Object Tracking Using Piecewise Feature Clustering

Amol Ambardekar, Mircea Nicolescu and Monica Nicolescu
Department of Computer Science and Engineering
University of Nevada, Reno
U.S.A.
ambardek@cse.unr.edu, mircea@cse.unr.edu, monica@cse.unr.edu

ABSTRACT
Object tracking is a complex, yet essential task to be addressed in any video surveillance application. Many real-time techniques proposed in the literature rely on a frame-to-frame matching of objects. This paper describes a technique which takes into consideration the inherent temporal coherence that exists across frames, thus being able to robustly perform tracking while handling difficult situations such as object acceleration and partial occlusion. SIFT (Scale Invariant Feature Transform) approaches have been shown to perform well for object recognition, due to their robustness to noise, changes in illumination and viewpoint. In this work we propose to use a SIFT-based method for tracking image features across frames. Tracked SIFT features provide the displacement of each interest point in the image, which along with image coordinates and frame number constitute a feature vector. All feature vectors are added to a temporal buffer and clustered in order to identify and track coherently moving regions. The proposed clustering method uses an improved K-Means technique where K is determined using a CI (Confidence Interval) metric. We demonstrate our method in the context of a real-time traffic surveillance application.

KEY WORDS
Computer vision, object tracking, and clustering analysis

1. Introduction

Robust and efficient matching of image features or regions is an important prerequisite to many problems in computer vision. The success of higher level processes such as object recognition, classification or tracking depends heavily on the quality of image matching. A simple and fast approach to matching is through region correlation – however, this technique has little use in real world applications where changes in illumination, viewpoint or scale, as well as partial occlusions are common. Objects that undergo partial occlusion or even complete occlusion for a few frames pose particular challenges. Various techniques [1][2] that rely on the continuous appearance of the entire object for the purpose of recognition fail in this context. Employing local features in matching helps alleviate this problem and has led to successful approaches in many applications including object recognition, texture recognition, and image retrieval [3][4][5]. It has been shown that SIFT-based descriptors perform particularly well in this respect [6].

Visual object tracking appears as an essential component of many applications in videoconferencing, vision-based surveillance and monitoring, or image and video understanding. A wide range of methods have been presented in literature, targeting the problem of visual tracking. One of the most common methods – with many variations – is blob tracking, which usually emphasizes the difference between the current image observation and a model of the background. Such approaches assume that the image regions (blobs) extracted in this manner correspond to the actual foreground objects in the scene [7]. These algorithms have particular difficulty handling shadows, non-stationary parts of the background and occlusions.

A closely related approach to blob tracking is based on tracking active contours representing the boundary of an object. Active contour-based tracking algorithms [8] represent the outline of moving objects as contours, which are updated dynamically in successive frames. These algorithms have drawbacks, such as their tracking accuracy is limited by a lack of precision in the location of the contour.

Feature-based tracking algorithms perform tracking of objects by extracting elements, clustering them into higher level features and then matching the features between images. Feature-based tracking algorithms can further be classified into three subcategories: global feature-based methods, local feature-based methods, and dependence-graph-based methods. Global feature-based methods rely on features like color, centroid, and perimeters [9]. The features used in local feature-based algorithms include line segments, curve segments, and corner vertices [10]. In general, feature-based tracking methods can adapt successfully and rapidly to allow real-time processing and tracking of multiple objects with the exception of dependence-graph-based methods [11].

In a different direction of research, recent efforts employ statistical models for representing video content. Each frame is represented in feature space (e.g. color, texture)

via models such as a GMM model. Tracking across frames is then achieved by extended models, such as HMMs [12]. Greenspan *et al.* proposed a piecewise GMM model for probabilistic video modeling [13]. The video sequence is divided into temporal buffers which are segmented using GMM modeling. The results show a great potential for considering piecewise clustering as a viable option for tracking. However, their approach considers an entire image for modeling and therefore is less suitable for real-time application. A further difference between our approach and [13] is that color information is used in the feature vectors described in [13] instead of image velocity components. The results illustrate primarily objects without texture, where textured objects might indeed be difficult to handle by their algorithm.

Our approach proceeds with a stage of background modeling and foreground extraction. SIFT features are extracted at points of interest inside the regions detected as foreground, then used for matching from one frame to the next. Each matched pair of SIFT features provides a displacement vector, equivalent to the image velocity of the point of interest. Within a temporal buffer we collect feature vectors that consist of this displacement, the image coordinates and the frame number. The frame number is normalized by the size of the temporal window considered. All feature vectors are processed using an improved K-Means algorithm to find the most dominant clusters, where the number of clusters is determined using a novel cluster statistics metric. The contribution of this work is the development of a robust, real-time SIFT-based approach for object tracking that enforces the inherent temporal coherence across image frames, therefore handling difficult situations caused by significant object acceleration and partial occlusion.

The remainder of the paper is organized as follows: Sections 2 describes the piecewise feature clustering algorithm, Section 3 presents and discusses our experimental results and Section 4 provides the conclusions and future extensions of this work.

2. Piecewise Feature Clustering

2.1. Background Modeling and Foreground Detection

Our method proceeds by detecting foreground regions in each image frame. This part of the system detects the moving objects (blobs) as simple image regions, without any assumption about the objects that they represent – e.g., vehicle or non-vehicle.

In general, we assume that a static camera is used to capture the video in our application domain. However, due to inherent changes in the background itself, such as wavering trees and flags, rain or water surfaces, the background of the video may not be completely stationary. These types of backgrounds are referred to as quasi-stationary backgrounds. For our application purposes, we impose as requirements for the background modeling process that it is able to handle such quasi-stationary backgrounds, while being fast enough for real-time operation.

We propose using a simple recursive learning method to model the background in order to satisfy these requirements. We employ an adaptive background model for the entire region of awareness, and for segmenting the novel objects that appear in foreground. Our approach involves learning a statistical color model of the background, and process a new frame using the current distribution in order to segment foreground elements.

The algorithm has three main stages: learning, classification and post-processing. In the *learning* stage, we establish the background model using recursive learning [14]. We use all channels (red, green, and blue) of a color image for increased robustness. We assume that the pixel values tend to have Gaussian distribution and we estimate the mean and variance of the distribution using consecutive frames. In the *classification* stage, we classify the image pixels into foreground and background pixels based on the background model. As we assume a fixed camera position, we also specify a region of interest (ROI) in the scene where objects are expected to appear (the image regions corresponding to the road). This is done in order to further diminish the effects of quasi-stationary backgrounds such as moving tree branches. Another advantage of using a ROI template is that it reduces the overall area to process for foreground object detection, hence speeding the algorithm.

We calculate the background model by updating the values of mean and variance in the background model for each pixel in the image. Mean and variance are calculated by using the following formulas:

$$mean(x,y) = (1 - LR) \times mean(x,y) + LR \times I(x,y)$$

$$var(x,y) = (1 - LR \times LR) \times var(x,y) + (LR \times (I(x,y) - mean(x,y))^2$$

where LR is the learning rate.

In the *classification stage*, the pixel is classified as foreground if it satisfies the following inequality:

$$|I(x,y) - mean(x,y)| > T \times \sqrt{var(x,y)},$$

where, T is the threshold that determines the amount of foreground to be detected.

Finally, in a *post-processing* stage, we group the detected foreground pixels into connected components and we create a list of blobs (foreground objects) associated with the current image frame.

2.2. Estimation of a Feature Vector

We propose to use a five-dimensional feature vector $v = (vx, vy, x, y, t)$, where vx and vy are the image velocity components of an interest point, x and y are the coordinates of the interest point in the image plane, and t

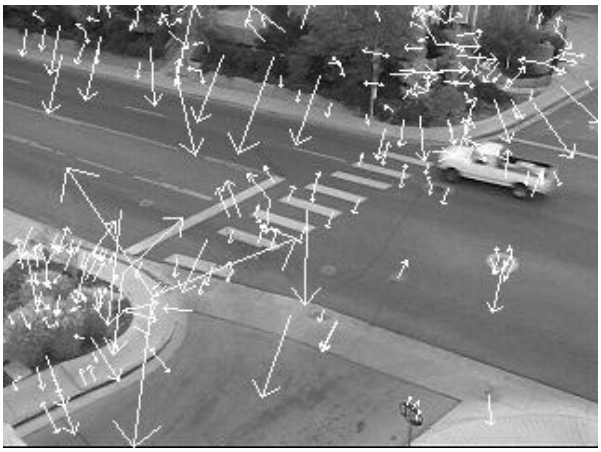

Fig. 1. SIFT features detected in an image.

is time (frame number) normalized with respect to the temporal window considered.

The interest points are determined by finding local extrema in the Difference-of-Gaussian (DoG) scale-space. The SIFT features are extracted for all such interest points detected. Each feature is classified as corresponding to one of the foreground blobs or to the background. For every feature corresponding to a particular blob, we try to find a match with the features found in the next frame within a neighborhood window. We consider that a match is found between two SIFT features when the Euclidean distance between the best match feature and the next best match feature is more than 60%. All the matching features for a corresponding blob are averaged to find the displacement vector. The new position of the blob is determined by the previous position of the blob and the displacement vector. Fig. 1 shows an example of detected SIFT features in an image. The arrows represent the direction and position of the SIFT features detected. Fig. 2 shows the pseudo-code of the algorithm used for finding the best match SIFT features in consecutive frames. If match is found it returns the displacement vector between the matched features.

```
Match(f1, PrevFeatureList(PF))
-matchScore1=99999
-matchScore2=99999
-For each SIFT feature f2 in the PF (Feature
list of the previous frame)
  matchScore= match(f1, f2)
  If matchScore<matchScore1 then
    matchScore2=matchScore1
    matchScore1=matchScore
  Else If matchScore<matchScore2 then
    matchScore2=matchScore
-If matchScore1<0.6*matchScore2 then
  SIFT features f1 and f2 match
  -Return the displacement vector between f2
  and f1
-Else
  Return displacement vector=0
```

Fig. 2. Algorithm for calculating the displacement vector.

2.3 Clustering

Feature vectors corresponding to the foreground blobs are inserted into the temporal buffer. For all our experiments we used a temporal window of N=10 frames. After N frames have elapsed, all the feature vectors in the buffer are subjected to a clustering analysis process. Fig. 3 depicts the block diagram of the piecewise feature clustering stage for object tracking, which also estimates the number of tracks (clusters). The subsequent N frames provide another set of clusters, which are matched using a nearest neighbor algorithm to create time consistent tracks. Fig. 4 shows the pseudo-code of the algorithm used for piecewise feature clustering.

K-means [15] is one of the simplest unsupervised learning algorithms that address the well-known clustering problem. In order to classify a given data set into a certain number of clusters (assume *k* clusters) fixed a priori, the algorithm defines *k* centroids, one for each cluster. These centroids should be placed in such a way so that they are as far away from each other as possible. K-means may produce a poor clustering as the final result highly depends on initial cluster choices.

A simpler way to prevent poor clustering due to inadequate seeding is to modify the basic *k*-means algorithm to obtain the *improved k-means algorithm* [16].

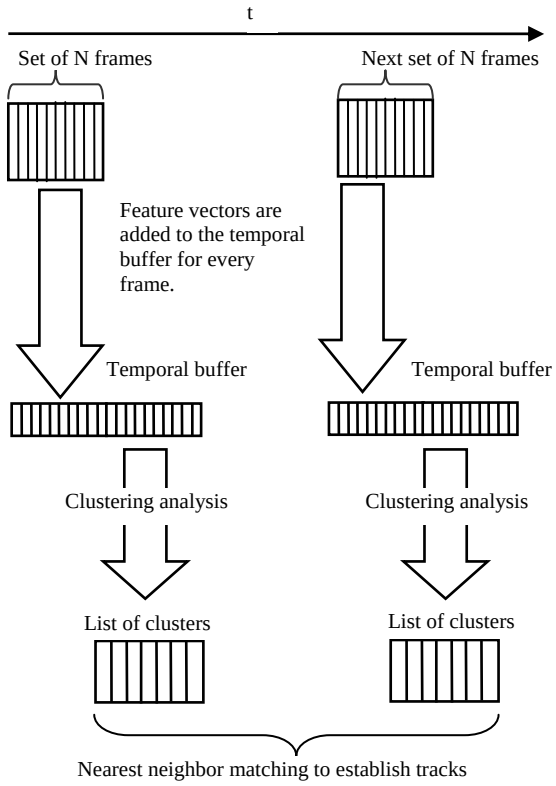

Fig. 3. Block diagram of the piecewise feature clustering.

```
PiecewiseClustering(PrevFeatureList(PF), Frame
Number(FN), Temporal Buffer Length(N),
PrevClusterList)
-Calculate the SIFT features for the current
frame and add them to the CurFeatureList
-For each SIFT feature f1 in the PrevFeatureList
(Feature list of the previous frame)
  DisplacemntVector(DV)=Match(f1,PF)
  If |DV|>0 then
    FeatureVector=(DVx, DVy, x, y, MOD(FN, N))
    -Insert the FeatureVector into the
    Temporal Buffer
-If MOD(FN, N)=0
  -Do Clustering anlysis of the feature
  vectors in Temporal Buffer using
  Improved K-Means
  -Add the clusters to the CurClusterList
  -Do nearest neighbor matching for the
  clusters in CurClusterList and
  the clusters in PrevClusterList
  -Clear the temporal buffer
PrevClusterList=CurClusterList
PrevFeatureList=CurFeatureList
```

Fig. 4. Algorithm for the piecewise feature clustering.

In *improved k-means algorithm*, we start with a large number of uniformly distributed seeds in the bounded 5-dimensional feature space, but we reduce them considerably by eliminating those that are too close to one another. The test threshold is the average distance between seed vectors, which may be computed from a sample of seeds. Thus, we obtain a smaller number *K* of initial seeds that are uniformly distributed. Next, we assign each of the feature vectors to a seed with minimum distance and then eliminate the empty clusters and any clusters of a size less than *p*. Iteration now converges quickly to a large number of relatively small clusters. The closest ones are merged until stopping criteria are met or until there is only one cluster remaining. The XB (Xie-Beni) measure [17] is one of the metrics used in previous approaches as a stopping criterion. However, the Xie-Beni clustering validity measure used with the improved k-means algorithm does not always estimate the correct number of clusters. We introduce a new CI metric (Confidence Interval metric) that improves the clustering validity and gives optimal clusters in most cases.

The CI metric uses the covariance of the clusters and tries to find the variance in the direction of the line joining the two cluster centers:
if $C * (CI12 + CI21) >= |D|$
then merging of the clusters continues.

Here C is the constant which controls the overlap, $CI12$ is the confidence interval of cluster 1 in the direction of cluster 2, $CI21$ is the confidence interval of cluster 2 in the direction of cluster 1, and D is the Euclidean distance between the cluster centers.

After eliminating small clusters, the improved K-means algorithm proceeds with merging. The conventional technique is a simple and effective one, merging the two clusters with the minimum distance between their cluster centers. But this strategy might fail in the case where two clusters are very close but still very compact and there is almost no overlap between them. In this case merging such clusters leads to erroneous results, because the minimum distance merging technique does not take into consideration the spread of individual clusters. To overcome this issue, we propose to use a confidence interval overlap measure (CI_{OM}) in order to choose the best neighboring clusters which naturally belong to one cluster. The confidence interval overlap measure is calculated by the following formula for clusters 1 and 2:

$$CI_{OM} = (CI12 + CI21) / |D|.$$

This ratio is calculated for all combinations of clusters and the two clusters for which the ratio is maximal are chosen to be merged. Using this merging technique along with the minimum distance merging approach has been shown to produce very good results in most of our experiments.

3. Experimental Results

We used the proposed approach for tracking vehicles in a vision-based traffic surveillance application. Fig. 5 shows an example of a tracked vehicle, where the green square represents the bounding box for the detected foreground region and the red arrow represents the displacement vector calculated using SIFT feature matching. Fig. 6 shows an example of successful object tracking using piecewise feature clustering, where the red circles denote the clustered interest points. Their common color implies that they are in the same cluster (track) in this case. The vehicle shown in Fig. 6 was tracked successfully for more than 40 frames. Fig. 7 illustrates an example where *object acceleration* is present – a vehicle is turning and consequently the image velocity incorporated in the feature vector changes significantly. The successful tracking of the vehicle shows that our proposed algorithm is able to work even with such changes in the image velocity (which may be quite significant when the vehicle is close to the camera). Fig. 8 shows an example of successful tracking of three vehicles at the same time. The tracks for these vehicles are shown in three different colors, representing the separate tracks correctly identified. A potentially difficult situation should be noted in this example: the red circles overlap the green circles as the vehicle corresponding to red circles has arrived at the same place in the image where the vehicle corresponding to the green circles was a few frames ago. Despite this overlapping in the x and y coordinates, the algorithm successfully distinguished the tracks due to the inherent separation in the higher, five-dimensional feature space used, and that includes the normalized time component. Fig. 9 illustrates an example of object tracking in the presence of *partial occlusion*, where the green circles show that the car was successfully tracked although it passes behind parked cars and trees. The red circles represent another car that is currently out of this frame.

Fig. 5. An example foreground object detection (the green bounding box) and displacement vector calculated using SIFT feature matching (the red arrow).

Fig. 6. An example of a successful object tracking using piecewise feature clustering.

Fig. 7. An example of object tracking while the object (vehicle) was turning.

Fig. 8. An example showing three objects being tracked simultaneously. The circles of three different colors represent three different tracks.

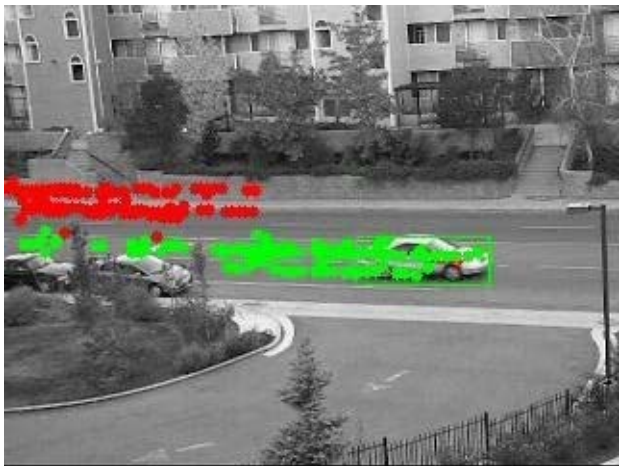

Fig. 9. An example of object tracking in the presence of partial occlusion.

4. Conclusions and Future Work

In this paper we have described a novel approach for object tracking using unsupervised clustering, through an improved k-means algorithm with a confidence interval metric. Our experimental results demonstrate the contribution of this work, as a robust, real-time SIFT-based approach that enforces the inherent temporal coherence across image frames, therefore handling difficult situations caused by significant object acceleration and partial occlusion.

Although the proposed algorithm works well in general, its performance degrades when the scene contains a large number of moving objects. In the presence of many blobs that are close to each other, the method tends to group these clusters together. This situation sometimes leads to an increased variance for the newly grouped cluster which attracts more distinct clusters to group with it, ultimately

grouping all the feature vectors into a single cluster. The reason is that the current approach uses the piecewise clustering technique where clustering is done separately for each piece and the cluster matching is done on the nearest neighbor basis. The problem of over-merging can be addressed if the information from the previous piece can be used while clustering a new piece of video. A Kalman filter can be used to estimate the position of cluster centers and these cluster centers can be used as seeds for clustering during the next set of frames. GMM can also be used as a viable option to the improved k-means algorithm employed in this paper. The accuracy of the clustering can also be increased if the SIFT features are augmented to the feature vector used for clustering.

Acknowledgements

This work was supported by the Office of Naval Research award N00014-06-1-0611.

References

[1] E. Belogay, C. Cabrelli, U. Molter, R. Shonkwiler, "Calculating the Hausdorff distance between curves," *Information Processing Letters*, v. 64, pp. 17-22, 1997.

[2] A. Thayananthan, B. Stenger, P.H.S. Torr, R. Cipolla, "Shape context and chamfer matching in cluttered scenes," *CVPR*, v. 1, pp. 127-133, June 2003.

[3] V. Ferrari, T. Tuytelaars, and L. Van Gool, "Simultaneous Object Recognition and Segmentation by Image Exploration," *Proc. Eighth European Conf. Computer Vision*, pp. 40-54, 2004.

[4] S. Lazebnik, C. Schmid, and J. Ponce, "Sparse Texture Representation Using Affine-Invariant Neighborhoods," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 319-324, 2003.

[5] K. Mikolajczyk and C. Schmid, "Indexing Based on Scale Invariant Interest Points," *Proc. Eighth Int'l Conf. Computer Vision*, pp. 525-531, 2001.

[6] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 27(10), pp. 1615-1630, 2005.

[7] D. Magee, "Tracking multiple vehicles using foreground, background and motion models," *Proceedings of ECCV Workshop on Statistical Methods in Video Processing*, 2002.

[8] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Toward robust automatic traffic scene analysis in real-time," P*roceedings of International Conference on Pattern Recognition*, 126–131, 1994.

[9] R. Polana and R. Nelson, "Low level recognition of human motion," *Proceedings of IEEE Workshop Motion of Non-Rigid and Articulated Objects*, pp. 77–82, 1994.

[10] B. Coifman, D. Beymer, P.McLauchlan, and J. Malik, "Areal-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Res.: Part C*, v. 6(4), pp. 271–288, 1998.

[11] T. J. Fan, G. Medioni, and G. Nevatia, "Recognizing 3-D objects using surface descriptions," *IEEE Trans. Pattern Recognit. Machine Intell.*, v. 11, pp. 1140–1157, 1989.

[12] C. Bregler, "Learning and recognizing human dynamics in video sequences," *IEEE Computer Vision and Pattern Recognition (CVPR)*, June 1997.

[13] H. Greenspan, J. Goldberger, A. Mayer, "Probabilistic space-time video modeling via piecewise GMM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26(3), pp. 384 – 396, March 2004.

[14] A. Tavakkoli, M. Nicolescu, G. Bebis, "Robust Recursive Learning for Foreground Region Detection in Videos with Quasi-Stationary Backgrounds," *Proceedings of International Conference on Pattern Recognition*, pp. 315-318, 2006.

[15] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of 5th Berkeley Symposium*, v. 1, pp. 281–297, 1967.

[16] Carl G. Looney, "Interactive clustering and merging with a new fuzzy expected value," *Pattern Recognition*, v. 35, pp. 2413-2423, 2002.

[17] X. L. Xie and G. Beni, "A Validity Measure for fuzzy Clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, v. 13(8), pp. 841-847, 1991.