MVA2007 IAPR Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, JAPAN

Action Recognition of Insects Using Spectral Clustering

Maryam Moslemi Naeini[1], Greg Dutton[2], Kristina Rothley[2], and Greg Mori[1]
[1]School of Computing Science,[2]School of Environmental Management
Simon Fraser University, Burnaby, BC
{mmoslemi, gdutton, krothley, mori}@sfu.ca

Abstract

We propose a technique to recognize actions of grasshoppers based on spectral clustering. We track the object in 3D and construct features using 3D object movement in segments of video which discriminate between different classes of actions. We sample from these feature vectors and compute the eigenvalues and eigenvectors of affinity or similarity matrix. Then, we perform K-means algorithm to cluster component from each of dominant eigenvectors of the affinity matrix. These dominant eigenvectors are embedding coordinate of video segments in our embedding space. We experimented with our method on a noisy track of one insect to validate our approach.

1 Introduction

Biologists are interested in analyzing the behavior of individual or colonies of insects. Actions of insects are studied under variation of environment variables like illumination and temperature during hours of video. It is a hard task to watch hours of video to see what they are doing; therefore, building an automated system that could track and classify insect actions is very useful. In this work, we present a novel application of methods in the area of computer vision to this problem in order to track and recognize insect actions.

In our work, the behavior of grasshoppers is analyzed. The behavior consists of movement and different actions such as standing still, walking, and jumping. We use a simple tracking algorithm for an individual insect. We create a 3D track of the insect, via 2D tracking in video from each of a pair of calibrated cameras, and then performing triangulation.

Given this 3D track of the grasshopper in a long video sequence, we wish to find interesting actions. We pose this as a clustering problem, finding groups of frames from the video that correspond to similar actions. We define a similarity between groups of frames based upon a motion cue. The frames of the video are then clustered using spectral clustering [6]. Since our application involves long video sequences, we employ the Nystrom extension [5], a sampling technique that can be used in computing eigenvectors for spectral clustering. Our contribution is developing an application of spectral clustering to clustering actions in long video sequences, and a particular method of sampling for the Nystrom extension to cope with rare actions.

The rest of this paper is organized as follows: In section 2 we review previous work. We present our main approach for tracking and clustering algorithm in section 3. The experimental results are given in section 4 and we conclude our paper in section 5.

2 Previous Work

In our work we demonstrate that the Nystrom extension can be applied to clustering problems with a large number of frames, and rare activities. In related work, Zhong et al. [10] build a co-occurance matrix over vector quantized spatial motion features, and perform clustering using this. They use this to find unusual events in long video sequences. In their case, the co-occurance matrix is sparse, and eigensolving is efficient.

Other work on automatically analyzing the behaviour of animals includes Balch et al. [1], who have developed methods for tracking multiple ants, and suggest the use of Hidden Markov Models for analyzing their behaviours.

Belongie et al. [4, 2] analyze the behaviour of mice in caged environments by first tracking and then computing spatio-temporal patch features.

Also related is the work of Subramanya et al. [7] on analyzing GPS and wearable sensor data, in this case on human subjects.

3 Main algorithm

In this section we present two main parts of our algorithm. First we discuss the method we use to find the 3D track of the object; next, we explain how to cluster the 3D information into different action classes.

3.1 Stereo Tracking

We first start by tracking the grasshopper using a stereo camera setup. We setup two cameras and calibrate them by employing Bouguet's camera calibration toolbox [3]. Tracking this insect is difficult due to its very small size and its color changes as it is walking. In addition, the insect makes occasional jumps which are so fast that sometimes it is very hard to see. To overcome these difficulties, we used a fixed painted background and image differencing to detect the object, instead of tracking

Funding provided by CFI and BCKDF as part of the SDATS project. Kris Rothley received funding from the NSERC Discovery Grant, and cooperation from Terasen Gas

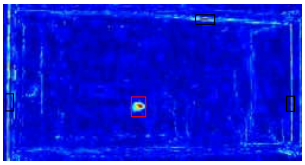

Figure 1. Difference image before smoothing

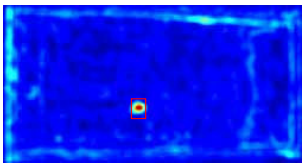

Figure 2. Difference image after smoothing

Figure 3. Experimental enviroment

algorithm based on color histogram or motion models of the target, which tend to oversmooth the jumps.

We employ a background subtraction technique to track the object. We smooth the difference image using a Gaussian filter and take the pixel of maximum sum of red, green and blue values as center of the location of the insect. The difference image is shown before and after smoothing in Figure 1 and Figure 2. The noise is mainly because of presence of slight changes in the background, for example in the border of the cage. Our background image is set to be the average of all the frames up to time t to make the algorithm more robust to slight changes in illumination or other variations in background image [8]. Employing these techniques we track the object in each of the cameras separately. By doing stereo triangulation we compute the 3D location of the object.

3.2 Motion Features

Using the 3D track enables us to specify the location of the object at each frame. The next and more challenging step is to cluster different actions of the insect such as jumping, walking and standing still using its movements between frames. Although the tracker always points to the object, the location information is noisy. This noise is more when the object is not moving which makes the clustering task more difficult.

We define a set of motion features based upon this tracker output which we will use to describe grasshopper tracks. The motion features will be clustered using spectral clustering (described in the following section). Obtaining a good motion feature is a critical task that impacts the quality of clustering. The word 'good' means that the feature should be as different as possible between the actions which are in two separate classes, and as similar as possible between actions within a same class.

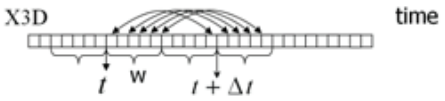

Figure 4. Feature vectors

Constructing the motion feature is a crucial part and since we are using only the 3D position we smooth that using a Gaussian filter to remove the noise in the tracker output. Then for each non-overlapping window of size W of 3D position of the object we compute the difference between x_t (location of grasshopper in 3D at time t) and $x_{t+\delta_t}$ for each of the frames in this window. This feature is illustrated in Figure 4. So our feature vector V_t for window of size W of 3D coordinates sequence in time will be:

$$V_t = \{|x_k - x_{k+\delta_t}| : k = t, t+1, \ldots, t+W\} \quad (1)$$

We will perform clustering on these W dimensional feature vectors. The motivation for employing these features instead of the gradient of x is to eliminate noises in the tracker by using δ_t frames instead of 1 frame used for gradient computation. So when the grasshopper is standing still but the tracker is noisy, this feature is designed to smooth the gradient computation.

3.3 Spectral Clustering

In spectral clustering, an affinity matrix W that indicates the similarity between each pair of data is constructed. An entry W_{ij} of this matrix stores the similarity between nodes i and j. In our work nodes i and j are blocks of frames which are described using our motion feature. The data is clustered by analyzing the eigenvalues and eigenvectors of this matrix. In this work, we employ the normalized cuts method [6] of spectral clustering. We compute the leading eigenvectors of W, and cluster data points using kmeans in the embedding space given by these eigenvectors.

When dealing with large amount of data, computing eigenvalues and eigenvectors of a large matrix is an expensive task. For our application, there will be thousands of nodes, so constructing, storing, and computing the eigenvectors of the matrix W will be intractable. To overcome this limitation, we apply the Nystrom extension [5] which provides a method for extrapolating eigenvectors computed on a portion of W to the entire matrix.

Following the notation in Fowlkes et al. [5], given an N by N affinity matrix W,

$$W = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \quad (2)$$

where A is an n by n sub-matrix of W containing a set of randomly chosen sample points. If $n << N$, eigenvectors U of A can be computed efficiently, and then extended as $\bar{U}$ to the entire matrix W by:

$$\bar{U} = \begin{bmatrix} U \\ B^T U \Lambda^{-1} \end{bmatrix} \quad (3)$$

where Λ is the diagonal matrix of eigenvalues of A.

An important point for our application is that if rare activities exist, and are not randomly chosen in matrix A, the extended eigenvectors given by (3) will not be accurate. In the next section we describe how we handle this problem automatically.

3.4 Clustering Actions

In this section we provide the details of how we apply spectral clustering to our task of clustering the actions of grasshoppers.

After constructing features in section 3.2 we compute the distance d_{ij}, between nodes i and j using Euclidean distance, and the weight between nodes is:

$$W_{i,j} = exp(-\frac{d_{ij}^2}{\sigma_i \sigma_j}) \quad (4)$$

We apply local scaling instead of a fixed scaling [9]. In this method, the scaling factor σ is a function of distance between nodes. Our choice for σ is :

$$\sigma_i = d(V_i, V_k) \quad (5)$$

V_k is the k_{th} neighbor of node i. k in this formula should not be very large or small. If it is large, weights between all the points become large and if it is small, most of the nodes get a low similarity. In our experiments, k is set to 10.

The reason for not using the simple Gaussian function is that the distances between clusters are not the same and if we have a tight cluster within a background cluster and use a constant σ, it leads to weights that may not describe the real similarity between nodes.

We then randomly choose data samples to construct the matrix A for use in the Nystrom extension to perform spectral clustering on W. However, we note that without samples for small clusters, such as jumps, the Nystrom extension in (3) will be inaccurate. Therefore, we augment the set of random data samples with a fixed number r of data points. These points are chosen based upon affinities in B, finding samples which are furthest away from the originally randomly chosen samples. In our experiments, we found the results to be insensitive to the setting of this parameter r.

We then compute eigenvectors and eigenvalues for this augmented matrix $\hat{A}$, and use the Nystrom extension to extend these eigenvectors to the entire matrix W. Finally, we perform k-means clustering on the resulting embedding coordinates.

We summarize our action recognition algorithm below:

1. Construct the graph using features in section 3.2.
2. Sample from the nodes randomly.
3. Augment these samples with the $r = 4$ furthest nodes from these samples.
4. Compute distance between samples $D^A_{nsamp \times nsamp}$ using L_2 distance.
5. Compute distance between samples and rest of the nodes, $D^B_{nsamp \times nrest}$ using L_2 distance.
6. Sort rows of D^A matrix and choose the j_{th} column as σ_A compute affinities A, the between samples matrix using (4).
7. Sort columns of D^B matrix and choose the j_{th} row as σ_B compute affinities B between samples and rest of the nodes Matrix using (4)
8. Compute the eigenvalues of affinities using one shot technique in [5].
9. Use the K largest eigenvectors $E = [E_1 E_2 ... E_k]$
10. Cluster rows of matrix E which are the embedding coordinate in K-dimensional embedding space using K-means algorithm

4 Experimental Results

We tested the presented algorithm on 16500 frames of a video of one grasshopper. Figure 3 shows how we set up the cameras for our experiments. We use two cameras and calibrate them by calibration toolbox in Matlab[3]. Then we apply the background subtraction to get the 2D track in each camera separately and get the 3D coordinate using the triangulation procedure in this toolbox. We smooth this track by a Gaussian filter and divide the track into non overlapping windows of size 5, $W = 5$, for each frame in this window we compute the difference between 3D position of x and that is the feature vector or nodes of the graph.We also set δ_t=5 for our experiments. We build this graph once and run experiments with different number of cluster centers.

We manually supply ground truth labelling of these frames into 3 classes of distinct actions – standing still, walking and jumping.

We run our code 200 times for each value of number of clusters. The reason for this is the randomness in sampling of Nystrom method and initialization of K-means algorithm. The number of samples are 100.

Correctness of clustering is measured by the purity of a cluster. To compute the correctness, in each round we find the number of frames for each action that has been fallen to each cluster. Then clusters are labeled with the action that has maximum number of frames in them. We do it for each cluster then add the number of frames of an action in clusters that are labeled with the same label and divide this sum by the total number of frames of each

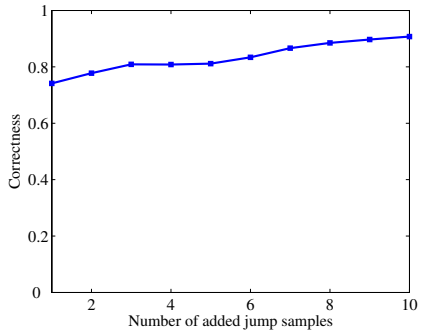

Figure 5. Effect of Number of added jump samples on performance of detecting jump actions.

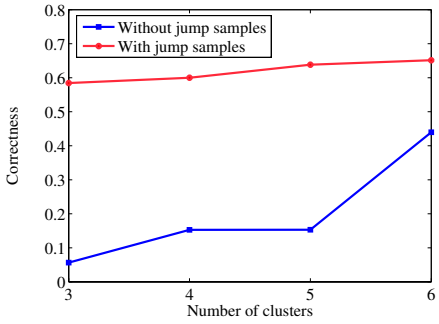

Figure 6. Impact of sampling from jumps on performance. Curves show correctness of frames labelled as jumping with and without samples from this class.

action. This number will be the fraction of actions that are correctly classified.

The average of the correctness is shown for each of them in Figure 7 with respect to number of clusters. The plot shows this fraction for each action and for all of them together. As it is shown in this figure our overall performance is above 80 percent and the graph is almost smooth for $K > 5$.

Figure 6 shows the importance of having samples from rare activities. In our experiments jump frames are rare and their features are very different from walking and standing still. We checked in our experiments whether the jumps are sampled or not and plotted the correctness of recognized jump frames in both cases. As it is shown in Figure 6 there is a big change if we do not sample jump frames. In this case the computed eigenvectors which are the embedding coordinate will not lead to a good clustering because we estimate the eigenvectors of the whole affinity using them and if there were no samples of the unusual actions the Nystrom extension will not accurately reconstruct the eigenvectors.

We also did experiments using different values of r to analyze the effect of this parameter on the performance of the algorithm. As it can be seen in Figure 5, having more samples could result in a slightly better performance but the method is relatively stable for different values. More importantly, if we do not have any samples from the rare actions we cannot recognize them correctly.

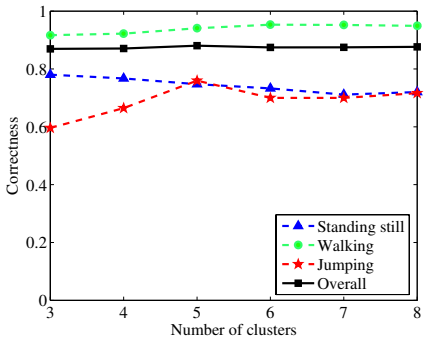

Figure 7. Impact of number of clusters on performance

5 Conclusion

In this paper we have developed features using 3D track of the object and applied the spectral algorithm to recognize actions of the object. It uses samples points from the data to cluster all of it and this will improve the performance. We show how to use the Nystrom extension in a problem that involves small clusters, and that the naive random sampling will have a substantial effect on performance on these clusters.

References

[1] T. Balch, Z. Khan, and M. Veloso. Automatically tracking and analyzing the behavior of live insect colonies. In *Fifth International Conference on Autonomous Agents*, 2001.

[2] S. Belongie, K. Branson, P. Dollar, and V. Rabaud. Monitoring animal behavior in the smart vivarium. In *MB*, 2005.

[3] Camera Calibration Toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[4] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[5] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), February 2004.

[6] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[7] A. Subramanya, A. Raj, J. Bilmes, and D. Fox. Recognizing activities and spatial context using wearable sensors. In *Conference on Uncertainty in AI (UAI)*, 2006.

[8] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 255–261, 1999.

[9] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Proceedings of NIPS 2004*, 2004.

[10] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004*, pages 819– 826, June 2004.