International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:7, 2008

Finding Sparse Features in Face Detection Using Genetic Algorithms

H. Sagha, S. Kasaei, E. Enayati, M. Dehghani

Abstract— Although Face detection is not a recent activity in the field of image processing, it is still an open area for research. The greatest step in this field is the work reported by Viola and its recent analogous is Huang *et al*. Both of them use similar features and also similar training process. The former is just for detecting upright faces, but the latter can detect multi-view faces in still grayscale images using new features called 'sparse feature'. Finding these features is very time consuming and inefficient by proposed methods. Here, we propose a new approach for finding sparse features using a genetic algorithm system. This method requires less computational cost and gets more effective features in learning process for face detection that causes more accuracy.

Keywords— Face Detection, Genetic Algorithms, Sparse Feature**.**

I. Introduction

After proposing an innovative method in up-right face detection by Viola and Jones [1], there was so many effort to modify it to get better results. The modifications was done in detector structure [2,3,4], strong classifier learning [5,6], weak classifier learning [7,8,9], and feature space [10,7,11,12,13]. One of the most prominent one is proposed by Huang et al [14] who achieved drastic results in multi-view face detection. Their work is called *rotation invariant multi view face detection* (RIMVFD) at first we are going to describe it briefly and then we will discuss our modification over it.

RIMVFD system includes a set of innovative methods: *Width-First-Search* (WFS) tree detector structure, the *Vector Boosting* algorithm for learning vector-output strong classifiers, *Piece-wise function* as a domain-partition-based weak learner, the *sparse feature* in *granular space*, and the heuristic search for sparse feature selection.

A *sparse feature* is a newer version of rectangular features in Viola's work. These kinds of features have two common reasons for using. The most common reason is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. Also the second critical motivation is that feature based system operates much faster than a pixel-based system.

H. Sagha is with computer engineering department, Sharif university of technology, Tehran, Iran (corresponding author to provide e-mail: sagha@ce.sharif.ir).

S.Kasaei, is associate professor in faculty of computer engineering in Sharif university of technology (e-mail: skasaei@sharif.edu).

E.Enayati, is with Birjand university of medical science, Birjand, Iran

A sparse feature consists of a finite number of quadrangular feature sets called *granules*. These granules are defined in the *granular space*. This space as shown in Fig.1 is made up of four bitmaps: I_0, I_1, I_2 and I_3.

Denoting the scale variable of a granules as s ($s = \{0, 1, 2, 3\}$), each granular bitmap I_s is the result of smooth filtering by averaging over $2^s \times 2^s$ patches of the original image. Therefore, a granule $I_s(x,y)$ can be specified by the x-offset, y-offset, and the scale s.

In such a granular space, a sparse feature is represented as the linear combination of several granules as

$$\theta = \sum_i \alpha_i I_{s_i}(x_i, y_i), \alpha_i \in \{-1,+1\}, s_i \in \{0,1,2,3\} \quad (1)$$

where α_i is the combining coefficient, restricted to be bipolar (to have low cost of computation). These features have some advantages in comparison with Viola's Haar-like features [1]. The sparse granular features are highly scalable: they can be more versatile while keeping the same computation load, or more economic to compute if keeping similar structural complexity. Also, it needs just one memory access instead of four for each feature and has more flexibility [15]. Fig.2 illustrates two examples of sparse features.

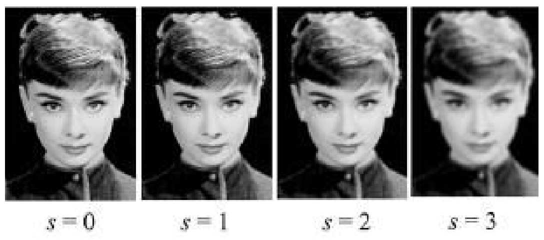


Fig 1: Granular space of a grayscale image
(*s* denotes the scale of granules).

Piece-wise function is an improved version of stump function proposed by Viola. In Viola's method, instead of dividing the feature space into two parts by a threshold, it divides it into several partitions in finer granularity and outputs various values for each bin [16]. Fig 3 shows these two methods.

Denote the samples that are grouped into the jth bin as

$$S_j = \{(x_i, \tilde{v}_i) \mid \theta(x_i) \in bin_j\} \quad (2)$$

Where $\theta(x_i)$ is the extracted feature value of x_i and bin_j is the jth bin after domain partition.

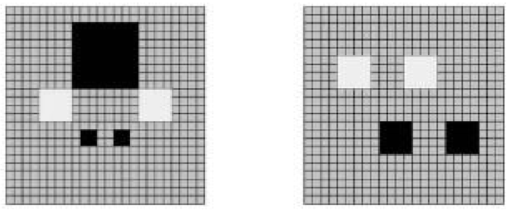


Fig 2: Two examples of sparse features.
[White blocks are positive granules and black ones are negative granules.]

The value of each bin is computed by minimizing the received loss of each bin by

$$loss_j(c_j) \propto \sum_{(x_i, v_i) \in S_j} w_i \exp(-v_i . c_j) \qquad (3)$$

where c_j is the value of the jth bin. Because this function is convex with regard to c_j, optimization procedure can be applied by Newton-step method.

Vector boosting is an augmented version of the AdaBoost classifier which is a learning method that combines some simple and robust classifiers to get a strong classification result. In AdaBoost algorithm, the output is just a scalar but in vector boosting, the output is a vector that is a representation of a part of the output space. The algorithm is similar to the AdaBoost (for more details refer to [17] and [14]). In RIMVFD each vector shows the pose of the face in an image. For example table 1 shows some desired vector of faces of non-faces.

Also, they used a *WFS tree* for parallel computation of the face-nonface detection and the pose estimation. Viola used a simple cascade of face detector learned by AdaBoost.

The best sparse feature is one that has the best fitness function

$$Fitness(\theta) = J(f(x; \theta, \mu^*), F(x)) - \beta \| \theta \| \qquad (4)$$

where μ^* is the optimal piece-wise function learned for θ, $\|\theta\|$ is the number of granules in θ, and β is a small penalty coefficient that can be set as 0.001. Also

$$\begin{aligned} J(f(x), F(x)) = & -\log(Loss(F(x) + f(x))) \\ & + \log(Loss(F(x))) \end{aligned} \qquad (5)$$

$$Loss(F(x)) = \frac{1}{n} \sum_{i=1}^{n} \left\{ \sum_{\tilde{v}_j \in V(x_i)} \exp\left[-\tilde{v}_j . \tilde{F}(x_i) \right] \right\}$$

where $F(x_i)$ is the output of the algorithm for datum x_i, $V(x_i)$ is the set of desired output vectors for pattern x_i, and $\tilde{v}_j$ is the augmented v_j which has an additional bias member. In fact, when $\tilde{v}_j$ and $\tilde{F}(x_i)$ are more apart, the $loss(F(x))$ value grows higher.

TABLE 1) DESIRED VECTOR OF SOME FACE PROFILES

Class		Desired Vector(s)
Faces	Left Profile	(1,0,0,0)
	Frontal	(0,1,0,0)
	Right Profile	(0,0,1,0)
Non-faces		(-1,0,0,0),(0,-1,0,0),(0,0,-1,0)

They used a simple open-list, close-list method as a heuristic to get best sparse features. This results in a high cost at the learning phase of about 2 days for a simple cascade and 2 weeks for a complete WFS tree, as they reported. This cost can be decreased using other methods like evolutionary learning methods. In this paper, we describe a genetic algorithm system adapted to learn sparse features for face detection purposes.

II. PROPOSED METHOD

We have adopted the *genetic algorithm* (GA) instead of simple open-list-close-list approach to obtain sparse features in less time and more accuracy. Genetic algorithm is a part of *evolutionary computing* (EC) method. The general scheme of the EC is shown in Fig.4, and its pseudo-code is shown in Fig.5.

To initiate a GA (or broadly an EC) we must define six items: representation, crossover function, mutation method, fitness function, parent selection, and survivor selection.

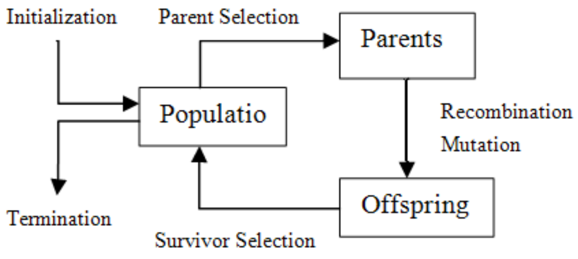


Fig 4: General scheme of EC.

```
Begin
  Initialize the population with random candidate solutions;
  Evaluate each candidate;
  Repeat until (termination condition is satisfied)   DO
    1-Select parents;
    2-Recombine pairs of parents;
    3-Mutate the resulting offspring;
    4-Evaluate new candidates;
    5-Select individuals for the next generation;
  OD
END
```

Fig 5: Pseudo-code of an EC.

Representation is the mapping from phenotype space (possible solutions within the original problem) onto genotype space consists of chromosomes (individuals as solution candidates). It can be bitwise, integer, float, or alike which denotes the parameters of the problem to be optimized. **Crossover function** is a definition of merging two chromosomes into one or two offspring chromosomes with the aim of finding the better candidates which lead to better solutions to the problem. **Mutation** defines some stochastic approaches for altering some genes in a chromosome to search extensively in the search space. **Fitness function** demonstrates the closeness to the best solution. **Parent selection** is a statistical fitness-based mechanism to select the parents that would be combined together. Finally, **survivor selection** stage selects some of the parents and new born children to transfer to the next generation; usually it is a fitness-based approach.

i) Representation: Here, to adapt a genetic system for learning a sparse feature we need a proper representation of pixel locations for representation. Also, according to RIMVFD, two other factors are needed, scale and coefficient. Thus, a chromosome denotes a sparse feature and consists of a finite number of granules {Granule #1, Granule #2,…, Granule #n}, where *n* is a constant or can vary. A granule has the structure [*x, y, s, α*].

Locations (*x, y*) are restricted to be an integer between 1 and 24 (or it can be normalized to be used in a popular GA program). Scale ***s*** is an integer between 0 and 4, which denotes the size of quadrangular. *α* is a real number between [1 to -1] which is a coefficient represented in (1).

ii) Crossover function: replaces randomly a granule with its counterpart in the other chromosome. Thus, a granular feature can be moved into other sparse features, so less time cost will be achieved by this movement of pre-computed granules. We use the steady state model $(\mu+\lambda) = (10+30)$. It means that during each iteration of genetic algorithm, the population is 10 and the number of children is 30. So the survivor selection mechanism must select 10 individuals out of 40 parents and generated individuals.

iii) Mutation function: is a simple mutation method which alters a value in a random part of a chromosome. It can change the value of '*x*', '*y*', or coefficient '*α*' by a Gaussian random variable or changing the scale '*s*' randomly out of 0, 1, 2 and 3.

iv) Fitness function: is the defined fitness function in RIMVFD shown in (4).

v) Parent Selection: is Fitness-Proportional (FP) which means the individuals that has higher fitness, has more chance to be selected for combination.

vi) Survivor Selection: is also Fitness-Proportional, moreover has the ability of elitism that keeps two best individuals in the population and sends them to the next generation without manipulating.

We used the repair mode; this method operates after applying the crossover and the mutation on variables, to be in the defined ranges.

We used AdaBoost instead of vector boost, because we just intend to find the upright face or non-faces. The output of each weak learner passes through the piece-wise function. A threshold is applied after the final superposition of weak classifiers. We set all thresholds equal to zero. The system can obtain better results if these thresholds get other pre-computed values.

III. Experimental Results

To evaluate our proposed method, we used a "weak" database that consists of 2157, 24×24 faces collected from a number of other databases with no preprocessing stage; some of them are rotated a bit instinctually. We used neither scaling nor rotating. We just normalized them to have zero mean and variance equal to one. This database also consists of 3118 non-faces. We defined the absolute number of granules in a chromosome equal to 8. Number of cascade is 7 with 2, 3, 5, 10, 15, 30, 45 weak classifiers, respectively. Training time was about half a day. The results over MIT database is 94% cooperated with 130 false detections; while in the same correction rate Viola and Huang got 167 and 150 false detections, respectively. It is a good result if you consider the weakness of database and also the number of cascades and weak learners. Table 2 shows the components of two databases used in Huang *et al.*'s method, Viola's method and our method. Fig 6 shows some of our training images and as it can be seen, some faces are rotated and some have not cropped well.

The run time does not differ from the original version because we just find sparse features through a faster algorithm and the structure has not changed. Here, α is not restricted to be bipolar, but the low number of features compensates the time of real number computations. Fig 7 shows the detected faces obtained by our proposed method.

Table 2: Components and statistics of different systems.

	Training Image		Training Time	Feat-ures	Accuracy	
	Faces	Non-Faces			C	FD
Our Method	2157	3118	12 hours	880	94%	130
Huang *et al.*	30000		2 days		94%	150
Viola	4916	10000		6061	94%	167

IV. Conclusion

Evolutionary computing is a well-known approach for optimization. Here, we used this approach for the face detection problem. This approach was adapted to the problem and resulted in improving the face detection process by getting better feature sets as sparse features. Also, it must be noted that the used database contained varying faces according to position and rotation and also it had few number of faces. Moreover, any preprocessing stage is not applied to modify the faces. The used threshold can also be set with some computations instead of setting it to zero. Finally, this approach can be applied for other applications such as multi-view face detection, image understanding, image segmentation, and image retrieval.

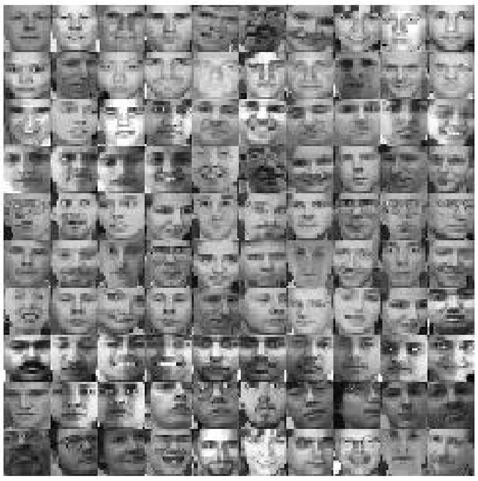
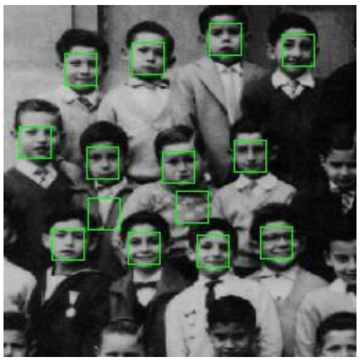
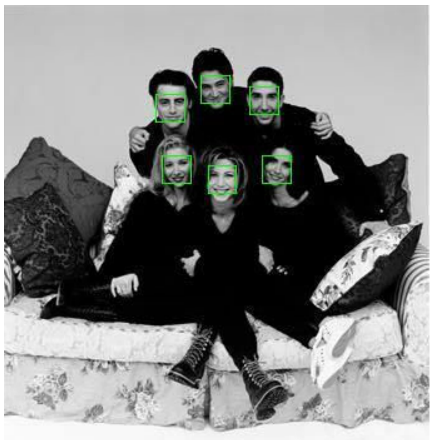


Fig 6: Some input images used in our training phase.





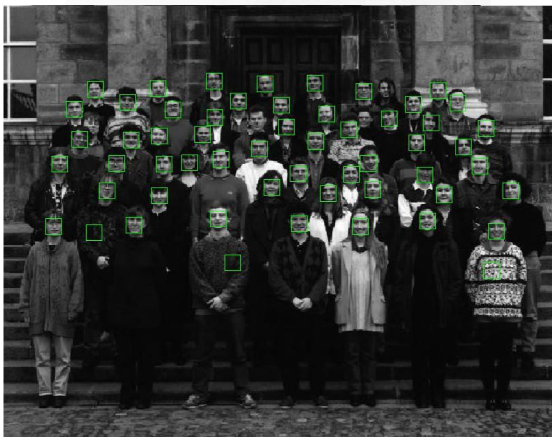


Fig 7: Obtained face detection results on some images in MIT database.

REFERENCES

[1] P.Viola, M.Jones. "*Rapid Object Detection Using a Boosted Cascade of Simple Features"*. Computer Vision and Pattern Recognition, 2001.

[2] S.Z. Li et al., "*Statistical Learning of Multi-View Face Detection*", Proc. Seventh European Conf. Computer Vision, pp. 67-81, 2002.

[3] M. Jones and P. Viola, "*Fast Multi-View Face Detection*", MERLTR2003-96, July 2003.

[4] C. Huang, H.Z. Ai, Y. Li, and S.H. Lao, "*Vector Boosting for Rotation Invariant Multi-View Face Detection*", Proc. 10th IEEE Int'l Conf. Computer Vision, 2005.

[5] R.E. Schapire and Y. Singer, "*Improved Boosting Algorithms Using Confidence-Rated Predictions*", Machine Learning, vol. 37, pp. 297-336, 1999.

[6] J. Friedman, T. Hastie, and R. Tibshirani, "*Additive Logistic Regression: A Statistical View of Boosting*" Annals of Statistics, vol. 28, pp. 337-374, 2000.

[7] C. Liu and H.Y. Shum, "*Kullback-Leibler Boosting*", Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 587-594, 2003.

[8] T. Mita, T. Kaneko, and O. Hori, "*Joint Haar-Like Features for Face Detection*", Proc. 10th IEEE Int'l Conf. Computer Vision, 2005.

[9] B. Wu, H. Ai, C. Huang, and S. Lao, "*Fast Rotation Invariant Multi-View Face Detection Based on Real AdaBoost*", Proc. Sixth Int'l Conf. Automatic Face and Gesture Recognition, pp. 79-84, 2004.

[10] R. Lienhart and J. Maydt, "*An Extended Set of Haar-Like Features for Rapid Object Detection*", Proc. IEEE Int'l Conf. Image Processing, 2002.

[11] S. Baluja, M. Sahami, and H.A. Rowley, "*Efficient Face Orientation Discrimination*", Proc. IEEE Int'l Conf. Image Processing, 2004.

[12] P. Wang and Q. Ji, "*Learning Discriminant Features for Multi-View Face and Eye Detection*", Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2005.

[13] Y. Abramson and B. Steux, "*YEF* Real-Time Object Detection*", Proc. Int'l Workshop Automatic Learning and Real-Time, 2005.

[14] C.Huang, H.Ai Y.Li, S.Lao. "*High Performance Rotation Invariant Multiview Face Detection". IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 4, APRIL 2007.

[15] C.Huang, H.Ai, Y.Li, S.Lao. "*Learning Sparse Features in Granular Space for Multi-View Face Detection". 7th International Conference on Automatic Face and Gesture Recognition (FGR)*, 2006.

[16] B.Wu, H.AI,C. Huang, S.Lao. "*Fast Rotation Invariant Multi-View Face Detection Based on Real Adaboosts*". 6th *IEEE International Conference on Automatic Face and Gesture Recognition (FGR)*, 2004

[17] C.Huang, H.Ai, Y.Li, S.Lao. "*Vector Boosting for Rotation Invariant Multi-View Face Detection".*10th *IEEE international conference on computer vision*, ICCV, 2005.