Swarm Search for Fast Face Detection with Neural Networks

Xinjian Fan, Masanori Sugisaka
Department of Electrical and Electronic Engineering, Oita University, Japan
Email: {msugi, fxinjian}@cc.oita-u.ac.jp

Abstract: This paper presents the application of particle swarm optimization (PSO) to improve the search speed of neural network based face detection problems. The method is based on the idea that the task of finding a well-matched subwindow (face) can be formulated as an integer nonlinear optimization problem (INLP). To find a face, we only need to find a local maximum filter response which value is above a given threshold. The proposed method has been tested and examined with a set of 42 images to demonstrate its effectiveness. The results confirm the potential of the proposed approach and show its superiority over the classical technique.

1. Introduction

Face detection is a very important task that serves as the first step of a large area of applications: automated face recognition, secure access control, advanced human-computer interfaces, etc. Its accuracy and efficiency have a direct impact on the usability of the whole system.

Face detection has been a well researched problem and there are many approaches on it [1]. Up to now one kind of the most successful methods is known as neural network-based methods [2-7]. In these methods, a face model (filter) is first learned from a large number of examples (face images and non-face images). And then a sliding window is used to scan all possible subwindows across multiple-scale images.

However, these methods are generally computationally expensive because: (a) the search window is a high-dimensional vector that has to be classified in a very non-linear space; (b) there are hundreds of thousands of windows to search. Although many efforts have been done to reduce the runtime of neural network based methods, most of them focused on reducing the computational complexity of classifiers [4],[5]. Only a few attentions were given to improving the search efficiency. In Ref. [6], the search window moves every q pixels (q=3~5) instead of every pixel. Thus the number of searched windows is only about $1/q^2$ of the exhaustive search, but with the disadvantage of lowering the system's performance. Many methods use skin color information to limit the search area [6],[7]. But color information is not always able to be used and it is very difficult to build a skin color model robust to illumination changes.

In this paper, to reduce computational cost while retaining high detection accuracy, we propose a new method to speed up neural network (NN) based face detection systems. The method is based on the idea that the face search (FS) problem can be formulated into an integer nonlinear optimization problem (INLP). To find a face, we only need to find a local maximum filter response which is above a threshold. The integer variables are parameters that represent a subwindow in the image. The objective function is based on the output of the face filter.

PSO is a novel evolutionary computation (EC) technique [8], which has been improved and applied to various problems. Although the original algorithm was basically developed for continuous optimization problem, it can be expanded to handle discrete variables easily. Furthermore, PSO has only a few parameters that make it easy-adjusted to get better performance. Therefore, PSO is expected to be suitable for FS formulated as an INLP.

Based on a NN-based face filter, this paper presents a PSO for the FS problem formulated as an INLP. The feasibility of the proposed method is demonstrated and compared with the exhaustive search method on a set of 42 test images with promising results. In this paper, we assume that there is only one face contained in the test image. The extension of the method to detect multiple faces will be done in our future work.

2. Neural network based face filter

The purpose of the face filter is to classify a window of size 20×20 pixels extracted from an image, as a face or as a non-face.

We use a retinally connected neural network [3] to serve as the face filter. The network takes a 20×20 pixel window as input. Each hidden unit receives inputs only from part of the input layer (called a receptive field). There are 3 kinds of receptive fields: four 10×10 pixel regions, sixteen 5×5 pixel regions, and six 20×5 pixel overlapping horizontal stripes. Each of these receptive fields has full connection to two hidden neurons. It has a single output. The output is a real value from -1.0 to 1.0, giving the likelihood as to what extent the input window looks like a face.

3. Formulation of FS as an INLP

Figure 1 shows a 3D plot of the neural network output with the image on the left as input. The highest peak represents the face location. It can be seen that the face filter is very selective: it responds strongly within a several pixel radius of the face while its output on the background is low. Moreover, around the face, the output of the neural filter is a monotonous and growing function. The properties lead to the following heuristic:

The face search (FS) problem can be formulated as an integer nonlinear optimization problem (INLP). To find a face, we only need to find a local maximum filter response which value is above a threshold. Let **T** represent an input image, **SW** represent a subwindow and *dv* be its detection value (the corresponding output of the neural network). With these notations the FS problem can be stated as:

$$\arg\max_{\mathbf{SW}} dv(\mathbf{SW}) \qquad \forall \mathbf{SW} \in \mathbf{T} \tag{1}$$

If $dv(\mathbf{SW})$ is higher than a given threshold, the corresponding portion of **SW** is declared as a face.

4. Particle swarm optimization

Particle swarm optimization (PSO) is a novel evolutionary computation method, modeled after the social behavior of flocks of birds [8]. PSO is a population based search process where individuals, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem at hand. The performance of particles is measured using a predefined fitness function which encapsulates the characteristics of the optimization problem.

Each particle i maintains the following information: $\mathbf{X}_i$, the *current position* of the particle; $\mathbf{V}_i$, the *current velocity* of the particle; $pbest_i$, the personal best position discovered by the particle so far, and also the best position found by the entire swarm so far, denoted by *gbest*. All particles start with randomly initialized velocities and positions. At iteration step t, the current velocity and position (searching point in the solution space) are updated by:

$$\mathbf{V}_i(t+1) = \omega\mathbf{V}_i(t) + c_1 r_1(t)(pbest_i - \mathbf{X}_i(t)) + c_2 r_2(t)(gbest - \mathbf{X}_i(t)) \tag{2}$$

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \mathbf{V}_i(t+1) \tag{3}$$

where w is the inertia weight, c_1 and c_2 are the acceleration constants, $r_1(t)$ and $r_2(t)$ ~ U(0, 1). The velocity of a particle will be set to a predetermined maximum velocity ($\mathbf{V}_{max}$) if it exceeds $\mathbf{V}_{max}$.

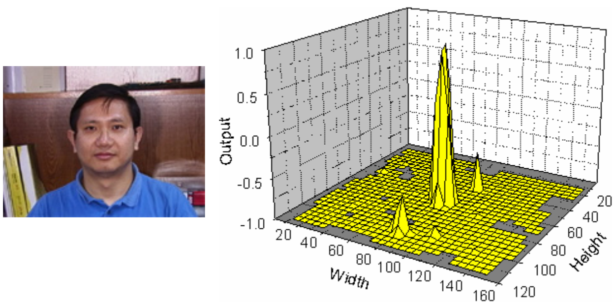

Fig. 1 *left*: an input image; *right*: 3D view of the neural network output, obtained by superposing the outputs of subwindows at all scales.

The features of the algorithm can be summarized as follows:

(a) PSO searches the solution space using a group of searching points like genetic algorithm (GA) and the searching points gradually get close to the optimal point using their *pbests* and the *gbest*.

(b) As explained in Ref. [9], the first term of the right side of Equ. (2) is corresponding to the exploration of the search space. The second and third terms of that are corresponding to the exploitation of the best solutions found so far. Namely, the method has a flexible and well-balanced mechanism to utilize exploration and exploitation in the search procedure.

(c) The original PSO was originally developed for nonlinear optimization problems with continuous variables. However, the method can be expanded to discrete problems easily [10].

(d) Because the update process of PSO is based on simple equations, the algorithm is easy to implement and computing economically. In addition, only a few input parameters need to be adjusted in PSO which makes it easy-adjusted to get better performance.

Due to the above features, PSO is expected to be suitable for the FS problem formulated as an INLP.

5. Face search using PSO

The main steps of the proposed method are shown in Figure 2. In the following, we will describe the approach in detail.

5.1 Encoding and rescaling

In our problem, each particle represents a subwindow in the input image. We use its center (C_x, C_y) and length S to encode a subwindow. To evaluate subwindows of different sizes using the neural network, we should rescale them to the size of 20×20 (the input size of the neural network). However, if this computation is done on every size of subwindows, it will be very time-consuming. To avoid it, we first build an image pyramid†:

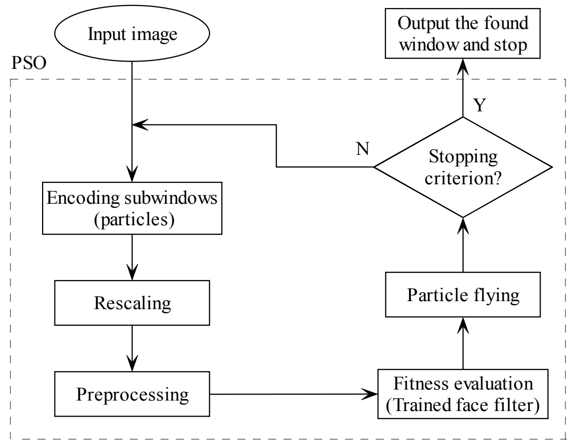

Fig. 2 Main steps of the proposed method

$$W \times H,\ \frac{W}{q} \times \frac{H}{q},\ \cdots,\ \frac{W}{q^k} \times \frac{H}{q^k},\ \cdots,\ \frac{W}{q^L} \times \frac{H}{q^L} \quad (4)$$

where W and H are the width and height of the input image respectively, and q is the scale factor. The top level (level L) should have a size more than 20×20:

$$\frac{\min(W,\ H)}{q^L} \geq 20 \text{ , gives}$$

$$L = \left\lfloor \frac{\ln(\min(W,\ H)) - \ln 20}{\ln q} \right\rfloor \quad (5)$$

Then we let S to be chosen among the following geometric sequence†:

$$20,\ 20q,\ \cdots,\ 20q^k\ \cdots,\ 20q^L \quad (6)$$

For a subwindow $\mathbf{SW} = (C_x, C_y, \lfloor 20q^k \rfloor)^T$, we find its mapped 20×20 window $\mathbf{SW}' = (C_x^{'}, C_y^{'}, 20)^T$ in level k of the pyramid by:

$$C_x^{'} = \left\lfloor \frac{C_x}{q^k} \right\rfloor,\ C_y^{'} = \left\lfloor \frac{C_y}{q^k} \right\rfloor \quad (7)$$

So each particle $\mathbf{X}$ is constructed as $\mathbf{X} = (C_x, C_y, k)^T$. C_x, C_y and k are defined in [10, W-10], [10, H-10] and [0, L] respectively.

5.2 Preprocessing

Before a 20×20 window is passed to the trained neural network, it is preprocessed with lighting correction (by subtracting a best fit linear function) and histogram equalization as in Ref. [2],[3]. The former reduces the effect of different lighting conditions and the latter improves contrast across the window.

5.3 Fitness evaluation

To evaluate each particle (subwindow), we directly use its detection value (the corresponding output of the neural filter): the larger its detection value (dv), the more the subwindow resembles a face. The fitness function $f(\mathbf{SW})$ is given as

$$f(\mathbf{SW}) = dv(\mathbf{SW}) \quad \mathbf{SW} \in \mathbf{T} \quad (8)$$

where $\mathbf{T}$ is the input image and $\mathbf{SW}$ is a subwindow, $dv(\mathbf{SW}) \in [-1,\ 1]$.

The corresponding subwindow of a particle may go beyond the image's boundary even if all its variables lie in the search boundary. To guarantee feasibility of solutions, a random repair method (RRM) is investigated in this paper. If a particle is checked to be infeasible, it will be forced to "fly" to a new position, which is randomly generated but feasible. The method works as follows:

If $\mathbf{SW} \notin \mathbf{T}$, then

Step 1: Randomly generate a new position $\mathbf{SW}'$.

Step 2: If $\mathbf{SW}' \in \mathbf{T}$, replace $\mathbf{SW}$ with $\mathbf{SW}'$; otherwise, go to step 1.

The proposed RRM has proven more efficient for our problem than the traditional penalty approach [11].

5.4 Particle flying

Based on their fitness, particles in the swarm are guided by Equ. (2) and (3) to fly to possible face regions in the image. New (C_x, C_y, k) generated by Equ. (2) and (3) are real values. When corresponding to a subwindow in the input image, they are transformed into integers by using the floor function. During flying, if a variable extends the defined search boundary, it will be set to the closest limit, i.e.

$$x_j = \begin{cases} x_{j\min} & \text{if } x_j < x_{j\min} \\ x_{j\max} & \text{if } x_j > x_{j\max} \end{cases} \quad (9)$$

where x_{jmin} and x_{jmax} are respectively the lower and upper search limit of variable x_j, $x_j \in \mathbf{X}$.

5.5 Stop criterion

The algorithm is stopped when 1) a "face" is found – the detection value of the best particle is above the given threshold or 2) the maximum iteration number is reached.

6. Experiments

A number of experiments were performed to evaluate the proposed method. The experiments were performed on 42 images with complex backgrounds. Some of the images were chosen from CMU Test Set [12] and other Internet resources; the others were taken by us in an indoor environment using a CCD camera. Each image contains only one face and all the faces can be detected by the neural filter. All the images have the same size of 320×240 and the face size ranges from 34×34 to 178×178. The threshold of the neural network output was set to 0.1.

According to pre-simulation, the parameters of PSO were set as:

c_1, c_2: 0.2,

w: 1.2,

$\mathbf{V}_{\max}$: 0.2×($\mathbf{X}_{\max}$-$\mathbf{X}_{\min}$),

Swarm size P: 60,

Maximum iteration number *MaxIt* is set to 70. But one restart is allowed, i.e., if the algorithm fails to find a face within *MaxIt* it will be re-initialized and perform a new search.

For each image in the test set, we ran our algorithm 100 times. The total detection results are listed in Table 1. Some examples are shown in Figure 3. The time consuming was reported on an AMD Athlon 750 MHz PC with Windows 2000 as its OS.

As shown in Table 1, the proposed search method yielded a high success rate (93.6%) on average (the best is 100% and the worst is 72%). Moreover, about 39% of the failures are because PSO fell into a false detection, the other failures are due to non-convergence. A further reduction of false detections can be achieved by arbitrating among multiple networks [3]. From the

† Each term in Equ. (4) and (6) is transformed from a real value to an integer value by using the floor function.

Table 1: Experimental results

Success	False	Non-convergence	ANSEs	APT (ms)
93.6%	2.52%	3.88%	1965	250

False: false detection rate; ANSEs: Average Number of Subwindow Evaluations; APT: Average Processing Time.

Fig. 3 Examples from the test set

examples shown in Figure 3, we can see that the proposed method maintains robustness in images which contain faces under a very wide range of conditions including scale, pose, position, complex backgrounds, illumination conditions, etc.

Table 2 gives the comparison of the proposed search method (called *swarm search*) with the exhaustive search method. It's clear that the time consuming and the number of subwindow evaluations of the proposed method are much less than those of the exhaustive search. Although with a little loss of detection rate (due to non-convergence), a great speedup has been achieved by using the swarm search compared to using the exhaustive search.

The method proposed by Viola and Jones [13] is about 2.7 times faster than ours even performing an exhaustive search. The reason is that they use a computationally extremely efficient face filter, which is made of a boosted cascade of classifiers built with the AdaBoost algorithm. However, it is possible to combine our swarm search method with their face filter to make a more powerful face detection system.

7. Conclusion

This paper presents a new search method for NN-based face detection. The proposed method formulates the problem of face search into an integer nonlinear optimization problem (INLP) and expands the basic PSO to solve it. The feasibility of the proposed method is demonstrated on a set of 42 images with promising results. With fine-adjusted parameters, PSO only requires less than 2000 evaluations of subwindows for finding the face in an image. The result is much more effective and superior over the classical exhaustive search method. Many object detection problems can be formulated as an INLP and the results indicate the possibility of PSO as a practical tool for various INLPs of object detection.

However, we have found that the method doesn't work well on some images, especially when the face size is very small. How to improve the robustness is the future work.

Table 2: Swarm search vs. exhaustive search

	Swarm search	Exhaustive search*	Ratio
ANSEs	1965	193737/2	1 : 49
APT (ms)	250	20169/2	1 : 40

* Because we only consider the single-face detection problem in this paper, for fair comparison, we suppose that it takes only half of an exhaustive search to find a face.

References

[1] M.H. Yang, D. Kriegman and N. Ahuja, "Detecting faces in images: a survey", IEEE Trans. Pattern Anal. & Mach. Intell., vol.24, no.1, pp.34-58, 2002.

[2] K.K. Sung and T. Poggio, "Example-based learning for view-based human face detection", IEEE Trans. Pattern Anal. & Mach. Intell., vol.20, pp.39-50, 1998.

[3] H.A. Rowley, "Neural Network-Based Face Detection", Thesis submitted for the degree of Doctor of Philosophy, School of Computer Science, Carnegie Mellon University, 1999.

[4] L. Huang, A. Shimizu, Y. Hagihara and H. Kobatake, "Face detection from cluttered images using a polynomial neural network", Proc. Conf. on Image Processing, pp.669-672, Thessaloniki, Greece, 2001.

[5] B. Fasel, "Fast multi-scale face detection", Technical Report COM-98-04, IDIAP, 1998.

[6] R. Feraud, O. J. Bernier, J. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks", IEEE Trans. Pattern Anal. & Mach. Intell., vol.23, no.1, pp.42-53, Jan. 2001.

[7] S. Karungaru, M. Fukumi and N. Akamatsu, "Human face detection in visual scenes using neural networks", IEEJ Trans., vol.122-C, no.6, pp.995-1000, 2002.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization", Proc. IEEE Int'l Conf. on Neural Networks, vol.4, pp.1942-1948, Perth, Australia, 1995.

[9] Y. Shi and R. Eberhart, "A modified particle swarm optimizer", Proc. IEEE Int'l Conf. Evolutionary Computation, pp.69-73, Anchorage, 1998.

[10] H. Yoshida, Y. Fukuyama, K. Kawata, S. Takayama, and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment", IEEE Trans. Power Systems, vol.15, no.4, pp.1232-1239, 2001.

[11] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods", Proc. 4th Annual Conf. on Evolutionary Programming, MIT Press, Cambridge, MA, pp.135-155, 1995.

[12] CMU Test Set: http://vasc.ri.cmu.edu/idb/images/face/frontal_images/images.tar

[13] P. Viola and M. Jones, "Robust real-time object detection", Proc. 2nd Int'l Workshop on Statistical and Computational Theories of Vision, Vancouver, Canada, 2001.