrust of the project is to exploit the colour relationships of targets, a reliable chromatic measurement system is required. It was found that a convenient method for extracting colour, was to use CIE Lab colour co-ordinates which preserve perceptual uniformity.

Uniformity in the colour space refers to the property that standard distance measures (i.e. Euclidean) are proportional to the change in perceived colour. Owing to the fact that the neural network uses radial basis functions for its kernels, this uniformity becomes very useful when discriminating between colour clusters, since a translation in feature space relates directly to a change in perceived colour. This gives the system a human-like ability to distinguish colours. It is of course also possible to create an arbitrary colour space aimed at maximising colour discrimination, however, this discrimination will only be as good as the training set, and may not necessarily preserve uniformity.

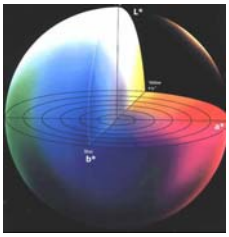

Figure 2: *CIE Lab Colour Space.*

The CIE Lab space (Figure 2) consists of two colour axes and a luminance L or brightness axis. The a values range from red to green, while the b values describe blue to yellow. The configuration is similar to the HSV colour space, though the Lab space is spherical and the co-ordinates, being rectangular, allow the Euclidean distance between points to be relative to the difference in colour. A disadvantage is its costly transformation.

Initially, during training, a motion segmented image is useful for isolating moving objects which need classification. However, once enough observations have been made, it is possible to switch back to classifying unsegmented images, which would decrease classification errors due to bad segmentation.

3.1. Colour Segmentation

Apart from motion segmentation, a colour segmentation scheme is required to generate a list of blobs and scene colours. This allows the object colours to be represented as features while retaining the spatial information (unlike histogram methods).

Various connected-region labelling algorithms have been tested (recursive region growing among others), however, pyramid segmentation[6] has provided the most promising results in terms of speed and quality.

The input image is down-sampled by a defined number of levels by means of Gaussian pyramid decomposition[2]. The pyramid is then built upwards (to the largest image), establishing links between a pixel on the current level and its candidate father on the adjacent level, if certain thresholds are met[6]. In this case, the thresholds are distances between pixel colour co-ordinates which in terms of CIE Lab space relates to closeness in perceived colour.

The output of the pyramid segmentation method (Figure 3) is then processed into a list of image regions (connected components), each representing a unimodal colour cluster in the image. The CIE Lab co-ordinates of these clusters are then used as features to train a neural network which can then classify future image regions into a basic colour set. Several clusters in this set can then be combined, by exploiting neighbourhood information, to form a high-level person/object descriptors.

Figure 3: *Pyramid Colour Segmentation.*

4. Neural Network

As described previously, the base function of the system consists of being able to correctly identify the set of target colour clusters between observations. The classification process entails providing a probability comparison between the set of target colours and the visible colour components in the latest image. Since the feature data is generated by selecting Gaussian colour clusters, a Gaussian kernel-based probabilistic network seemed the logical choice.

4.1. Kernel Adatron

Several classification techniques could be easily applied to this model, and one could argue that a simple K-Nearest Neighbour classifier could easily do the job. The selection of the Kernel Adatron in particular is based on the overall needs of the system. Firstly, each model needs to be easily structured for global synchronisation with network storage nodes. Secondly, a degree of compactness is required both for speed in classification/updates and in storage. Finally, since observed data may often fall near the edge of the cluster group misclassification could cause invalid updates to be made, thus causing the system reliability to degrade.

The kernel adatron (KA) is a simple implementation of a support vector machine, which allows maximisation of the margin in the feature space, thus forming a non-linear decision boundary in the input space[8]. Effectively, this provides an optimal decision boundary between classes. In the implementation, two target classes are represented by positive and negative unit Gaussians respectively. This allows a classification process to simply sum the Gaussian weights and classify between the two classes by the sign of the resulting probability (+1; -1). Fig-

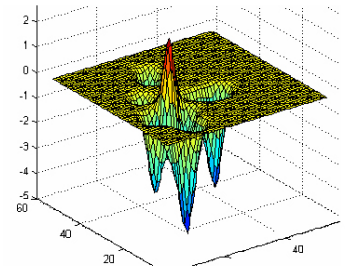

Figure 4: *Kernel Adatron example for 2-dimension features*

ure 4 shows an example of a constructed KA. The positive peak identifies the current target class, while the negative peaks show other classes. Regions where the graph is zero are unclassified.

The largest difference between the KA method as opposed to the PNN (Probability Neural Network) is that, during training, the KA assigns and updates a set of importance weights (α) which identify how close each kernel is to the decision boundary (support vectors). The primary advantage to our application here is that kernels with very small alpha values are not important to decision making and can therefore be discarded for classification. This results in a pseudo-compression scheme for the feature vectors, which constrains the models to a finite size and therefore limits the load of the network. Effectively, the compression works in a similar manner to an adaptive averaging filter, with the advantage that previous features are not necessarily averaged by time but rather by importance to classification.

4.1.1. Implementation

One of the features of the Kernel Adatron is that it is extremely easy to implement since it relies on only a few matrix operations. This implementation is extended from course literature by Green J. (Soft Computing 1999, UCT Press) which summarises theory presented in [8].

X_T — (m x n) matrix of m input features (rows) with n dimensions (cols).
T — (m x n) matrix of target values for n classes, each having m features.
A — (m x n) matrix of alpha values for n classes, each having m features.

The X_T matrix is a maintained set of features which determine the characteristics of each kernel. T assigns the target values for each feature vector. Columns in the T matrix represent each class, while row values identify whether the respective feature belongs (+1) or does not belong (-1) to the class. A typical target column will contain at least one $+1$ value. The A matrix has identical dimensions (and representative form) to the T matrix, but instead maintains the α weights (decision importance) for each feature (row) in the X_T matrix. If only one feature per class is stipulated, the A and T matrices become square.

4.1.2. Training:

For brevity the calculation of Euclidean distance between the input vectors has been omitted and the operation $d^2(x_1, x_2)$ is defined as the squared Euclidean distance between the elements of x_1 and x_2. The operation returns a symmetric matrix. For reference see the implementation of $dist2$ in the Netlab Matlab toolbox (Bishop and Nabney 1997).

Training proceeds in the following manner (referenced equations are defined below):

1. Calculate squared Euclidean distance matrix (Eqn 1)
2. Activate Gaussian kernels (Eqn 2)

   While less than i iterations **AND** $m < 0.99$ **DO:**

3. Calculate signed and weighted kernels (Eqn 3)
4. Update α values based on learning rate η (Eqn 4)
5. Calculate margin - distance from decision boundary to nearest point (Eqn 5).

$$D = d^2(X_T, X_T) \quad (1)$$
$$K = e^{-1/(2\sigma).D} \quad (2)$$

While less than i iterations **AND** $m < 0.99$

$$Z = K.A.T \quad (3)$$
$$\delta A = \eta(1 - Z.T) \quad (4)$$
$$m = 0.5\Big(min(Z^+) - max(Z^-)\Big) \quad (5)$$

4.1.3. Classification:

The classification procedure is identical to training steps 1 through 3 except that the distance matrix is now calculated between X and X_T, where X are the features to be classified.

4.1.4. Update:

Updating is simply a selection process where features classified positively in a class and meeting a similarity threshold are updated. In addition, features which were unclassified (sum of kernels = 0) and are significantly different from all other classes are appended to the X_T matrix. This addition obviously includes initialised updates to the A and T matrices as well.

5. Object Models

Once a reliable set of classified colour components is available, more complex grouping of the components can be used to create high-level visual object/person descriptors. Since several objects can share similar colours and since not all colours may always be visible, these descriptors cannot be represented by a combined feature vector.

One idea is to use a voting system where each object votes for a component based on basic region measurements, such as aspect ratio and relative magnitude, as well as its neighbouring colour relationships. Superposition of several separate decisions combined with the component's classification probability can then provide a comparative measure, indicating to which object model a colour component belongs.

6. Results and Discussion

Colour component tracking by means of kernel-based classification has proved to be a highly effective method for locating characteristic regions of both people and objects. The current

implementation is capable of allowing a user to interactively select several target colours to be tracked. Naturally this process can be automated to build models based on segmented or processed images. Figure 5 shows two examples where a shirt and a box have been selected for classification, though still images are not an ideal demonstration medium.

Figure 5: *Classification using unsegmented scene*

As seen in the right-hand image, errors can occur in classification when the target colour is visible in more than one place. Once the voting scheme has been completed, these outliers can be eliminated. Also note that not all colours available on each object were added to the model, thereby causing only a few pieces to be classified.

Currently several camera servers allow images to be transmitted to a processing client over a 100 Mbps LAN during testing. Input frame rate varies between 5 and 15 fps depending on network load. Processing involving feature extraction and classification operates at an average of approximately 4 fps on a 320 x 240 unsegmented colour image processed on a Pentium III 450 MHz (though not all processes have been optimised yet). Pyramid colour segmentation is performed using a CIE Lab image as the input. It was found that a threshold of between 8 and 15 for pyramid linking (See Section 3.1) yielded reasonable colour groupings.

Improved performance can be attained by using a motion segmented image as the input (Figure 5 shows an example). This reduces the processing required during pyramid colour segmentation (less pixels to process) and reduces the number of classifications necessary. In general, unsegmented input is only useful for scenes where continual environmental changes make segmentation unfeasible (e.g. reflective materials). On the other hand, the flexibility of the system to operate on either type of input may prove useful in applications such as tracking using a Pan Tilt Zoom or mobile camera.

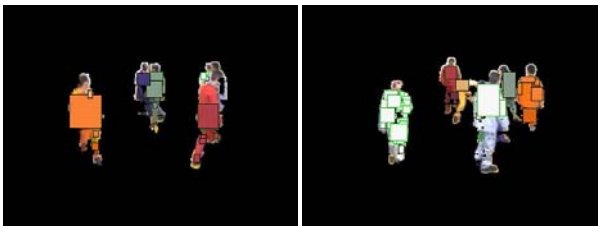

Figure 6: *Classification using motion segmented scene*

6.1. Issues to be resolved

One problem, currently being addressed, is that non-chromatic or large chromatic regions can make an exact target identification difficult. Generally, colours which are common to too many objects will cause ambiguity when used in object descriptors. A solution is to identify these colour components (by other features) and filter them out from further decision processes, not unlike removing a DC component from an electrical signal.

Another key issue is colour consistency between different cameras. Since the entire goal of the system is to be able to correctly identify targets between multiple views, this can be exceedingly complex when colour measurements from two cameras are decidedly different for the same object. In addition, not only will each camera present a varied representation of the same scene, but it is unlikely for them to be orientated identically, and therefore the environment's lighting configuration could also affect the way the camera perceives objects.

Testing has shown that certain vibrant colours are usually detected between views, even without correction. In the general case, however, a calibration scheme will be needed to ensure that colour features match between views. Several automatic methods have been reviewed which are aimed at providing colour constancy[1], based on certain assumptions (e.g. Lambertian reflectance), and independence from device gamma[3]. Most of these methods are aimed at offline processing for image retrieval systems and are therefore not well suited to real-time multi-camera environments. So far, the most promising solution seems to be to use a colour calibration object in order to determine a linear transformation between the cameras[4].

7. Conclusions

In conclusion, a realisable multi-camera implementation based on a distributed paradigm should prove to be a solid foundation for data exchange and synchronisation.

Approaching the tracking problem from a classification viewpoint in order to complement conventional tracking methods may prove a viable way of attaining a more robust person tracking system.

Using CIE Lab colour features allows good comparison in feature space. At present, pyramid segmentation shows the most promise in providing a breakdown of colour components both in terms of speed and quality.

A kernel-based classification approach was favoured since the features are already grouped in a unimodal fashion. The Kernel Adatron Support Vector Machine implementation is a simple method that ensures optimum classification while also shedding redundant features and allowing a compact class representation.

Although the present implementation can correctly identify targets between different cameras, the accuracy depends on lighting conditions and the perspective viewing angle of each camera. For this reason a more robust colour calibration method is needed to ensure colour constancy between multiple camera views.

Finally, a large benefit is that the system can classify targets without a motion segmented image, thereby extending its application to other areas including: identifying targets or locations with mobile cameras; and tracking targets with a Pan Tilt Zoom camera.

8. Acknowledgements

Many thanks to TSS Technology, De Beers and the NRF for their continued support of this research.

9. References

[1] Rizzi A., Gatta C., and Marini D. A new algorithm for unsupervised global and local color correction. *Pattern Recognition Letters*, 24(11):1663–1677, 2003.

[2] Jahne B. *Digital Image Processing*. Springer, New York, 1997.

[3] Finlayson G. and Xu R. Illuminant and gamma comprehensive normalisation in log rgb space. *Pattern Recognition Letters*, 24(11):1679–1690, 2003.

[4] Austermeier H., Hartmann G., and Hilker R. Colorcalibration of a robot vision system using self-organizing feature maps. *ICANN*, pages 257–262, 1996.

[5] Haritaoglu I., Cutler R., Harwood D, and Davis L. S. Detection of people carrying objects using silhouettes. Technical report, Computer Vision Laboratory, University of Maryland.

[6] Burt P. J., Hong T. H., and Rosenfeld A. Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Tran. On SMC*, 11(12):802–809, 1981.

[7] McKenna S. J., Jabri S., Duric Z., Rosenfeld A., and Wechsler H. Tracking groups of people. *Computer Vision and Image Understanding*, 80:42–56, 2000.

[8] Friess T., Cristianini N., and Campbell C. The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In *15th International Conference on Machine Learning*, 1998.