Genetic Algorithms for Automatic Object Movement Classification*

Omid David[1], Nathan S. Netanyahu[1,2], and Yoav Rosenberg[3]

[1] Department of Computer Science, Bar-Ilan University,
Ramat-Gan 52900, Israel
mail@omiddavid.com, nathan@cs.biu.ac.il
[2] Center for Automation Research, University of Maryland,
College Park, MD 20742, USA
nathan@cfar.umd.edu
[3] ProTrack Ltd.
P.O. Box 34384
Jerusalem 91343, Israel
yoav@protrack.co.il

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.

Keywords: Genetic algorithms, Parameter tuning, Computer vision, Automatic object recognition, Movement classification

1 Introduction

ProTrack is a company developing and selling products in the video surveillance area. 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 is the ability to distinguish between significant and insignificant motion. For instance, alerting any human motion while filtering out animal motions. Another common requirement is to detect human in an area where cars are usually moving and vice versa. For this reason, automatic object classification of the detected object is needed. The main requirement is to classify between four categories: walking man, crawling human, animal and car. It is also necessary that the classification

* A preliminary version of this paper appeared in *Proceedings of the 2010 Genetic and Evolutionary Computation Conference.*

will be performed on real-time video without adding significant processing load to the CPU. For this reason, the classification should be based on the low level information already generated during the motion detection stage. It is also required that the classification would be independent on the camera installation, i.e., the camera internal and external parameters, the view direction and the distance to the scene. No prior knowledge about the camera or the scene are assumed. Moreover, the classification computation should be completed during the time that the object is exposed to the camera while the camera is scanning. A typical time is 2-3 seconds.

The intrusion detection system usually works in far-field outdoor scene where the moving object's size can be as small as 4 pixels in its biggest dimension. With such small objects, classification performances can be very poor. A practical requirement is to be able to classify objects larger than 20 pixels.

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

Many different approaches have been proposed for object classification. Some methods are based on object detection and classification in still images. For example, in [6], object classification is based on the well known SIFT features. Other methods have been developed for object detection in video sequence where each video image is scanned for object of interest using special developed detectors. One example for such a method is provided by [8], where special pedestrian detector is presented. The detector is based on a shape filter and motion between two consecutive frames. In [4], pedestrians are detected using histogram of oriented gradients and the method is compared to other types of histograms. Object detection in these methods are based on shape and may require large dataset for learning. Moreover, lot of CPU power is required and false positive may be a problem since the detection is performed on each video image.

In other works on object classification in far-field video, a static camera is assumed and the methods are based on scene modeling. Moving objects are extracted and classification is performed based on the detected object motion behavior and shape. These methods are inappropriate to a scanning camera. For example, in [1, 2], far field object classification between man and car in a static camera is presented. Both apply background subtraction for moving objects detection and uses scene-dependant and scene-independent features for the classification. In [2], a feature vector for each moving object is computed. Some features are scene dependant and others are scene-independent. The results of the classification using only scene-independent features is not good enough and hence, scene-dependant features are used. A special technique is presented for automatic adaptation for new scenes. However, such a method is inappropriate for a scanning camera. Moreover, the method is not efficient for non-urban scenes

where the motion patterns are less repetitive. In [1], a discriminative feature is used to measure non-rigidity of the object based on the changes in the histogram of the oriented gradients of a moving object. This scene-independent feature is combined with scene-dependant feature such as velocity, direction and size. The scene is divided to blocks and object behavior at each block is learned and modeled as a Gaussian mixture. This scene-dependant method is not appropriate for scanning camera as well.

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.

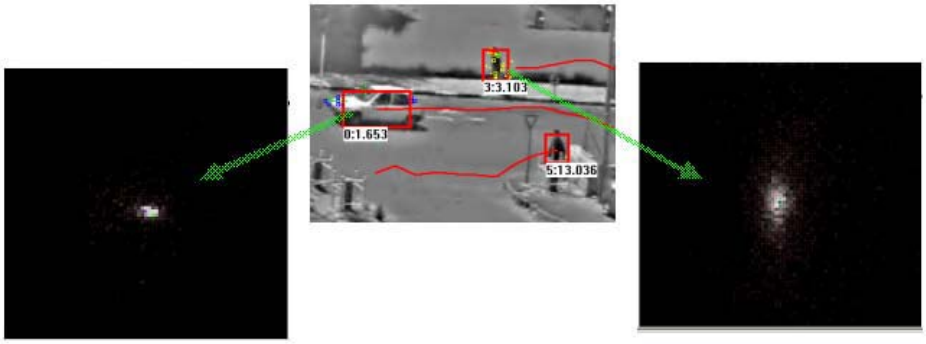

Fig. 1. An example of micro-tracker variance for human and car.

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 1000 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. Using this straightforward Euclidean distance we obtained roughly 75% classification accuracy. To improve the classification accuracy, *weighted distance* [7] is computed for feature vectors X and Y 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.

Parameter	Value
Velo	20
Acc	12
AccRate	13
ShapeRatio	20
Velo2Height	15
ObjectSize	19
AvgVelo	17
Velo Proj	9
VeloUpDist	14
VeloLowDist	22
VeloLeftDist	17
VeloRightDist	15
VeloUp	6
avg VeloLow	12
avg VeloLeft	14
avg VeloRight	9
dist low/avg dist	17
avg low/avg velo	21
abs(left-right)/avg velo	18

Table 1. Evolved weights due to GA.

For our learning purposes we define the chromosome as a list of weights. For example, having 19 weights (for the 19 parameters) and allocating 5 bits per weight (so that the range of weights is 0 to 31), the chromosome consists of 95 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 ($Population = 200, P_m = 0.005, P_c = 0.75$). Our experiments show 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.

Table 1 provides the evolved weights for the 19 parameters. Note that some of the weights are set to low values while others have higher values. Testing this set of weights in a weighted nearest neighbor framework yields on average 88% classification accuracy, i.e., a considerable improvement over standard, non-evolutionary methods.

To further improve the classification rate, we define *confidence rate* as the distance of the second nearest neighbor from a different category than that of the nearest neighbor (D_2) divided by the distance of the nearest neighbor (D_1):

$$confidence = \frac{D_2}{D_1}$$

The higher this confidence ratio, the farther the "second best" category, and thus, the higher the confidence of the classification. For example, if this ratio is close to 1, the second best option is almost as good as the best option, and thus the classification confidence is low. This is illustrated in Figure 2.

Using this confidence ratio, we decide that if the ratio for any sample drops below 1.5, instead of using the classification provided by nearest neighbor classification, we use the classification provided by another rule-based algorithm. Although the performance of this simple rule-based algorithm is considerably inferior to that of GA-based NN, using it only in cases where GA-based NN gives a low confidence ratio improves the overall classification rate from 88% to 90.1%. Table 2 provides the classification rate per category.

Car	Animal	Crawling	Walking
94.6%	79.8%	92.4%	93.6%

Table 2. Classification rate for each category.

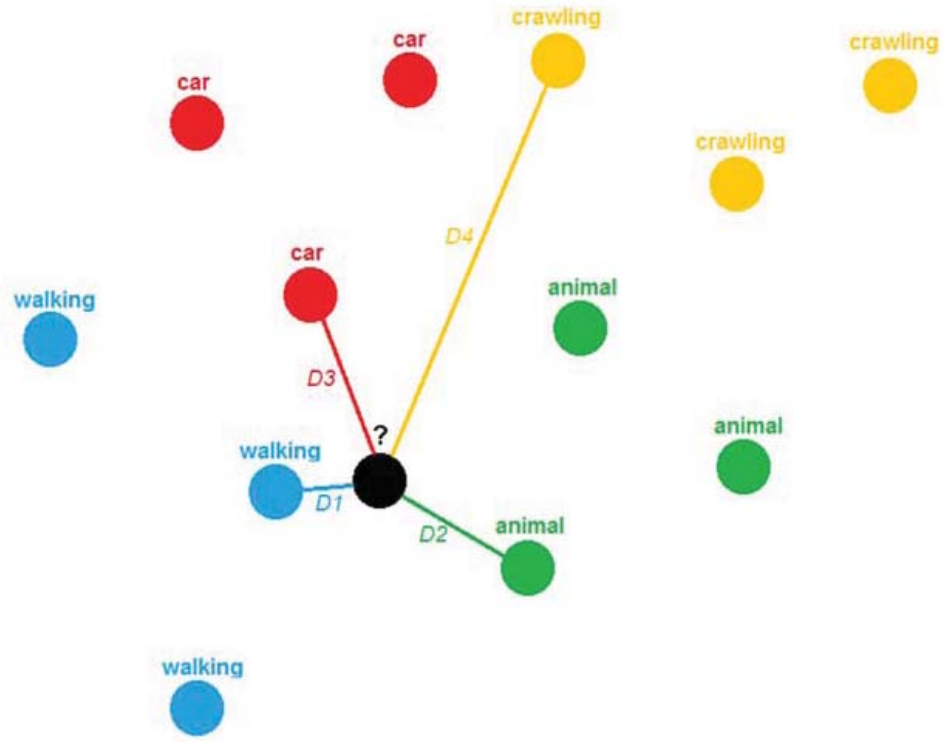

Fig. 2. Illustration of confidence ratio; in the example shown, it is computed by the ratio $\frac{D_2}{D_1}$.

5 Concluding Remarks

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

Acknowledgements

The support of the VULCAN Consortium under the Israeli Ministry of Industry, Trade and Labor is gratefully acknowledged.

References

1. B. Bose and E. Grimson. Improving object classification in far-field video, In *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 181–188, Washington, DC, 2004.
2. L. Chen, R. Feris, Y. Zhai, L. Brown, and A. Hampapur. An integrated system for moving object classification in surveillance videos, In *Proceedings of the 5th IEEE International Conference on Advanced Video and Signal based Surveillance*, pp. 52–59, Santa Fe, NM, 2008
3. T. Cover and P. Hart. Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*, Vol. 13, No. 1, pp. 21–27, 1967.
4. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection, In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 886–893, San Diego, CA, 2005.
5. R. Duda, P. Hart, and D. Stork. *Pattern Classification*, 2nd edition, Wiley, New York, 2001.
6. D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints, *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 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*, Vol. 28, No. 7, pp. 1100–1110, 2006.
8. P. Viola, M. Jones and D. Snow. Detecting Pedestrians Using Patterns of Motion and Appearance, *International Journal of Computer Vision*, Vol. 63, No. 2, pp. 153–161, 2005.