An example is given in Fig. 2 for the three level zero mapping. The search for zeros starts from the highest level. The first pixel at the third level is a zero. Its corresponding pixels at lower levels (child pixels) are then checked. All child pixels are zero here. In this sense, the zero in a higher level is 'mapped' into lower levels. We record this case by using a symbol '0' at the third level and its child pixels will not be coded. The second and third pixels at the third level are not zeros and the zero mapper will not change them. The fourth pixel is again a zero, but some of its child pixels are not zeros. This is recorded by using the symbol 'K' to indicate that the zero mapping is not applicable here. The same process is then repeated at the second level for the pixels, which have not been zero-mapped from the third level. The first level will not be searched, since these pixels do not have any children. When the zero searching is completed, the 'map encoder' removes the pixels which can be zero-mapped, as represented by the empty space in Fig. 2. Unlike the zerotree technique [3], the zero mapping does not require the coding of the positions of those nonzero coefficients. The low frequency component, A3, does not participate in the zero mapping, since it typically contains few zeros. These simplify the coding process.

Before the data are processed by the adaptive arithmetic encoder, two methods are used to improve its efficiency. First, the data in the sub-band files containing vertical edges (V1, V2 and V3) are reordered by scanning in the vertical direction, because the change of vertical edges in the vertical direction is generally slower. Similarly, the data in files containing the diagonal features (D1, D2 and D3) are rearranged by scanning in the diagonal (zigzag) direction. To determine the scanning direction of the low frequency file, A3, the energy of sub-bands containing horizontal (H1), vertical (V1), and diagonal (D1) edges is calculated. A3 is scanned in the direction of highest energy. Secondly, all sub-band files are packed into one file, which can further enhance the coding efficiency.

Results and conclusions: To evaluate the proposed algorithm, the well-known Lena and Barbara images [3] were chosen for the test. The former is rich in textures and segments, while the latter has many high frequency features. For both images, Fig. 3 shows that the new algorithm significantly outperforms JPEG. It is also noted that for the same peak signal-to-noise ratio (PSNR), the blocking effects are absent here, while they become unacceptable when compressed by JPEG. The new algorithm also outperforms the embedded zerotree wavelet (EZW) algorithm. The improvement for Barbara is better than that for Lena, indicating that the new algorithm can code the high frequency features efficiently. These results show the great potential of wavelet transforms in image compression.

25 August 1999
Electronics Letters Online No: 19991305
DOI: 10.1049/el:19991305

R.H.G. Tan, J.F. Zhang, R. Morgan and A. Greenwood (*School of Engineering, Liverpool John Moores University, Byrom Street, Liverpool L3 3AF, United Kingdom*)

References

1 WALLACE, G.K.: 'The JPEG still picture compression standard', *Commun. ACM*, 1991, **34**, pp. 30–44

2 MALLAT, S.G.: 'A theory for multiresolution signal decomposition: the wavelet representation', *IEEE Trans.*, 1989, **PAMI-11**, pp. 674–693

3 SHAPIRO, J.M.: 'Embedded image coding using zerotrees of wavelet coefficients', *IEEE Trans.*, 1993, **SP-41**, pp. 3445–3462

4 SAID, A., and PEARLMAN, W.A.: 'A new, fast and efficient image codec based on set partitioning in hierarchical trees', *IEEE Trans. Circuits Syst. Video Technol.*, 1996, **6**, pp. 243–250

5 EFSTRATIADIS, S.N., TZOVARAS, D., and STRINTZIS, M.G.: 'Hierarchical partition priority wavelet image compression', *IEEE Trans.*, 1996, **IP-5**, pp. 1111–1123

Supervised texture segmentation using support vector machines

K.I. Kim, K. Jung, S.H. Park and H.J. Kim

An approach to the problem of supervised texture segmentation using nonlinear support vector machines (SVMs) is presented. For each texture class a nonlinear SVM is constructed which separates that class from the other classes. The segmentation then works by applying all the SVMs to an input image and arbitrating between the SVM outputs. Experimental results show the effectiveness of the proposed method.

Introduction: In the past few decades, methods for analysing texture in digital images have received considerable attention [1]. Among these methods, filtering is attractive because of its simplicity [2]. Using this method, a high dimensional image can be represented by a relatively small set of feature statistics, and these are usually extracted by a set of well-selected filters. However, the selection of a good set of filters is not an easy task. Recently, approaches have been proposed in which the filtering scheme has been embedded in the neural network architecture to produce texture segmentation [3, 4]. As a result, the problem of obtaining a filter set for a given texture classification task can be reduced to the training of neural networks, such as multilayer perceptrons (MLPs) and generalised Hebbian networks (GHNs). Accordingly, the generalisation performance of neural networks is an important factor for the segmentation results in this scheme. Support vector machines (SVMs) have been recently proposed for pattern classification and nonlinear regression [5]. Their appeal lies in their strong connection with the underlying statistical learning theory. That is, an SVM is an approximate implementation of the method of structural risk minimisation [6]. For several applications, SVMs have been shown to provide a better generalisation performance than traditional techniques, including neural networks [7].

In this Letter we present an approach in which nonlinear SVMs are used to solve the problem of supervised texture segmentation. For each texture class an SVM is constructed which separates that class from all the other classes. The segmentation works through applying all the SYMs to an input image and arbitrating between each SVM output. The SVMs and arbitrator are all trained according to the supervised learning method to minimise any texture classification error. Experimental results suggest that the proposed method could be applied to real-world applications.

Nonlinear SVMs: Consider a two-class pattern classification problem. Let the training set of size N be $\{(\mathbf{x}_i, d_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \Re^m$ is the input pattern for the ith example and $d_i \in \{-1, +1\}$ is the corresponding desired response. The nonlinear SVM first performs a nonlinear mapping ϕ: $\Re^m \rightarrow \Re^{m_1}$. Let $\{\phi_j(\mathbf{x})\}_{j=1}^{m_1}$ denote a set of nonlinear transformations from the input space $\Re^m$ to the feature space $\Re^{m_1}$. This mapping can be represented as follows:

$$\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_{m_1}(\mathbf{x})]^T \quad (1)$$

An SVM can be trained to construct a hyperplane $\mathbf{w}^T\Phi(\mathbf{x}) + b = 0$ for which the margin of separation is maximised. Using the method of Lagrange multipliers, this hyperplane can be represented as

$$\sum_{i=1}^{N} \alpha_i d_i \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}) = 0 \quad (2)$$

where the auxiliary variables α_i are Lagrange multipliers. These α_is can be found by solving the following problem:

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j d_i d_j K(x_i, x_j) \quad (3)$$

subject to the constraints

$$\sum_{i=1}^{N} \alpha_i d_i = 0 \quad (4)$$

$$0 \leq \alpha_i \leq C \quad \text{for} \quad i = 1, 2, ..., N \quad (5)$$

where C is a constant which is used to balance contributions from the first and second terms. This performs a similar role as the regularisation parameter in the radial basis function [6] and affects the generalisation performance of the SVM. The value of this parameter is determined empirically to be 10. $K(\mathbf{x}_i, \mathbf{x}_j)$ is the inner-product kernel defined by

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j) \quad (6)$$

One of the most commonly used forms of this kernel is a polynomial. We may use a polynomial kernel of degree 3 defined by

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T\mathbf{x}_j + 1)^3 \quad (7)$$

From the Kuhn-Tucker condition, for training examples which lie within the decision boundary only, the corresponding multipliers are nonzero. We can observe that only these examples, called support vectors, affect the construction of the hyperplane.

The operation performed by an SVM is similar to that of a feedforward neural network. Furthermore, if a tangent hyperbolic or radial function is selected for the inner-product kernel, an SVM can be operated exactly like a two-layer perceptron or radial basis function network (RBFN) [6]. However, the underlying theory of SVMs avoids the need for heuristics, which are often used in the design of conventional neural networks. That is, the number of hidden neurons and their weight vectors are automatically and optimally determined according to the number of support vectors and their values, respectively.

Textured image segmentation: To date, we have focused on the discussion of SVMs for the solution of two-class problems. However, for texture classification, it is necessary to choose between R (≥ 2) classes. The optimal design for multi-class SVM classifiers is still an active area of research. One frequently used method is the 'one versus all' approach. This works by constructing an SVM for each class, which separates that class from all the other classes, and then using an expert to arbitrate between outputs of each SVM in order to produce the final decision. Fig. 1 shows the architecture of the texture classifier. The input to the classifier comes from an $M \times M$ window in the input image and each of the R output units of the classifier represents the class of the central pixel in the input window. Following the configuration of the autoregressive texture model [2], only the shaded pixels from the input window are fed to the classifier. The kernel number n_r for each rth SVM is the same as the number of support vectors obtained when training for classifying texture r and the other classes. The output of each SVM is normalised as [-1, 1] by applying a tangent hyperbolic function. An MLP is used as the arbitrator. It has one hidden layer composed of two neurons. All the neurons between each adjacent layer are fully connected. Accordingly, the proposed texture classifier has a three-layer architecture as shown in Fig. 1.

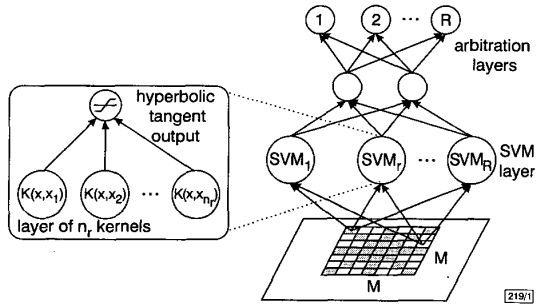

Fig. 1 *Architecture of texture classifier*

The training of the texture classifier is a two-stage process. The SVMs are first trained with raw textures and the arbitrator is then trained with the SVM outputs using the back-propagation algorithm. The texture image segmentation works through the application of the texture classifier to all pixels in the input image and the smoothing of the resulting 'classification image'. A median filter of size 5×5 is used so that spurious image variations are removed.

Experimental results: To demonstrate the effectiveness of the proposed method, an artificially generated texture image from the Brodatz texture set was segmented. This image was composed of 16 different textures (Fig. 2). The size of the image was 512×512 and the size of each textured subimage was 128×128. For the training and testing of the texture classifier, the grey scale of the input image was normalised to [0, 1]. The training data set was obtained by randomly selecting 1000 of the input patterns from each texture. These correspond to ~6% of the input patterns obtained from the whole input image. Patterns lying on the boundaries were excluded from the training set because in this case it is difficult to assign the desired value for the training pattern. The experiments were performed with input window sizes 7×7, 11×11, 15×15 and 19×19. As summarised in Table 1, the correct segmentation rates according to the texture classifier alone were 78.0, 81.8, 83.7 and 79.5%, respectively; however, with smoothing they were 90.8, 92.8, 93.4 and 91.1%, respectively. The rather low segmentation rate for the classifier that used the 19×19 input window can be attributed to the instability of classification for the patterns lying on the texture boundaries. Fig. 2 shows the result of image segmentation using the 15×15 input window.

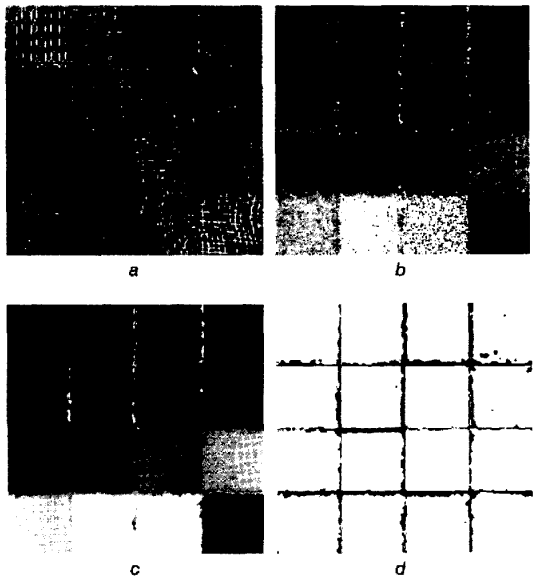

Fig. 2 *Example of texture segmentation*

a Image composed of 16 textures
b Classification result using texture classifier alone
c Segmentation result with smoothing
d Misclassified pixels illustrated as black grey levels

Table 1: Correct segmentation rate

Window size	Using texture classifier alone	With smoothing
	%	%
7×7	78.0	90.8
11×11	81.8	92.8
15×15	83.7	93.4
19×19	79.5	91.1

Conclusions: We have presented a supervised texture segmentation method using SVMs. This method works by constructing a texture classifier which is composed of several 'one versus all' SVMs. Experimental results indicate that the proposed method is likely to be very useful for supervised texture segmentation and confirm the excellence of the generalisation performance of SVMs.

Some of the main applications of this method include page-layout segmentation, and text location in video image and object detection. Accordingly, future work will include experiments on these real-world applications, as well as further studies into the fine-tuning of parameters such as input window size and the structure of the arbitrator.

7 September 1999
Electronics Letters Online No: 19991317
DOI: 10.1049/el:19991317

K.I. Kim, K. Jung, S.H. Park and H.J. Kim (*Department of Computer Engineering, Kyungpook National University, 1370, Sangyuk-dong, Pukgu, Taegu, 702-701, Republic of Korea*)

References

1 TUCERYAN, M., and JAIN, A.K.: 'Texture analysis' *in* 'Handbook of pattern recognition and computer vision' (World Scientific, 1993)
2 RANDEN, T., and HUSOY, J.H.: 'Filtering for texture classification: a comparative study', *IEEE Trans.*, 1999, **PAMI-21**, (4), pp. 291–310
3 JAIN, A.K., and KARU, K.: 'Learning texture discrimination masks', *IEEE Trans.*, 1996, **PAMI-18**, (2), pp. 195–205
4 PATEL, D.: 'Page segmentation for document image analysis using a neural network', *Opt. Eng.*, 1996, **35**, (7), pp. 1854–1861
5 HAYKIN, S.: 'Neural network: A comprehensive foundation' (Prentice Hall, 1999), 2nd edn.
6 VAPNIK, V.: 'The nature of statistical learning theory' (Springer Verlag, 1995)
7 SCHOLKOPF, B., SUNG, K., BURGES, C., GIROSI, F., NIYOGI, P., POGIO, T., and VAPNIK, V.: 'Comparing support vector machines with Gaussian kernels to radial basis function classifiers'. A.I. Memo 1599, MIT, December 1996

Comments on password-based private key download protocol of NDSS'99

Seungjoo Kim, Byungchun Kim and Sungjun Park

The weaknesses of the password-based protocol for downloading a private key presented by Perlman and Kaufman at NDSS'99 are discussed.

Introduction: In a recent paper [1], Perlman and Kaufman describe a password-based private key download protocol, which is a method for the client machine Alice with a guessable password to securely retrieve her public and private key from the server Bob. Their scheme is based on EKE [2] and SPEKE [3], with modifications to improve the security and the performance of NetWare version 4 [4, 5]. The Perlman-Kaufman protocol enables Alice to be able to log-in from a completely generic server with no previous knowledge of the user or any specific server public keys.

In this Letter we show that, unlike the protocol of [2, 3], the Perlman-Kaufman protocol does nothing to protect the password against the Denning-Sacco attack. Furthermore, in [1], Perlman *et al.* claim that there is no security advantage to strengthening their schemes against the stolen verifier attack because Alice's private key is encrypted under her password. But we will also show this is not true.

Password-based private key download protocol: We only describe the first protocol, i.e. 'Basic 4-msg EKE-based' scheme, in [1]. The server Bob has a database and knows, for each user: user name, $W = h(pwd)$, Y = user's private key encrypted with her password. The client Alice initially knows nothing:

(i) Alice types her password *pwd*, and then the workstation computes $W = h(pwd)$. Alice chooses random A and sends to Bob "'Alice', $\{g^A \bmod p\}$ encrypted with W".

(ii) Bob with W chooses B randomly, calculates $K = g^{AB} \bmod p$, and sends to Alice $\{g^B \bmod p\}$ encrypted with W.

(iii) Alice computes a hash image and sends message $h(K)$ to Bob.

(iv) Bob then downloads $\{Y\}$ encrypted with K.

Vulnerability to Denning-Sacco attack : The Denning-Sacco attack [2, 6] proceeds as follows: the attacker somehow obtains one of the session keys distributed in one record run of the keys distribution protocol. Armed with that knowledge, the attacker mounts a dictionary attack on the password and, on breaking the password, is able to impersonate one of the parties indefinitely. In all of the proposed protocols in [1], the server Bob sends the last message private key encrypted with her password. So the following attack is possible:

(1) The attacker records one run of the key distribution protocol and somehow obtains the old session key K.

(2) Iterating all possible choices of password P:

(i) Choose a candidate $\bar{P}$.

(ii) Compute $\bar{Y} = D_K(E_K(Y))$ where $E_K(Y)$ is taken from the last flow of the recorded run.

(iii) Decrypt $\bar{Y}$ with $\bar{P}$.

(iv) Check whether the decrypted private key corresponds to Alice's public key, e.g. raise the decrypted private key to g and verify whether it equals Alice's public key. A match in the last step indicates a correct guess of the password.

Vulnerability to stolen verifier attack: A stolen verifier attack [7] occurs when an intruder learns the server's password file and uses it to impersonate the user directly without an expensive dictionary attack. In most schemes in [1], both the client Alice and the server Bob use $W = h(pwd)$ directly, so the stolen verifier W can be used to masquerade not only as a server but also as a client. That is, their schemes are vulnerable not only to the server masquerade attack but also to the client masquerade attack. Furthermore, since the versions in §3.3 and §3.4 in [1], which are more computationally efficient, have Bob always using the same random number B, an attacker obtaining one of the session key can launch the following attack:

(1) An intruder learns the server's password file. Furthermore, he records one run of the key distribution protocol and somehow obtains the old session key K.

(2) In the case of the '4-msgs, stateless Bob, precomputation, EKE-based' protocol (in §3.3 of [1]), Bob needs to store, for each user: (username, Y, B, $E_w(g^B \bmod p)$), where B is the Alice-specific random number chosen by Bob when setting up Alice's account and $W = h(pwd)$.

(i) The attacker impersonating Alice sends 'Alice' to Bob.

(ii) Bob calculates $R = h(IP\ address, Bob's\ secret)$, and transmits $E_W(g^B \bmod p)$ and R.

(iii) The attacker picks a random number r and sends to Bob R, $(g^A)^r \bmod p$, 'Alice', $h(K^r)$, where g^{AE} is taken from the third flow of the recorded run.

(iv) Bob verifies $R = h(IP\ address, Bob's\ secret)$, computes $(g^{Ar})^B$, and verifies $h((g^{Ar})^B)$. Then Alice and Bob uses $(g^{Ar})^B$ as a new session key for the continuous session. The result is that, without a dictionary attack, the attacker can impersonate Alice directly and also compute a new session key continuously. A similar attack can also be applied in §3.4 of [1].

Conclusion: We have shown in this Letter that the Perlman-Kaufman private key download protocol is subject to a simple password guessing attack if the attacker can obtain an old session key. Furthermore, contrary to their assertions, the Perlman-Kaufman protocol needs to be strengthened against the stolen verifier attack.

1 September 1999
Electronics Letters Online No: 19991333
DOI: 10.1049/el:19991333

Seungjoo Kim, Byungchun Kim and Sungjun Park (*KISA (Korea Information Security Agency), 5th Floor, Dong-A Tower, 1321-6, Seocho-Dong, Seocho-Gu, Seoul 137-070, Korea*)

E-mail: skim@kisa.or.kr

References

1 PERLMAN, R., and KAUFMAN, C.: 'Secure password-based protocol for downloading a private key'. Proc. NDSS'99, 1999 Network and Distributed System Security Symp., February 1999,
2 BELLOVIN, S.M., and MERRITT, M.: 'Encrypted key exchange: password-based protocols secure against dictionary attacks'. Proc. IEEE Symp. Research in Security and Privacy, Oakland, May 1992
3 JABLON, D.: 'Strong password-only authenticated key exchange', *Comput. Commun. Rev., ACM SIGCOMM*, 1996, **26**, (5), pp. 5–26
4 LEE, R., and ISRAEL, J.: 'Understanding the role of identification and authentication in NetWare 4. Novel Application Notes, October