Optimization of Spatial Filters for Image Texture Feature Separation by Simulated Annealing

Frode Nergård and Trygve Randen

Høgskolen i Stavanger
P.O. Box 2557 Ullandhaug, 4004 Stavanger, Norway
http://www.hsr.no/tele-signal-group/papers/nerga96.ps.gz
E-mail: t.randen@hsr.no

ABSTRACT

In a work by Randen and Husøy, an approach to the design of filters for texture feature extraction for two textures has been developed. The feature extraction process were modelled as a filter with a subsequent local energy function. The filter was optimized with respect to a criterion of obtaining high distance between the modelled feature means and low modelled feature variances. However, the optimization was based on some simplifying assumptions. Using the same criterion, without making these simplifications, we propose three methods for finding the optimal filter. All methods have their basis in simulated annealing, and one of them, called annealing evolution, are also based on genetic algorithms. Experiments indicate that the suggested methods were able to find better filters than the methods proposed by Randen and Husøy.

1. INTRODUCTION

The modelled experimental setup is visualized in Fig. 1. Consider an input image consisting of only

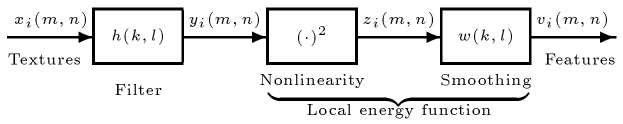

Figure 1. The experimental setup.

two textures, $x_1(m,n)$ and $x_2(m,n)$. The system output for texture $x_i(m,n)$ is the feature image $v_i(m,n)$. We are attempting to separate the outputs for the two textures in such a way that we can classify the two textures by a simple linear two-class discriminant classifier with minimum classification error probability. Randen and Husøy [1] proposed to maximize the filter with respect to the Fisher criterion [2]

$$J(\mathbf{h}) = \frac{(\mu_{v1} - \mu_{v2})^2}{\sigma_{v1}^2 + \sigma_{v2}^2}, \tag{1}$$

for the extracted features. In this paper we develop three new approaches to maximizing $J(\mathbf{h})$.

We choose the size of the filter $h(k,l)$ to $M \times N$. The filtered image $y_i(m,n)$ is given by $y_i(m,n) = h(k,l) * x_i(m,n)$, where $*$ denotes two-dimensional convolution. We define $\mathbf{h}$ as the vector formed by lexicographically ordering the rows of $h(k,l)$, and $\mathbf{x}_i(m,n) = [x_i(m,n) \cdots x_i(m,n-N+1) \cdots x_i(m-M+1, n-N+1)]^T$. The output for picture element (m,n) may now be written

$$y_i(m,n) = \mathbf{h}^T \mathbf{x}_i(m,n).$$

Similarly, the smoothing operation may be defined as

$$v_i(m,n) = \mathbf{w}^T \mathbf{z}_i(m,n).$$

We assume that the smoothing filter is an $N_w \times N_w$ rectangular filter.

Eq. (1) involves the feature means and variances for the two textures. By assuming wide sense stationarity Randen and Husøy have shown that the feature mean may be expressed by

$$\mu_{v_i} = \mathbf{h}^T \mathbf{R}_{xx_i}(0,0)\mathbf{h}$$

and the variance by

$$\sigma_{v_i}^2 = \frac{2}{N_w^2} \sum_{k=0}^{N_w-1} \sum_{l=0}^{N_w-1} (\mathbf{h}^T \mathbf{R}_{xx_i}(k,l)\mathbf{h})^2,$$

where

$$\mathbf{R}_{xx_i}(k,l) = E\{\mathbf{x}_i(m,n)\mathbf{x}_i^T(m+k, n+l)\}.$$

Randen and Husøy [1] previously designed texture filters optimized with respect to the Fisher criterion by both closed form and gradient search optimization. The closed form optimization is based on significantly simplifying assumptions, and the gradient search is likely to converge to a local optimum. Here, we use simulated annealing in order to find the maximum of the object function defined by the Fisher criterion, Eq. (1).

One special property with our object function is that multiplying $\mathbf{h}$ with a scalar factor does not affect the object function value, i.e.

$$J(\mathbf{h}) = J(s\mathbf{h}) \tag{2}$$

for any scalar s. Hence, for each $\mathbf{h}$ there exists a multidimensional line of filters in the filter space having exactly the same object function value. Therefore, an infinite number of optimal solutions and an infinite number of local maxima exist. If this property is not considered in the implementation of the simulated annealing algorithm, it may result in slow convergence.

2. SIMULATED ANNEALING AS FILTER OPTIMIZER

A simulated annealing algorithm is described in Fig. 2. This algorithm will maximize the object function $J(\mathbf{h})$. The description is a reformulated version of the algorithm proposed by Corana et al. [3] and adapted to the texture feature separation problem of Eq. (1).

A generated point (corresponding to a filter) is accepted if it is better than its adjacent point. Otherwise the point is accepted with some probability.

New points, $\hat{\mathbf{h}}$, are generated according to the rule $\hat{\mathbf{h}} = vd\mathbf{h}$ where v is a step length factor and d is a random number between 0 and 1. The step length factor is regularly adjusted on the basis of the relative number of new points that are accepted in an interval of time. More specifically

$$v' = \begin{cases} \frac{v}{1+c\frac{0.4-\frac{a}{N_s}}{0.4}} & \text{if } a < 0.4N_s \\ v(1 + c\frac{\frac{a}{N_s}-0.6}{0.4}) & \text{if } a > 0.6N_s \\ v & \text{otherwise,} \end{cases} \tag{3}$$

where N_s is the number of generated points and a is the number of them which were accepted. The motivation for this scheme is to try maintaining a 1:1 ratio between the numbers of accepted and rejected moves to obtain an efficient search [3].

Furthermore, we normalize the variables due to the property described in Eq. (2). This normalization can be regarded as a kind of bounding of the variables.

```
begin
    select initial temperature T > 0;
    select a temperature reduction factor r;
    select an initial filter h;
    select N_T, N_s and n;
    while termination criterion = FALSE do
        for m = 1 to N_T do
            for j = 1 to N_s do
                for i = 1 to n do
                    generate x from h;
                    δ = J(h) − J(x);
                    if δ ≤ 0
                        h = x;
                        record acception;
                        if J(h) > J(h_opt)
                            h_opt = h;
                        end
                    elseif e^(−δ/T) > r ∈ C[0, 1];
                        h = x;
                        record acceptation;
                    end
                end
                Normalize h;
            end
            Evaluate the step length;
        end
        T = rT;
        h = h_opt;
    end
end
```

Figure 2. Simulated annealing algorithm

After a number of iterations which each involves generation of one accepted or rejected point, the temperature is geometrically lowered and new iterations are made.

The simulated annealing has succeeded when it has arrived into a region with only one maximum - an anti-unimodal region. To get an indication whether the search has converged into such a region, gradient search [1] is applied to several subsequent stages, and is the gradient searches all converge to the same solution, it is assumed that an anti-unimodal region is found, and the random search is terminated.

We have implemented this algorithm in two versions that we denote here as *algorithm 1* and *algorithm 2*. Algorithm 1 selects n (see Fig. 2) equal to the dimension of $\mathbf{h}$ and generates new points in a cyclic fashion, which means that the moves are parallel with one function axis at a time. The number of accepted moves is recorded individually for moves in each coordinate direction which all have individual step length factors. Then we obtain a sort of tracking control which makes the search take reasonable step lengths which are adapted to the behavior of the object function.

Algorithm 2 lets $n = 1$ so that the innermost **for** statement can be removed. The moves are then generated in random directions, and only one step length factor is kept for the search.

3. ANNEALING EVOLUTION AS FILTER OPTIMIZER

The intuitive drawback with the simulated annealing algorithms described in the previous section is that the algorithms have no mechanisms of preventing the searches against examining regions of the function domains already examined. Hence, the searches become less efficient than they could be if the function domains were explored in more systematic manners.

An optimization technique which uses simulated annealing, but incorporates such a controlling mechanism is called *annealing evolution*. The algorithm is described in Fig. 3, and this is a reformulated version of the one described by Yong et al. [4]. The approach is in fact a combination of a category of algorithms called *genetic algorithms* [5], and simulated annealing. Genetic algorithms are based on the idea of "survival of the fittest" taken from the evolution theory in biology.

The search space must be bounded, something which do not create problems to the optimization problem here because of the property of Eq. (2). Initially, a population P consisting of N_P points is selected with a pseudo-random number generator with uniform distribution over the search space. Simulated annealing is employed in L steps from each individual point in the population. Thus, the population is increased to at most $N_P(L+1)$ points. Of these, we select the N_P points with highest function values. This population is now the *survival set*. Then, the temperature is geometrically lowered, and new simulated annealing searches from each of the points in the population are performed. This is repeated until the termination criterion is satisfied.

When the temperature is high, the new points are generated (more or less) uniformly over the function domain, but when the temperature is lowered the search narrows. In effect, the most promising parts of the function domain are selected and explored in more detail. Eventually, only one

begin
    Select initial temperature $T > 0$;
    Select a temperature reduction factor r;
    Initialize a population P on size N_P;
    $n_T = 0$;
    while termination criterion = FALSE **do**
        for $j = 1$ to N_P
            Represent P by $\mathbf{h}_{j1}$, $j = 1, \dots, N_P$
            $i = 1$;
            for $k = 1$ to L **do**
                generate $\mathbf{y}$ from $\mathbf{h}_{ij}$;
                $\delta = f(\mathbf{h}_{j1}) - f(\mathbf{y})$
                if $\delta \leq 0$
                    $i = i + 1$;
                    $\mathbf{h}_{ji} = \mathbf{y}$;
                elseif $e^{-\frac{\delta}{T}} > r \in C[0,1]$;
                    $i = i + 1$;
                    $\mathbf{h}_{ji} = \mathbf{y}$;
                end
            end
        end
        $T = rT$;
        $n_T = n_T + 1$;
        Order the h_{ji}'s after their function values in ascending order into P;
        Select the first N_P points of P as the new survival set P;
        Test the termination criterion;
    end
    The first point in P is the best filter found;
end

Figure 3. Annealing evolution algorithm

or a few regions are examined.

Because of the property expressed in Eq. (2), the vector of the variables is *normalized* as the points are accepted. Since the algorithm generates points uniformly over the function domain, the function domain must be bounded.

The vector of the variables is normalized by dividing by,

$$d = \sqrt{\sum_{i=1}^{n} \hat{h}_i^2}, \tag{4}$$

where $\hat{h}_i$, $i = 1, \dots, n$ are the n variables.

In the algorithm we must adjust the randomly generated step accordingly. For this purpose, it is intuitively reasonable to use the expected value of

d. Then,

$$E\{d\} = E\left\{\sqrt{\sum_{i=1}^{n} \hat{h}_i^2}\right\} = \sqrt{\sum_{i=1}^{n} E\left\{\hat{h}_i^2\right\}}$$

For a uniformly distributed stochastic variable $\hat{h}_i$ with lower and upper boundaries L_i and R_i respectively, we find the second moment expectation

$$E\left\{\hat{h}_i^2\right\} = \frac{1}{R_i - L_i}\int_{L_i}^{R_i} \hat{h}_i^2\,dx = \frac{1}{3}\frac{R_i^3 - L_i^3}{R_i - L_i}. \quad (5)$$

Allowing of individual boundaries for each x_i, Eq. (3) and Eq. (5) yields

$$D = E\{d\} = \frac{1}{\sqrt{3}}\sqrt{\sum_{i=1}^{n}\frac{R_i^3 - L_i^3}{R_i - L_i}}, \quad (6)$$

which is used in generation of new points.

As mentioned, the points generated should tend from initially having a uniform distribution over the function domain, to have more and more non-uniform distribution. To obtain this, we propose the following rules for random generation of a point $\tilde{\mathbf{h}} = [\tilde{h}_1\ \tilde{h}_2 \ldots \tilde{h}_n]$ from a point $\hat{\mathbf{h}} = [\hat{h}_1\ \hat{h}_2 \ldots \hat{h}_n]$

$$\tilde{h}_i = \begin{cases} \hat{h}_i + \gamma\frac{R_i - \hat{h}_i}{D}e^{-n_T\lambda} & \text{with likelihood 0.5} \\ \hat{h}_i - \gamma\frac{\hat{h}_i - L_i}{D}e^{-n_T\lambda} & \text{with likelihood 0.5} \end{cases} \quad (7)$$

where γ is a random number between 0 and 1, n_T is the number of temperature reductions, D is the divisor found in Eq. (6). The non-uniformity parameter, λ, yields most uniformity at the value of 0 and less uniformity at higher values.

4. EXPERIMENTAL RESULTS AND CONCLUSION

For all algorithms, the initial temperature was set to 0.2 (according to the selection scheme proposed by Yong et al. [4]). The other parameters selected for algorithm 1 were $N_T = 5 \times M \times N$, i.e. for an 8×8 filter $N_T = 5 \times 8 \times 8 = 320$, $N_s = 20$, and $r = 0.85$. For algorithm 2 we selected $N_T = 20 \times M \times N$, $N_s = 40$, and $r = 0.85$. For the annealing evolution algorithm we selected $L = 50$, $N_P = 100$, $r = 0.9$ and $\lambda = 0.5$.

If the temperature is zero, simulated annealing performs a search similar to gradient search. However, the search directions are random, and not systematic as with gradient search. Hence, using an explicit gradient search, instead of simulated annealing is much more efficient. To refine the results we therefore finalize by a gradient search.

We tested the algorithms on optimizing filters of sizes up to 8×8. The pairs of textures we used were the Brodatz D5-D24, D5-D92, and D24-D92 [6].

In most cases the algorithms found better filters than the closed form solution followed by gradient search [1]. The annealing evolution algorithm executes many times faster than algorithm 1 and algorithm 2, but is slower than the technique in [1]. Annealing evolution may furthermore be speeded up by implementing the inherent paralellism in the problem. However, algorithm 1 gave overall best result of all of the approaches.

Of course, three texture pairs is not a large test set, but nevertheless the tests show that random search may give improvements to previously reported optimization techniques, at the cost of increased computational time. The number of tests were limited due to the computational complexity of the problem. A more detailed description of the experimental results, along with more details on the algorithms may be found in [7].

REFERENCES

[1] T. Randen and J. H. Husøy, "Optimal texture filtering," in *Int. Conf. on Image Proc.*, (Washington, D.C.), pp. 374–377, Oct. 1995.

[2] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis*. New York, NY, USA: Wiley, 1973.

[3] A. Corana, M. Marchesi, and S. Ridella, "Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm," *ACM Transactions on Mathematical Software*, vol. 13, pp. 262–280, September 1987.

[4] L. Yong, K. Lishan, and D. Evans, "The annealing evolution algorithm as function optimizer," *Parallel Computing*, no. 21, pp. 389–400, 1995. Elsvier Science B.V.

[5] L. Davis, *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold, 1991.

[6] P. Brodatz, *Textures: A photographic album for artists and designers*. NY: Dover, 1966.

[7] F. Nergård, "Random search optimization of filters for texture identification in digital images," Master's thesis, Høgskolen i Stavanger, 1996.