Genetic Algorithms for Automatic Classification of Moving Objects[*]

Omid David-Tabibi
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
mail@omiddavid.com

Nathan S. Netanyahu[†]
Dept. of Computer Science
Bar-Ilan University
Ramat-Gan 52900, Israel
nathan@cs.biu.ac.il

Yoav Rosenberg
ProTrack Ltd.
P.O. Box 34384
Jerusalem 91343, Israel
yoav@protrack.co.il

Moshe Shimoni
ProTrack Ltd.
P.O. Box 34384
Jerusalem 91343, Israel
moshe.shimoni@gmail.com

ABSTRACT

This paper presents an integrated approach, combining a state-of-the-art commercial object detection system and genetic algorithms (GA)-based learning for automatic object classification. Specifically, the approach is based on applying weighted nearest neighbor classification to feature vectors extracted from the detected objects, where the weights are evolved due to GA-based learning. Our results demonstrate that this GA-based approach is considerably superior to other standard classification methods.

Categories and Subject Descriptors: I.2.6 [Artificial Intelligence]: Learning—*Parameter learning*

General Terms: Algorithms.

Keywords: Genetic algorithms, parameter tuning, computer vision, automatic object recognition

1. INTRODUCTION

Protrack is a company developing and selling products for video surveillance. A major application developed in the company is a *far-field* outdoor intrusion detection system based on a real-time video motion detection technology for a *scanning camera*. The system alerts for any moving object in the scanned area. The algorithm is robust to camera noise and can filter back and forth motion such as trees swaying in the wind. However, the main remaining gap concerns the ability to distinguish between significant and insignificant motion (e.g., alerting any human motion while filtering out animal motion) and to detect humans in an area where cars are usually moving and vice versa. For this reason, automatic classification of (detected) moving objects is needed. The main requirement is to classify according to the following four categories: "walking man", "crawling human", "animal", and "car".

[*]The support of the VULCAN Consortium under the Israeli Ministry of Industry, Trade and Labor is gratefully acknowledged. A full paper version is currently underway.

[†]Nathan Netanyahu is also affiliated with the Gonda Brain Research Center at Bar-Ilan University, and the Center for Automation Research, University of Maryland at College Park, MD 20742 (email: nathan@cfar.umd.edu).

In this paper we briefly describe how ProTrack's module detects objects and extracts their feature vectors. We then discuss how these feature vectors can be used for standard object classification, as well as classification due to GA-based learning. Our results demonstrate a significant improvement due to the latter.

2. BACKGROUND

Numerous approaches have been proposed for object classification. Some methods draw on shape information for object detection in still images [4, 6, 8]. However, these methods may require large datasets for learning and considerable CPU power. Also, false positives may pose a problem since the detection is performed on each video frame. Other object classification methods [1, 2] in far-field video that are based on scene modeling assume a static camera, i.e., they are not applicable to a scanning camera.

Similar to [1, 2], ProTrack's method is based on classifying an object based on its motion pattern and various shape parameters. The input to the classifier is the low-level data generated by the motion detection algorithm. In contrast to common motion detection algorithms, our motion detection is based on deploying a net of *micro-trackers* in the video image, where each micro-tracker computes the backward path along the image sequence of a 5×5 pixel patch in the current video frame. Given the image patch center location $P(t)$ for the current video frame t, the computed backward path contains the patch locations $P(t-1), ..., P(t-N)$ in the previous video frames $t-1, ..., t-N$, respectively. When a micro-tracker computes a consistent backward path of an image patch with an overall translation above a certain threshold (e.g., 6 pixels), it is considered to be part of a moving object. Using standard clustering techniques, the micro-trackers are grouped into moving objects (see Figure 1).

The paths computed by the micro-trackers contain valuable information about the motion pattern of different parts of the detected moving object. This information can be used to distinguish between rigid objects (like cars) and non-rigid objects (like walking humans).

The object classification algorithms presented below have a classic structure. A feature vector, which is a compact representation of the essential information about a moving object, is extracted for each moving object. A learning dataset containing samples of each motion category is then built and used in the learning stage to train the system and compute a classification rule based on the above feature vectors.

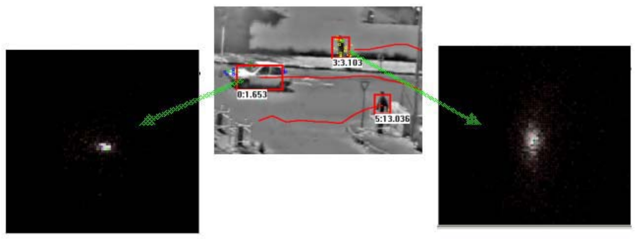

Figure 1: An example of micro-tracker variance for human and car.

For each detected object, the following feature values are calculated: velocity, acceleration, acceleration rate, shape ratio, velocity to height ratio, object size, average velocity velocity projection, average velocity along the x-axis, average velocity along the y-axis, and nine moment values (with respect to velocity variances), i.e., a total of 19 feature values. Given a 19-feature set of values, the classifier will issue one of the four categories: “walking human”, “crawling human”, “animal”, or “car”.

3. CLASSIFICATION USING GAUSSIAN MIXTURE MODEL

We first applied standard classification based on Gaussian mixture model [5], assuming that the features are distributed according to a multivariate Gaussian distribution for every object class. Given the training set, the mean vector M_i and the covariance matrix COV_i are estimated for each object category C_i. Given the set F_i of all the feature vectors belonging to class C_i, the elements of the mean vector M_i and the covariance matrix COV_i are estimated by:

$$M_i^k = E[F_i^k],$$

$$COV_i^{k,l} = E[(F_i^k - M_i^k)(F_i^l - M_i^l)],$$

where k, l, refer respectively to the k-th, l-th features of a given feature vector.

In the testing phase, given a feature vector V, the conditional probability $P_i = Pr(V|C_i)$ is computed for each object category C_i and the selected category for the object is the one that maximizes this probability.

The training dataset contains 100 samples of each of the four object categories. 75% of samples from each category were randomly selected for training and the remaining 25% for testing (i.e., total of 300 training samples, and 100 validation samples). This process was conducted several times, randomly selecting each time the training and testing sets.

This approach yielded on average 63% classification accuracy (i.e., given a object, on average 63% of the time it is classified correctly by the above method). A possible explanation for this low classification rate is that the Gaussian model does not adequately capture the underlying feature distribution.

4. GA-BASED NEAREST NEIGHBOR CLASSIFICATION

We also applied is *nearest neighbor* (NN) classification [3]. In this approach, each dataset sample contains d parameters (19 in our case) and a class label (for training). Note that each sample can be represented as a point in d-dimensional space. Given a new unclassified sample, it will be assigned the category of that training sample whose “distance” to the unclassified sample is the smallest. (That is, the unclassified sample is assigned the label of its nearest neighbor sample in the training set.)

Given two points in space, how should the above distance be computed? Using the straightforward Euclidean distance, without attaching different weights to the various features, might not be representative, as it is obvious that not all of the 19 features are equally important in deciding the classification of the object. Instead, a *weighted distance* [7] for feature vectors X and Y is computed by

$$\mathrm{NN}_w = \sqrt{\sum_{i=1}^{d} w_i (X_i - Y_i)^2},$$

where a weight $w_i (i = 1, ..., d)$ is learned for each of the d parameters. Thus, if the significance of a certain parameter is relatively low, its weight can be as small as 0, not affecting the distance calculation at all.

Manually setting the 19 weights in our case required an infeasible amount of trial and error. Having reduced, however, the problem to that of parameter optimization, the values can be automatically optimized using genetic algorithms.

For our learning purposes we define the chromosome as a list of weights. For example, having 19 weights (for the 19 parameters) and allocating 6 bits per weight (so that the range of weights is 0 to 63), the chromosome consists of 114 bits. Finally, the fitness function is defined as follows. During the learning phase, each organism (a set of weights) classifies (due to its weighted distance calculation) a set of test samples for which we know the expected correct classification. The higher the classification accuracy of the organism, the higher the fitness value will be. At the end of the learning process, we select the best set of weights.

We used a standard implementation of GA with proportional selection and uniform crossover ($OrgNum = 200, P_m = 0.005, P_c = 0.75$). Our experiments showed that just by allowing for a weighted distance calculation, the classification accuracy increases considerably in comparison to standard classification due to Gaussian mixtures, as well as ordinary nearest neighbor classification (for which we obtained roughly 75% accuracy). Incidentally, even basic 0/1 Boolean weights results in an improved classification rate.

The following weights were evolved for the 19 parameters: [20, 57, 1, 19, 4, 54, 43, 32, 0, 2, 7, 20, 25, 28, 19, 60, 13, 41, 4]. Note that some of the weights are set to very low values (0, 1, 2), while others have higher values. Testing this set of weights in a weighted nearest neighbor framework yields on average 90% classification accuracy, i.e., a considerable improvement over standard, non-evolutionary methods.

5. CONCLUDING REMARKS

In this paper we presented a genetic algorithms based approach for automatically learning the feature weights for nearest neighbor classification. The results show a significant improvement over standard non-evolutionary classification methods.

6. REFERENCES

[1] B. Bose and E. Grimson. Improving object classification in far-field video, *CVPR 2004*.

[2] L. Chen, R. Feris, Y. Zhai, L. Brown, and A. Hampapur. An integrated system for moving object classification in surveillance videos, *AVSS 2008*.

[3] T. Cover and P. Hart. Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection, *CVPR 2005*.

[5] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, 2001.

[6] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, 60(2): 91–110, 2004.

[7] R. Paredes and E. Vidal. Learning Weighted Metrics to Minimize Nearest-Neighbor Classification Error, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1100–1110, 2006.

[8] P. Viola, M. Jones and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance, *International Journal of Computer Vision*, 63(2): 153–161, 2005.