FAST IMAGE ANALYSIS USING KOHONEN MAPS

D.Willett, C. Busch, F. Seibert,
Visual Computing Group
Darmstadt Computer Graphics Center
Wilhelminenstraße 7, D 64283 Darmstadt
Tel: +49 6151 155 255, Fax: +49 6151 155 299
E–mail: busch@igd.fhg.de

Abstract – The following paper considers image analysis with Kohonen Feature Maps. These types of neural networks have proven their usefulness for pattern recognition in the field of signal processing in various applications. The paper reviews a classification approach, used in medical applications, in order to segment anatomical objects such as brain tumors from magnetic resonance imaging (MRI) data. The same approach can be used for environmental purposes, to derive land–use classifications from satellite image data. These applications require tremendous processing time when pixel–oriented approaches are chosen. Therefore the paper describes implementation aspects which result in a stunning speed–up for classification purposes. Most of them are based on geometric relations in the feature–space.

The proposed modifications were tested on the mentioned applications. Impressive speed–up times could be reached independent of specific hardware.

1 INTRODUCTION

Image classification is a crucial step in the image processing pipeline. Typical applications for image classification are the interpretation of medical data or remote sensing data. In medical applications modern image acquisition techniques like magnetic resonance imaging (MRI) supply 3D data sets of high resolution and quality. 3D data need to be classified in order to separate different tissue types such as brain tumors in the image data. Further processing will use the classified data as input for 3D reconstruction algorithms of the volume. Advanced volume renderers, as in [5],[9] or [11] require opacity–maps, as well as 3D surface reconstruction methods like marching cubes [10] or Delaunay triangulation [12] require a description of the surface, which can be easily derived from the classified data.

Environmental applications use remote sensing data in order to achieve semantic ground information. Remote sensing data stemming from satellite–based sensors like SPOT, Landsat–TM or ERS1 can serve as multispectral data input for environmental control systems. Robust image analysis of the data delivers a land–use–classification.

The following paper reviews in Section 2 a pixel–oriented discrimination method for image data based on topological mappings of Kohonen[7],[8]. Application studies as in [4] and [1] have shown that this method can either perform feature extraction, clustering and classification in a unique approach or – in a more traditional manner –separate the single steps and calculate the independent components of the classification pipeline [13],[2].

Corresponding to the immense amount of data that is analyzed in the above mentioned applications, the use of the Kohonen Feature Map can be extremely expensive, especially when long feature–vectors are considered. Thus Section 3 proposes several modifications of Kohonen's algorithm. They were inspired by the insight, that most of the computational time is wasted in calculations of Euclidean distances in the feature–space in order to determine the neuron which bears the closest weight–vector to a presented input–vector. This task corresponds to the nearest neighbor search in a multidimensional space. It can be shown that most of the weight–vectors could be excluded from consideration.

In addition basic constraints can be used, like the similarity of consecutive input–vectors as they appear in the classification of consecutive pixels stemming from a homogenous region in an image. Those modifications can be used for clustering or classification tasks. For classification purposes further optimization is possible, since only a neuron's class assignment has to be determined and not the closest neuron itself.

The modifications were implemented and tested on the described applications. Section 4 reports some results in terms of speed–up times. The times are not based on any specific hardware, but nevertheless computing the image classification on powerful hardware like vector– or parallel–processors can realize further speed–up.

2 CLUSTERING AND CLASSIFICATION

2.1 General Remarks

In general, a robust classification pipeline for the automatic recognition, classification and visualization of the data can be divided into the following three tasks:

I) feature extraction
II) cluster analysis
III) supervised classification

Those steps can be solved separately, or in a single approach. Kohonen Feature Maps are capable of solving that task in a unique paradigm, since they allow subspace mapping, visualization of a multidimensional texture feature–space and supervised classification. This is explained in detail in [1] and [3].

2.2 Kohonen Mapping

The Kohonen Map as introduced in [7] or [8], is a self–organizing network which is basically trained without supervision. It organizes a set of input patterns in a topological structure represented by neurons, where the relations between different patterns are preserved.

To use the Kohonen map for cluster analysis, the Kohonen Map can be configured with a 3D output–layer as shown in Figure 1. The neurons in the input–layer pick up the data from the feature–extractor or directly from the image. The weights associated with each connection of an output–layer's neuron are adjusted during training where only one single neuron can be active at a time. A time–dependent neighborhood implies an update in the neuron's environment as well. After the self–organizing training–procedure, each neuron in 3D represents a cluster in the multidimensional feature–space. Therefore the network can be used for cluster analysis and dimensionality reduction as described in [3].

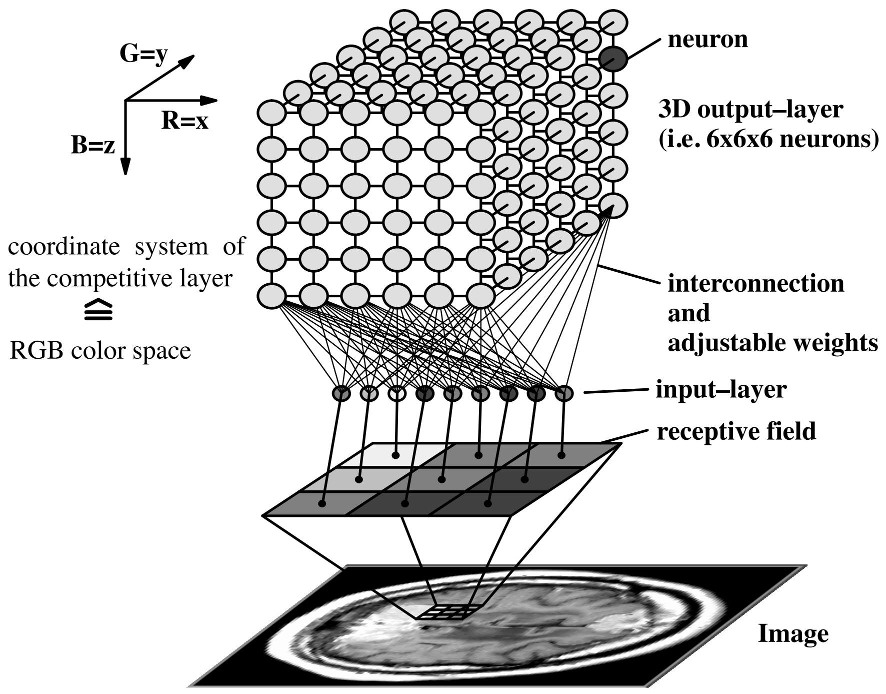

Fig. 1: Topology of the 3D–Kohonen map.

Essential for training as well as for the work procedure of the network is the spatial distance in the feature–space, since it decides which neuron in the output–layer is activated. For a presented data–vector **x** the distance is calculated to all weight–vectors $\mathbf{m}_i$ representing the connections between the input–layer and the competitive layer. If N is the dimension of the data, the Euclidean distance d_i between **x** and $\mathbf{m}_i$ is defined as

$$d_i = \| \boldsymbol{x} - \boldsymbol{m}_i \| = \sqrt{\sum_{j=1}^{N} (x_j - m_{ij})^2} \qquad (1)$$

The neuron *c* with the minimum distance is activated, where

$$d_c = \min_i \{d_i\} \qquad (2)$$

The activated neuron is of fundamental importance. On one hand in its environment the weight–vectors will be updated according to the training rules [7]. On the other hand its coordinate will be taken as RGB–information for clustering tasks or its class assignment will be taken as class information for a classification task. Fast and effective calculation of a winning neuron will be the subject of the next Section.

In order to use a self–organized Kohonen Feature Map for supervised classification a class assignment for each neuron is required. User defined training areas can set up a training set, where a class–assignment exists for each sample, i.e. a feature–vector. Sequentially presentation of labeled input–vectors and subsequent majority voting can lead to class–assignment of each output neuron. For optimal description of the Bayes decision boundary, additional postprocessing with learning vector quantization (LVQ) is recommended [7].

3 SPEED–UP METHODS

3.1 Remarks

The most time consuming part of the classification is the determination of the neuron in the output–layer which is activated by a given input. It is the neuron whose weight–vector is the closest to the input–vector in the feature–space.

According to equation (1) and (2) the activated neuron is determined by calculating the exact Euclidian distances of each weight–vector to the input–vector and selecting the one with the least distance. Slight changes of this process can bring the first improvements.

Avoiding the square–root–function. In order to calculate the Euclidean distance of two vectors, the square–root–function is used. For the given purpose only the relation between the Euclidean distances is of importance. Thus we simply change equation (1) and (2) and calculate the squared distances

$$d_i^2 = \| \boldsymbol{x} - \boldsymbol{m}_i \|^2 = \sum_{j=1}^{N} (x_j - m_{ij})^2 \qquad (3)$$

and determine the activated neuron c with

$$d_c^2 = \min_i \left\{ d_i^2 \right\} \qquad (4)$$

So in the following text the *distance* in general refers to the square of an Euclidian distance.

Furthermore the term *best vector* will be used for the temporary closest vector found. A *good vector* in general will be any weight vector close to the input–vector.

Threshold Summation. During the calculation of the distances, the *best* distance can be used as a threshold [6]. If, while performing the summation in formula (3) we reach a sum greater than the threshold, even if j < N, then we can stop and disregard this vector.

Usage of Correlated Input. The threshold summation and other optimization given in this text are most effective whenever a *good* vector is found early during the calculation, so that the determination of the exact distance of many other vectors can be avoided. In a lot of applications the input presented in one step correlates to the input presented in the previous step. If such a correlation is obvious, as in the pixel-based image–segmentation, the activated neuron of the previous step should be considered first in the activation–algorithm.

3.2 Immediate Activation

Once a map has been trained, its weight–vectors remain static. The distances between the weight–vectors themselves can be calculated in advance in order to be used to optimize the calculations. A first approach is illustrated in the figure below which refers to a 2–dimensional feature–space. The circle around each weight–vector has a radius r_i of half the minimum distance to all other vectors.

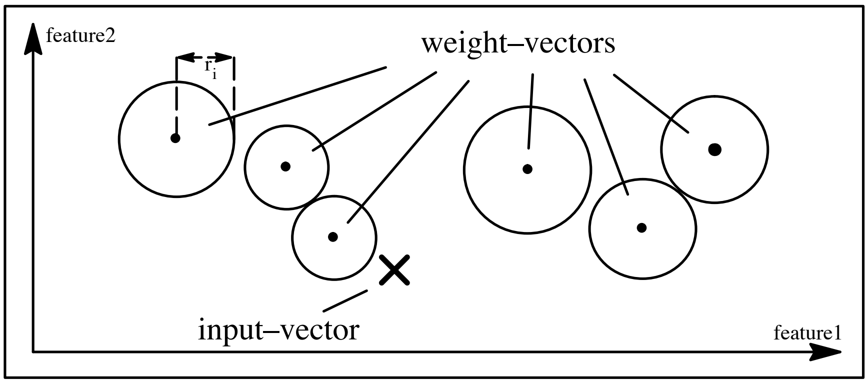

Fig. 2: Areas of immediate activation around the weight–vectors in the feature–space

Once an input–vector turns out to be situated inside of an activation area,

$$d_i^2 \le r_i^2 \quad \Rightarrow \quad d_i \le r_i \quad \Rightarrow \quad d_i = \min_j \left\{ d_j \right\} \qquad (5)$$

the closest weight–vector is the one in the center of that area. Thus the activated neuron is found.

When using labeled neurons, like for classification purposes, only the label of the activated neuron is relevant. Areas of neurons with the same label may overlap. In Figure 3 the labels of the neurons are represented by different shadings of the areas.

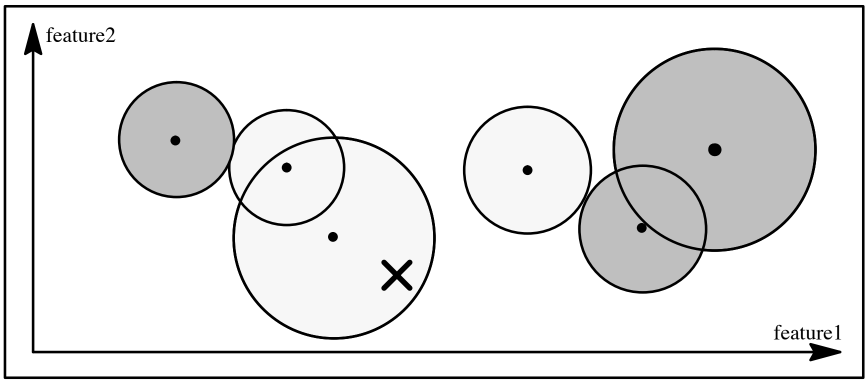

Fig. 3: Areas of immediate classification (identical shading refers to identical classes)

The immediate activation does not give an idea of how to find the weight–vector in whose circle the input might be situated. In the worst case it is the last vector that is being considered and the immediate activation does not save time at all. For this reason Section 3.3 and 3.4 present two relations which can be used in addition to the immediate activation to avoid the determination of several distances.

3.3 Triangle Relation

From a mathematical point of view we consider the feature–space as a metric space. In a metric space the triangle–relation (6) is valid, where d is the distance between two points.

$$d(A, C) \leq d(A, B) + d(B, C) \tag{6}$$

Looking at any two weight–vectors $\mathbf{m_1}$ and $\mathbf{m_2}$ and an input–vector $\mathbf{x}$, where the vectors express the coordinates of points in the feature–space, the relation can be transposed to (7).

$$d(\boldsymbol{x}, \boldsymbol{m}_2) \geq d(\boldsymbol{m}_1, \boldsymbol{m}_2) - d(\boldsymbol{x}, \boldsymbol{m}_1) \tag{7}$$

Assuming that the distance between $\mathbf{m_1}$ and $\mathbf{m_2}$ is at least twice as high as the distance between $\mathbf{m_1}$ and $\mathbf{x}$, it can be concluded that in no case $\mathbf{m_2}$ can be closer to $\mathbf{x}$ than $\mathbf{m_1}$.

$$d(\boldsymbol{m}_1, \boldsymbol{m}_2) \geq 2\, d(\boldsymbol{x}, \boldsymbol{m}_1) \Rightarrow d(\boldsymbol{x}, \boldsymbol{m}_2) \geq d(\boldsymbol{x}, \boldsymbol{m}_1) \tag{8}$$

The activation–algorithm can make use of this circumstance by not considering those vectors whose distance to the actual *best* vector is at least twice that high as the distance of the *best* vector to the input–vector. For the case that the squared distances are determined, equation (8) can be transposed to:

$$d^2(\boldsymbol{m}_1, \boldsymbol{m}_2) \geq 4\, d^2(\boldsymbol{x}, \boldsymbol{m}_1) \Rightarrow d(\boldsymbol{x}, \boldsymbol{m}_2) \geq d(\boldsymbol{x}, \boldsymbol{m}_1) \tag{9}$$

where $d(\mathbf{x},\mathbf{m_i})$ corresponds to d_i in equation (3).

3.4 Minimum Distances Derived from the Vector–Sums

In Section 3.3 we used the triangle relation to derive a minimum value for a specific distance. Once this minimum value turns out to be greater than the *best* value to that point of the calculation, the exact distance does not need to be determined anymore.

Another possibility which allows the determination of such a minimum value with only a few calculations is based on the absolute sum–values of the vectors. Using the relation

$$d^2(\,\boldsymbol{m}_1\,,\,\boldsymbol{m}_2\,)\ \geq\ \frac{\left(\sum \boldsymbol{m}_1 - \sum \boldsymbol{m}_2\right)^2}{N} \qquad (\,10\,)$$

$$\text{where}\quad \sum \boldsymbol{m}_i \;:=\; \sum_{j\,=\,1}^{N} m_{ij} \quad\text{and}\quad \boldsymbol{m}_i := (m_{i1}, m_{i2}, \ldots, m_{iN})$$

one gets another lower boundary for the distance of two vectors, which can be derived from the sum–values of the vectors. The correctness of relation (10) is fairly easy to prove.

3.5 Maximum Likelihood–Search

The immediate activation presented in Section 3.2 offers a fast way to determine the closest vector. The time needed to find out that an input is inside an area of immediate activation mainly depends on the order in which the vectors are looked at. In the best case the closest vector is considered first and the immediate activation prevents the determination of all other distances. In the worst case the closest vector is being looked at after all other vectors, so that the immediate activation does not help at all.

The main aspect of the Maximum Likelihood–Search is the usage of the minimum values presented in Sections 3.3 and 3.4 to control the order in which the vectors are considered. The idea is that the vector with the least minimum value has the highest probability to be the vector that is being searched for.

3.6 Approximate Determination of a *Good* Vector

As mentioned in Section 3.1 the success of most of the presented methods largely depend on how fast a *good* vector is found. If there is no correlation between the consecutive input–vectors, or if the correlation is weak, another method can be applied. The activation is split up into a two–pass algorithm. During the first pass an approximation is applied to determine a *good* vector as an approximate solution. The second pass should be handled like a normal (optimized) activation–algorithm, starting with the *good* vector found in the first pass.

A method to determine a *good* vector shall now be suggested. It allows the fast determination of a vector which is at most *n* times farther away from the input–vector than the accurate closest vector. The parameter *n* can be set to any real value greater than or equal to one.

The basic idea of the approximation is the application of the triangle–relation (3.3) and the vector–sums (3.4) to mark those vectors that certainly cannot be more than *n* times closer to the input than the *best* vector found so far. The marked vectors do not need to be considered during the further approximate pass of the algorithm. Once all vectors are marked or have been considered, the *best* vector found to that point is at most *n* times farther away from the input than the accurate closest vector.

3.7 N–Tree Based Search

Another example of a fast nearest neighbor search is the organization of the given weight–vectors in an n–dimensional Quadtree. Following this approach the weight–vectors are ordered according to their positions in hierarchically refined hypercubes, where each cube contains up to 2^n cubes of half of its edge lengths. This subdivision is done until a fixed number of vectors is inside the cube. In this way the depth of the subdivision is controlled by the density of the vectors in the feature–space.

The regular box oriented structure allows a reduced calculation of distances by a privileged search for neighbors in related cubes. Additionally the methods described in Sections 3.1 and 3.4 can be used for a further reduction of time consuming calculations.

As shown in Section 4 the usability of N–Trees for next neighbor search depends on the number of output neurons and the dimension of the input data. High dimensional input data accompanied by only a small number of output neurons causes a complex and sparse internal structure which results in low performance. The reverse case of lower data dimension and a large number of vectors shows the advantage of N–Trees based search in comparison to methods described in previous Sections.

4 APPLICATION AND RESULTS

4.1 General Remarks

The methods outlined in Section 3 were tested on two different applications where both deal with multidimensional image data. Both applications face an 8–class problem.

The medical application aims at segmentation of brain tumors in MRI–Data which is required in order to gain knowledge about localization and extension of the tumor in the skull. That knowledge can be used as input for 3D–renderers of 3D–surface reconstruction algorithms. Section 4.2 refers to a volume data set of 5 slices, where each slice is recorded as a two–channel image of size 256x256 pixels. The environmental application uses satellite image data, which was recorded from Landsat–TM. For environmental control the six–channel image data was analyzed with 1000x100 pixels in each channel. The classified image data outlines the actual land–use and illustrates the impact of pollution sources in the environment.

4.2 Measured Times of Calculation

The success of the different methods given in Section 3 widely depends on the data that the Kohonen–Map is used for. Furthermore it depends on the size of the feature–vector. In order to give an idea of the amount of possible optimization, the following table shows the measured times of calculation for different kinds of image–segmentations on a HP–Workstation 720 and a DEC–Workstation 5000/240.

Image data / Method applied	Clustering 5 slices of 256x256 Pixels with a map of 54 input–neurons and a 6x6x6 output–layer on a HP 720	Classifying 5 slices of 256x256 Pixels with a map of 54 input–neurons and a 6x6x6 output–layer on a HP 720	Classifying a Picture of 1000x100pixels with a map of 54 input–neurons and a 6x6x6 output–layer on a DEC 5000/240	Classifying a Picture of 1000x100 pixels with a map of 6 input–neurons and a 9x12x18 outputlayer on a DEC 5000/240
Determination of all distances while avoiding the square–root–function (see 3.1)	**57.10** minutes	**57.10** minutes	**20.97** minutes	**26.22** minutes
All aspects from 3.1, Immediate Activation (see 3.2), Triangle Relation (see 3.3)	**4.32** minutes	**3.33** minutes	**11.15** minutes	**15.15** minutes
As above and additionally use of Relation (10) (see 3.4)	**4.11** minutes	**3.03** minutes	**5.72** minutes	**10.11** minutes
Maximum Likelihood–Search (see 3.5)	**4.39** minutes	**4.24** minutes		
2–pass determination with an approximating first pass (see 3.6)	**6.53** minutes	**4.00** minutes		
N–Tree–Search (see 3.7)			**31.88** minutes	**4.81** minutes

Fig. 4: Measured times according to different modifications and methods

5 CONCLUSION

We conclude that image segmentation based on Kohonen Feature Maps is an excellent tool to realize pixel–oriented analysis of images. Unfortunately the implementation of the straightforward algorithm leads to enormous computation times. In order to make the image analysis acceptable for applications optimizations of the algorithm are required. The proposed modifications fulfill this requirement since our results demonstrate that a reduction to approx. 5% of the standard implementation could be reached.

The time consuming part of the Kohonen Feature Map reduced by our modifications is the determination of the weight vector next neighbored to the input–vector. So the proposed optimizations realize a fast and effective next neighbor search which can be directly transposed to other applications in the field of classification and computational geometry.

The presented results demonstrate that image analysis can be computed in a acceptable time even without parallelization on special and expensive hardware. Nevertheless a subset of the aspects in this paper can be applied on a vector architecture.

6 ACKNOWLEDGMENTS

The authors would like to thank the research division of the German Telekom for the financial support for this work within the project KAMEDIN. Furthermore many thanks to the GAF (Munich) and the Department of Biophysics and Medical Radiation Physics of the German Cancer Research Center (Heidelberg) for kindly providing the image data.

7 REFERENCES

[1] C. Busch, M. Gross: Interactive Neural Network Texture Analysis and Visualization for Surface Reconstruction in Medical Imaging. Computer Graphics forum, Vol.12, No.3,(EUROGRAPHICS'93), pp.C49–C60, (1993)

[2] M. Gross, R. Koch, L. Lippert, A. Dreger: Segmentierung und Klassifikation von Texturen mittels Wavelets und neuronalen Netzen, DAGM–Proceedings, to be published (1994)

[3] M. Gross, F. Seibert: Visualization of Multidimensional Data Sets using a Neural Network. The Visual Computer, Vol.10, No.3, pp.145–159, (1993)

[4] M. Gross, F. Seibert: Neural network image analysis for environmental protection. In Grützner (Edts.): Visualisierung von Umweltdaten 1991, GI, Berlin – Heidelberg – New York: Springer (1991)

[5] K.H. Höhne, M. Bomas, A. Pommert, M. Riemer, C. Schiers, U. Tiede, G. Wiebecke: 3D Visualization of Tomographic Volume Data using the Generalized Voxel Model, The Visual Computer, Vol.6, No.1, pp.28–36, (1990)

[6] R. Koch: Entwicklung eines 2D und 3D Texturanalysesystems basierend auf einer Merkmalextraktion mit Wavelets, Diplom–thesis, Computer Science Department, Technische Hochschule Darmstadt, (1994)

[7] T. Kohonen: The Self–Organizing Map. Proceedings of the IEEE, Vol. 78, No. 9, pp. 1464–1480, (1990)

[8] T. Kohonen: Self–Organization and Associative Memory, Berlin – Heidelberg – New York: Springer (1984)

[9] M. Levoy: Display of Surfaces from Volume Data, IEEE CG&A, Vol. 8, No. 5, pp. 29–37, (1988)

[10] W.E. Lorensen, H.E. Cline: Marching cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Vol.21, No.4, pp.163–169, (1987)

[11] G.M. Nielson: Visualization in the Scientific Discovery Loop, EUROGRAPHICS'93 tutorial notes, (1993)

[12] F. Preparata, M. Shamos: Computational Geometry. An Introduction, New York: Springer Publishing Company, (1985)

[13] F. Sauerbier, D. Scheppelmann, H.P. Meinzer: Segmentierung biologischer Objekte aus CT– und MR– Schnittserien ohne Vorwissen, DAGM–Proceedings, pp.289–293, Springer, (1989)