2000 15th International Conference on Pattern Recognition

(V.2) (V.B)

IEEE Computer Society Publications

The world-renowned IEEE Computer Society publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available from most retail outlets. Visit the Online Catalog, *http://computer.org*, for a list of products.

IEEE Computer Society Proceedings

The IEEE Computer Society also produces and actively promotes the proceedings of more than 141 acclaimed international conferences each year in multimedia formats that include hard and softcover books, CD-ROMs, videos, and on-line publications.

For information on the IEEE Computer Society proceedings, send e-mail to *cs.books@computer.org* or write to Proceedings, IEEE Computer Society, P.O. Box 3014, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1 714-821-8380. FAX +1 714-761-1784.

Additional information regarding the Computer Society, conferences and proceedings, CD-ROMs, videos, and books can also be accessed from our web site at *http://computer.org/cspress*

Revised 9 November 1999

ORGANISERS:

AERFAI
Asociación Española de Reconocimiento de Formas y Análisis de Imágenes

Centre de Visió per Computador

UAB
Universitat Autònoma de Barcelona

IRI
INSTITUT DE ROBÒTICA I INFORMÀTICA INDUSTRIAL

UPC

SPONSORS:

Generalitat de Catalunya
Departament de la Presidència
Comissionat per a Universitats i Recerca

MINISTERIO DE CIENCIA Y TECNOLOGÍA

FUNDACIÓ CATALANA PER A LA RECERCA

hp invent

Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number PR00750
IEEE Order Plan Catalog Number PR00750
ISBN 0-7695-0750-6
ISSN 1051-4651

Evolving Fuzzy Neural Network for Camera Operations Recognition

Irena Koprinska, Nikola Kasabov
Department of Information Science, University of Otago, Dunedin, New Zealand
e-mail: {ikoprinska, nkasabov}@infoscience.otago.ac.nz

Abstract

This paper reports an application of evolving fuzzy neural network (EFuNN) for camera operations recognition. EFuNN features one-pass learning, dynamical growing and shrinking architecture and ability to accommodate new knowledge without the need to retrain the network on both the original and new data. The network learns from pre-classified examples in the form of motion vector (MV) patterns, extracted from MPEG compressed video, in order to distinguish between six classes: static, panning, zooming, object motion, tracking and dissolve. The performance of EFuNN is compared with LVQ and the results are discussed. In addition, the impact of the number of membership functions and the contribution of the rule node aggregation are analyzed.

1. Introduction

Camera operations recognition is an important issue in content-based organization of video databases. At the stage of video parsing, it is critical to distinguish gradual shot transitions from the false positives due to camera operations since they both exhibit similar temporal variances. Detecting camera operations is also needed at the step of video indexing and retrieval. Since camera operations explicitly reflect how the attention of the viewer should be directed, the clues obtained are useful for index extraction and key frame selection.

With the widespread use of MPEG, methods for camera operations recognition that process directly the compressed stream were proposed. As camera operations exhibit specific patterns in the field of MVs, most of the approaches are based on MV patterns analysis. For example, Zhang et al. [5] compute measures based on the MV direction and then used thresholds to recognize pan/tilt and zoom. Manual tuning of the thresholds, however, is unlikely to be practical. Patel and Sethi [4] apply decision trees (DTs) built through a process of supervised learning to distinguish between zoom, pan, track, static, object motion and ambiguous. However, DTs delineate the concept by a set of axis-parallel hyperplanes which constrains their accuracy in realistic problems. On the other hand, as only MVs of P frames are used, the temporal resolution is low. In order to overcome these problems, a learning vector quantization (LVQ) [2] that learns from pre-classified MV patterns from both P and B frames was used in [3]. LVQ, though, like most of the static neural networks employing supervised learning, has predefined topology, requires multiple passes on the training set and suffers from "catastrophic forgetting", i.e. is not capable to accommodate new data without retraining on both the original and new data.

Overcoming these limitations is highly desirable in most practical applications, including camera operations recognition. Recently, evolving fuzzy neural networks (EFuNN) [1] were introduced as a possible solution. The goal of this paper is to study the potential of EFuNNs for camera operations recognition.

2. Evolving Fuzzy Neural Networks

EFuNNs are general purpose fuzzy neural networks that have a five-layer structure (Figure 1):

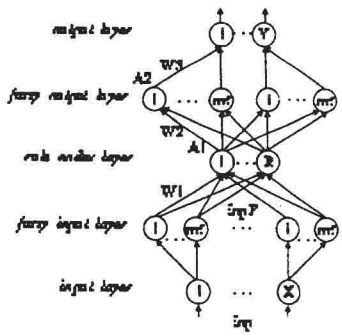


Figure 1. EFuNN's architecture

The *input* layer represents crisp input variables. The *fuzzy input* neurons stand for the fuzzy quantification of input variables using membership functions. The *rule nodes* evolve through learning and represent prototypes of data mapping between the fuzzy input and fuzzy output spaces. Each rule node is defined by two weight vectors: W1 and W2. The former is adjusted via unsupervised learning based on the similarity between the fuzzy input and the prototypes already stored. W2 is updated by LMS algorithm to minimize the fuzzy output error. The *fuzzy output* neurons stand for the fuzzy quantization of the output variables while the *output* nodes represent the real output values. There are several options for EFuNN growing [1]. We used the 1-of-n method, outlined below.

0-7695-0750-6/00 $10.00 © 2000 IEEE

2.1. EFuNN algorithm

Data pre-porocessing and EFuNN initialization:

1. Normalize current input vector (training example) inp_i in [0,1].
2. Fuzzify inp_i using triangular membership functions: $inpF_i = fuzzify(inp_i)$.
3. Set the defuzzifying weights W3 between the fuzzy outputs and real outputs as:

$W3_y = \frac{y}{mf-1}$, where $W3_y$ is the $W3$ vector for the class y, $y=1..Y$, Y is the number of classes, mf is the number of membership functions.

EFuNN's training:

1. Create the first rule node r_1 to represent the first example: $W1_1 = InpF_1$, $W2_1 = target_1$
2. While $(i<N)$ (i.e. there are training examples):
   $i=i+1$;
   For the i^{th} fuzzy training example $(InpF_i, target_i)$:
   a) calculate the *normalized fuzzy local distance D* between the fuzzy input vector $InpF_i$ and the already stored prototypes in the rule nodes r_j, $j=1..R$, where R is the current number of rule nodes:

$$D(InpF_i, r_j) = \frac{\sum_{j=1}^{R} \left| InpF_i - W1_{r_j} \right|}{\sum_{j=1}^{R} W1_{r_j}}$$

   b) calculate the activation $A1_{r_j}$ of the rule nodes r_j, $j=1..R$: $A1_{r_j} = 1 - \frac{D(InpF_i, r_j)}{2}$
   c) find the rule node r_{j^*} with highest activation $A1$
   d) if $A1_{r_{j^*}} < sThr$ then
      create a new rule node:
      $W1_i = InpF_i$, $W2_1 = target_1$; $j=j+1$;
   else
   - propagate the activation of r_{j^*} to the action neurons: $A2 = A1_{r_{j^*}} \cdot W2_{r_{j^*}}$
   - calculate the fuzzy output error: $Err = A2 - target_i$
   - find the action node k^* with highest $A2$
   - if $((k^* \neq t)$ or $(Err(k^*) > errThr))$ then
     create a new rule node:
     $W1_i = InpF_i$, $W2_1 = target_1$;
     $j=j+1$;
     else update the input and output connections of rule node k^*:

$$W1_{k^*}^{t+1} = W1_{k^*}^{t} + lr1 \cdot D(inpF_i, r_j)$$

$$W2_{k^*}^{t+1} = W2_{k^*}^{t} + lr2 \cdot Err_{k^*} \cdot A1_{j^*}$$

   e) if $(i = iAg)$ then aggregate:
   - for each rule node r_j, $j=1..R$ find the subset of rule nodes r_a, $a=1..A$, $A<R$ for which the *normalized Euclidian distances* $D_{euc}(W1_{r_j}, W1_{r_a})$, $D_{euc}(W2_{r_j}, W2_{r_a})$ are below the thresholds $w1Thr$, $w2Thr$, respectively:

$$D_{euc}(W1_{r_j}, W1_{r_a}) = \frac{\sqrt{\sum_{j=1}^{m} (W1_{r_j}^{m} - W1_{r_a}^{m})^2}}{\sqrt{m}} < w1Thr$$

$$D_{euc}(W2_{r_j}, W2_{r_a}) = \frac{\sqrt{\sum_{j=1}^{l} (W2_{r_j}^{l} - W2_{r_a}^{l})^2}}{\sqrt{l}} < w2Thr$$

   where m and l are the numbers of fuzzy input and fuzzy output nodes, respectively.
   - merge the nodes r_a and update $W1_{r_j}$, $W2_{r_j}$:

$$W1_{r_j} = \frac{\sum_{a=1}^{A} (W1_{r_a})}{A}, W2_{r_j} = \frac{\sum_{a=1}^{A} (W2_{r_a})}{A}$$

   - delete r_a, $a=1..A$; $j=j-1$;

Classification of new examples by EFuNN

An example that has not been seen during the learning, is first fuzzified and then propagated via EFuNN. The propagation from rule nodes to the output layer is restricted only for the winning rule node. The example is classified as an instance of the class y, where y is the index of the output neuron with the highest value.

To sum up, EFuNN is a dynamic architecture where the rule nodes grow if needed and shrink by aggregation. New rule units and connections can be added easily without disrupting existing nodes. In addition, EFuNN needs one-pass learning

3. Data Description

3.1. Classes

Following [3] we consider six classes: 1) *static:* stationary camera and little scene motion; 2) *panning:* camera rotation around its horizontal axis; 3) *zooming:* focal length change of a stationary camera; 4) *object motion:* stationary camera and large scene motion; 5) *tracking:* moving object being tracked by a camera and 6) *dissolve:* gradual transition between two sequences where the frames of the first one get dimmer and these of the second

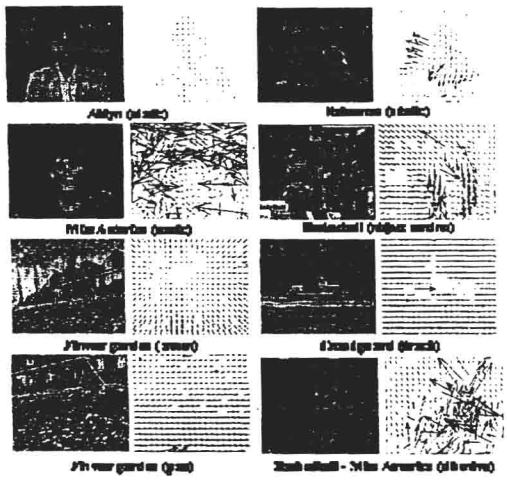


Figure 2. MV patterns corresponding to the 6 classes

one get brighter. While four of these classes are camera operations, object motion and dissolve are added as they introduce false positives. Each of the classes is characterized by a specific pattern in the field of MVs of P and B frames in a MPEG encoded sequence, Figure 2. The well-known benchmark sequences *Akiyo, Salesman, Miss America, Basketball, Football, Tennis, Flower Garden,* and *Coastguard* were been used in our experiments.

Several aspects of data complicate learning. As it can be seen from Figure 2, the three static examples have rather different MV fields. While the frames of *Akiyo* can be viewed as ideal static images, there are occasionally sharp movements in *Salesman.* The MV field of *Miss America* is completely different as the encoder we used [6] generates MVs with random orientation for the homogeneous background. Hence, it will be advantageous to use a classification system capable incrementally to adapt to new representatives of the static class without the need to retrain the network on the originally used data.

3.2. Feature extraction

Data pre-processing and feature extraction were done as in [3]. MVs of P and B frames are extracted and smoothed by a vector median filter. Based on them, a 22-dimentional feature vector is created for each frame. The first component is a measure for how static the frame is. It is calculated as the fraction of zero MVs using both the forward and backward MV components. The forward MV area is then sub-divided in 7 vertical strips, for which the average and standard deviation of MV directions and the average of MV magnitudes are computed.

In order to build the EFuNN classifier, the MV patterns of 1200 P and B frames (200 for each class), have been visually examined and manually labeled.

4. Experimental results and discussion

The goal of the experiments was fourfold: 1) to test the overall classification performance of EFuNN for camera operations recognition; 2) to analyze the individual classes detection; 3) to find how the different number of fuzzy membership functions influence the EFuNN performance; 4) to assess the contribution of rule nodes aggregation.

For the evaluation of the EFuNN classification results we used 10-fold cross validation. Apart from the various values for *mf* and *w1Thr*/*w2Thr* as discussed below, the EFuNN parameters were set as follows: *sThr*=0.92, *errThr*=0.08, *lr1*=0.05, *lr2*=0.01, *nAgg*=60.

Table 1 shows the classification accuracy of EFuNN with different number of membership functions when applied for video frames classification. The respective number of nodes are presented in Table 2. For the needs of comparison, Table 3 summarizes the results achieved by LVQ using the public domain package LVQPack [7].

Table 1. EFuNN classification accuracy [%] on the training and testing set (*w1Thr*=*w2Thr*=0.2)

mf	acc. [%] on training set	acc. [%] on testing set
2	85.8 ± 1.5	84.5 ± 2.4
3	91.4 ± 1.1	86.8 ± 4.5
4	95.5 ± 0.6	91.6 ± 2.9
5	95.5 ± 0.4	89.3 ± 4.3
6	95.2 ± 0.9	88.6 ± 4.5

Table 2. Number of nodes for the various EFuNN architectures (*mf*=2÷6)

	mf				
nodes	2	3	4	5	6
input	22	22	22	22	22
fuzzy inp.	44	66	88	110	132
rule	30±2	101.3±5.5	183.1±5.5	204.9±9.6	229.5±7.9
fuzzy out.	12	18	24	30	36
output	6	6	6	6	6
total	114	213	323	362	425

Table 3. LVQ performance

accuracy [%] on training set	accuracy [%] on testing set	nodes (input & codebook)	training epochs
85.4 ± 2.5	85.8 ± 2.2	60 (22 inp., 38 cod.)	1520

As it can be seen from Table 1, EFuNN achieves best classification accuracy when 4 membership functions are used. Further increase in their number almost does not affect the accuracy on the training set but results in worse accuracy on unseen examples due to overtraining. On the other hand, Table 2 indicates that increasing the number of the membership functions implies considerable growth in the number of rule nodes and, hence, the computational

complexity of the EFuNN's training algorithm. As a result, learning speed slows down significantly. However, depending on the specific application a suitable trade-off between the learning time and the accuracy can be found.

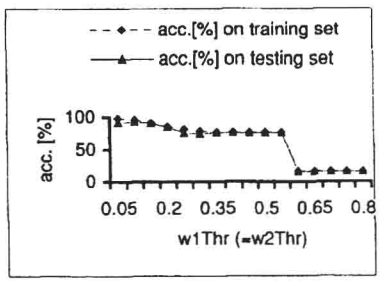
The performance of EFuNN compares favourably with LVQ in terms of classification accuracy (see Table 1,2 and Table 3). Another advantage of EFuNN is that it requires only 1 epoch for training in contrast to LVQ's multi pass learning algorithms that needs 1520 epochs in our case study. It should be noted, however, that the LVQ network is much smaller than the EFuNN architectures.

Table 4. EFuNN classification accuracy [%] of the individual video classes (mf=1÷6)

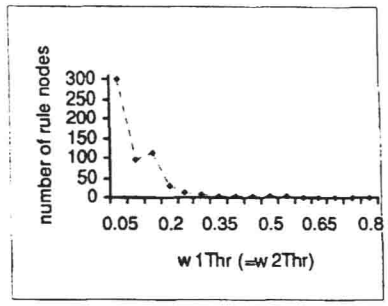
mf	zoom	pan	object motion.	static	tracking	dissolve
2	100	97.2±5.4	74.6±12.0	95.0±6.6	75.9±15.6	62.1±14.8
3	100	92.8±4.1	78.1±12.7	97.8±4.4	72.9±20.6	77.3±14.7
4	100	97.4±2.8	88.1±10.7	98.1±2.5	83.0±11.5	84.0±10.8
5	100	99.0±2.1	85.0±9.2	97.7±3.1	69.7±21.5	85.2±9.5
6	100	94.3±5.2	88.4±9.7	97.7±3.3	63.5±18.8	87.6±8.0

Table 4 summarizes the EFuNN classification of the individual classes. It was found that while zoom and pan are easily identified, the recognition of object movement, tracking and dissolve is more difficult. A detailed analysis indicates that the algorithm actually has difficulties to discriminate well between these three classes which also explains the large standard deviations. Despite the fact that the MV fields of *Miss America* were not typical for static videos and complicated learning, they are learned incrementally and classified correctly by EFuNN.

Figure 4a,4b show the impact of the aggregation on the classification accuracy and the number of rule nodes, respectively. Again, 10-fold cross validation was applied and each bullet represents the mean value for the ten runs. As expected, the results demonstrate that the aggregation is an important factor for good generalization and keeping the net's architecture at reasonable size. The best performance in terms of good trade-off between the recognition accuracy and net size was obtained for *w1Thr=w2Thr*= 0.15, 0.2. When the aggregation parameters are between 0.25 and 0.55, the accuracy on the testing set drops with about 10% as the number of rule nodes becomes insuffici-



a)



b)

Figure 4. Impact of the aggregation on the: a) classification accuracy and b) number of rule nodes *(mf=2)*

ent. Further increases in the values of the aggregation coefficients result in networks with one rule node which obviously can not be expected to generalize well.

5. Conclusions

An application of EFuNN for camera operations recognition was presented. EFuNN learns from examples in the form of motion vector patterns extracted from the MPEG-2 stream. The success of the neural net can be summarized as high classification accuracy and fast training. Future work will focus on the possibility to further improve the performance combining image information with audio features and text captions. The approach will be also extended to incrementally accommodate new classes, e.g. other common types of camera operations.

Acknowledgements

The work was supported by the *New Zealand Foundation for Research, Science and Technology*, UOO-808. The EFuNN implementation is a part of RICBIS and is available at http://divcom.otago.ac.nz/infosci/kel/CBIIS.html.

6. References

[1] N. Kasabov, "Evolving connectionist and fuzzy connectionist systems – theory and applications for adaptive, on-line intelligent systems", *In: Neuro-Fuzzy Techniques for Intel. Inf. Processing*, Heidelberg, Physica Verlag, 1999, pp. 111-114.

[2] Kohonen, "The Self-Organizing Map", *in Proc. of the IEEE*, v.78(9), 1990, pp. 1464-1480.

[3] I. Koprinska and S. Carrato, "Video Segmentation of MPEG Compressed Data", *in Proc. ICECS'98*, 1998, pp. 243-246.

[4] N.V. Patel and I.K. Sethi, "Video Shot Detection and Characterization for Video Databases", *Pattern Recognition*, v. 30, 1997, pp. 583-592.

[5] H. Zhang, C.Y. Low and S.W. Smoliar, "Video Parsing and Browsing Using Compressed Data", *Multimedia Tools & Applications*, v.1, 1995, pp. 89-111.

[6] http://www.mpeg.org/MPEG/MSSG/VMPEG

[7] http://nucleus.hut.fi/nnrc

Adaptive Combination of Classifiers and its Application to Handwritten Chinese Character Recognition

B.H. Xiao, C.H. Wang and R.W. Dai
Institute of Automation, Chinese Academy of Sciences
dai@ht.rol.cn.net

Abstract

Motivated by the idea of metasynthesis, a new adaptive classifier combination approach is proposed in this paper. Compared with previous integration methods, parameters of the proposed combination approach are dynamically acquired by a coefficient predictor based on neural network and vary with the input pattern. It is also shown that many existing integration schemes can be considered as special cases of the proposed method. This approach is tested in application on handwritten Chinese character recognition. The experimental results demonstrate that this method can result in substantial improvement in overall performance.

1. Introduction

Recently, there is a popular belief that features and classifiers of different types could complement each other to some extent, therefore fine combination of multiple classifiers would result in improvement in performance [2-5]. However, how to effectively combine multiple classifiers remains an unsolved problem.

In general, classifier combination strategies can be divided into two main categories according to the levels of information produced by various classifiers, i.e. abstract-level or measurement-level information.

Abstract-level information includes class labels, rankings of classes, etc. The abstract-level information is usually sufficient for problems with a small number of classes. But for large class set problems such as handwritten Chinese character recognition, use of only abstract-level information could lead to many conflicting decisions which are difficult to handle.

Measurement-level information includes estimates of posteriori probabilities, similarities or distances to prototypes, fuzzy membership values, etc. If these outputs are supplied, a sum rule and some other linear combination have been suggested [4,5]. The performance of combination is, of course, dependent upon the selected coefficient vector. The question is then how to establish the appropriate coefficient vector.

In this paper, we focus on classifier combination in the second category, as the measurement-level outputs are more discriminative than the abstract-level outputs for large class set problems. Motivated by the idea of metasynthesis [1], we develop a common adaptive combination framework for this kind of classifiers. Compared with previous integration methods, parameters of this combination approach are dynamically acquired by a coefficient predictor based on supervised learning neural network (NN) and vary with the input pattern. That is to say, our goal is no longer to find an optimal coefficient vector, but to find an optimal coefficient predictor. It is also shown that under different assumption and using different approximations, many existing combination schemes can be derived, such as majority voting, weighted voting, dynamic classifier selection and classifier fusion based on linear static model. In a handwritten Chinese character recognition (HCCR) experiment where three classifiers were used to discriminate between 3755 classes, the results of comparative study demonstrate the surprising effectiveness of this adaptive classifier combination approach.

This paper is organized as follows. In Section 2, the adaptive classifier combination scheme is formulated and some commonly used combination strategies are also derived. Section 3 gives three individual classifiers for handwritten Chinese character recognition, and expatiates on the implementation of the adaptive combination. Experimental results are compared in Section 4. Finally, Section 5 offers discussion and concluding remarks.

2. Theoretical framework of adaptive classifier combination

2.1. Individual classifiers

Consider a pattern recognition problem where an input pattern X is to be assigned to one of m possible classes. Let S represents the m possible classes, and C represents the n available classifiers.

$$S = \{S_i\}_{i=1}^{m} \text{ and } C = \{C_j\}_{j=1}^{n}; \quad (1)$$

0-7695-0750-6/00 $10.00 © 2000 IEEE

such as images, is not obvious. Much of the literature regarding fractal analysis has been concerned with the estimation of fractal dimension, given a discrete data set. An important aspect of a fractal object is its fractal dimension. In order to understand the concept of a frcatal dimension, we will introduce the *Hausdorff dimension*.

Hausdorff dimension is the dimension singled out by Mandelbrot when he defined "fractal". It is perhaps a bit more difficult to define than some of the other kinds of dimension that have been (and will be) considered. There exist a variety of fractal dimensions. Among them, Hausdorff dimension is the most useful of the fractal dimension because it is suitable for any sets and based on a mathematical tool — *measure theory*, which makes analysis easy.

For any set F and $\delta < 1$, $H^s_\delta(F)$ is a non-increasing function of s. It can be shown that $H^s(F)$ is also a non-increasing function of s. In fact, the stronger conclusion is that if $t > 0$ and $\{U_i\}$ is a δ-cover of F, we have

$$H^t_\delta(F) \leq \sum_i |U_i|^t \leq \delta^{t-s} \sum_i |U_i|^s. \qquad (1)$$

We take the infimum, that is

$$H^t_\delta(F) \leq \delta^{t-s} H^s_\delta(F).$$

Definition 1 Let $\delta \to 0$, if $H^s(F) < \infty$, then $H^t(F) = 0$ for $s < t$. Therefore, there exists a critical value of s, such that $H^s(F)$ jumps from ∞ to 0 at this point. This critical value is called the *Hausdorff Dimension* of F, and it is symbolized by $dim_H F$.

Formally, we have

$$dim_H F = \inf\{s : H^s(F) = 0\} = \sup\{s : H^s(F) = \infty\}, \qquad (2)$$

and

$$H^s(F) = \begin{cases} \infty & \text{if } s < dim_H F \\ 0 & \text{if } s > dim_H F \end{cases}$$

If $s = dim_H F$, probably $H^s(F)$ is 0 or ∞, or may satisfy

$$0 < H^s(F) < \infty.$$

A Borel set is called an *s-set* if the latter condition as shown in the above is satisfied.

3. Algorithms for Estimating the Fractal Dimensions

One of the basic characteristics of a fractal is its dimension. Estimates of the dimension tend to be very inaccurate as data size is reduced. There is a measurement of the complexity of a geometric object, which we will commonly call the "fractal dimension," although there are several varying concepts and terms, the big brother of which is the Hausdorff-Besicovitch dimension.

3.1 Computing 1-D Divider Dimension

Box computing dimension is one of the most widely used dimensions. Its popularity is largely due to its relative ease of mathematical calculation and empirical estimation.

Let F be a non-empty and bounded subset of $\mathbb{R}^n$, $\xi = \{\omega_i : i = 1, 2, 3, ...\}$ be covers of the set F. $N_\delta(F)$ denotes the number of covers, such that

$$N_\delta(F) = |\xi : d_i \leq \delta|,$$

where d_i stands for the diameter of the i-th cover. This equation means that $N_\delta(F)$ is the smallest number of subsets (with length δ) which covers the set F, and their diameters d_i's are not greater than δ.

The upper and lower bounds of the box computing dimension of F can be defined by the following formulas:

$$\underline{\dim_B} F = \liminf_{\delta \to 0} \frac{\log_2 N_\delta(F)}{-\log_2 \delta}, \qquad (3)$$

$$\overline{\dim_B} F = \overline{\lim_{\delta \to 0}} \frac{\log_2 N_\delta(F)}{-\log_2 \delta}, \qquad (4)$$

where the overline stands for the upper bound of dimension while the underline for lower bound.

Definition 2 If both the upper bound $\overline{\dim_B} F$ and the lower bound $\underline{\dim_B} F$ are equal, i.e.

$$\liminf_{\delta \to 0} \frac{\log_2 N_\delta(F)}{-\log_2 \delta} = \overline{\lim_{\delta \to 0}} \frac{\log_2 N_\delta(F)}{-\log_2 \delta},$$

the common value is called *box computing dimension* or *box dimension* of F, namely:

$$\dim_B F = \lim_{\delta \to 0} \frac{\log_2 N_\delta(F)}{-\log_2 \delta}. \qquad (5)$$

Further discussions on fractal theory can be found in [3]. The modified box computing dimension method gives a very good estimate of fractal dimension. It can be easily shown that computation complexity of other approaches, including the original box counting dimension, is much higher than that of this approach. Thus it has the advantages of simplicity in computation and improvement in efficiency.

3.2 Estimating 2-D Fractal Signature

To compute the fractal dimension, we need to measure the area of the gray level surface. The idea of the blanket technique is based on the equivalent definition of the box computing dimension. In the blanket technique, all points of the 3-D space at distance δ from the gray level surface are considered. For example, the image is represented by a gray-level function $g(i, j)$. The covering blanket is defined

by its upper surface $u_\delta(i,j)$ and its lower surface $b_\delta(i,j)$. Initially, $\delta = 0$ and given the gray-level function equals the upper and lower surfaces, namely:

$$g(i,j) = u_0(i,j) = b_0(i,j).$$

For $\delta = 1, 2, ...,$ the blanket surfaces are defined as follows:

$$u_\delta(i,j) = \max\left\{u_{\delta-1}(i,j)+1, \max_{|(m,n)-(i,j)|\le 1} u_{\delta-1}(m,n)\right\} \quad (6)$$

$$b_\delta(i,j) = \min\left\{b_{\delta-1}(i,j)-1, \min_{|(m,n)-(i,j)|\le 1} b_{\delta-1}(m,n)\right\} \quad (7)$$

A point $f(x,y)$ will be included in the blanket for δ when $b_\delta(x,y) < f(x,y) < u_\delta(x,y)$. The blanket definition uses the fact that the blanket of the surface for radius δ includes all the points of the blanket for radius $\delta-1$, together with all the points within radius 1 from the surfaces of that blanket. Eq. (7) ensures that the new upper surface u_δ is higher than $u_{\delta-1}$ by at least 1, and also at a distance of at least 1 from $u_{\delta-1}$ in the horizontal and vertical directions.

The volume Vol_δ of the blanket is computed from u_δ and b_δ:

$$Vol_\delta = \sum_{i,j}(u_\delta(i,j) - b_\delta(i,j)). \quad (8)$$

In this subsection, computing fractal dimension by measuring surface area will be presented. The basic idea is that a 2-D pattern can be mapped onto a gray-level function. Furthermore, this function can be mapped onto a surface which can be used to approximate its fractal dimension.

As the volume Vol_δ of the blanket is measured with radius δ, the area of a fractal surface can be deduced, which is called *fractal signature (FS)*

$$A_\delta = \frac{Vol_\delta}{2\delta}, \quad (9)$$

or

$$A_\delta = \frac{Vol_\delta - Vol_{\delta-1}}{2}. \quad (10)$$

According to the definition of *Minkowski Dimension* and [4], the area of a fractal surface is:

$$A(\delta) \approx \beta\delta^{2-D}, \quad \delta = 1, 2, ...,$$

from which the fractal dimension D can be computed. Since the dimension can be regarded as a slope on a log-log scale, only two points are needed to get the dimension. We use two values of δ to compute the fractal dimension, namely, we take $\delta = \delta_1$ and δ_2, then

$$A_{\delta_1} \approx \beta\delta_1^{2-D}, \quad A_{\delta_2} \approx \beta\delta_2^{2-D}. \quad (11)$$

From Eq. (11), we have

$$\frac{A_{\delta_1}}{A_{\delta_2}} \approx \frac{\delta_1^{2-D}}{\delta_2^{2-D}}.$$

Taking the logarithm of both sides yields:

$$2 - D \approx \frac{\log_2 A_{\delta_1} - \log_2 A_{\delta_2}}{\log_2 \delta_1 - \log_2 \delta_2},$$

$$D \approx 2 - \frac{\log_2 A_{\delta_1} - \log_2 A_{\delta_2}}{\log_2 \delta_1 - \log_2 \delta_2}. \quad (12)$$

Thus, the fractal dimension D has been computed.

4. Application

Two applications of the fractal theory are presented in this section, namely: (1) the fractal technique is used to extract the features for 2-D objects; (2) the fractal signature is employed to identify different scripts.

4.1 Applying Fractal Technique to Feature Extraction

Pattern recognition requires the extraction of features from the regions of an image, and the processing of these features with a pattern classification technique. In particular, this approach reduces the dimensionality of a 2-D pattern by way of a central projection method, and thereafter, performs Daubechies' wavelet transform on the derived 2-D pattern to generate a set of wavelet transformation subpatterns, namely, curves that are non-self-intersecting. Further from the resulting non-self-intersecting curve, the divider dimensions are readily computed. These divider dimensions constitute a new feature vector for the original 2-D pattern, defined using the curves' fractal dimensions, see in Figure 1.

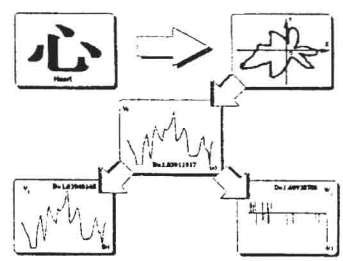


Figure 1. The feature extraction by wavelet sub-patterns and divider dimensions for the Chinese character "Heart"

The recognition rates obtained in the 3000 Chinese characters using our new approach is almost 99.8%, and the traditional method of extraction from the HV-projections is only 64%. The fractal technique is significantly faster than the traditional method based on Fourier transformation.

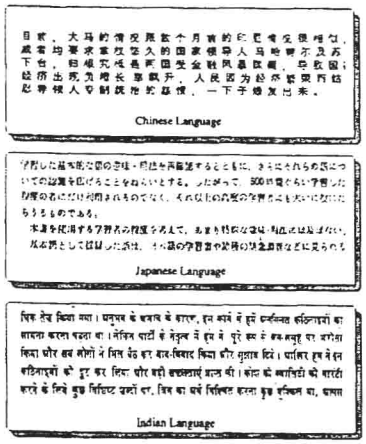


Figure 2. Samples of different oriental languages

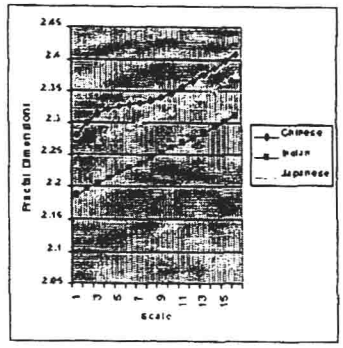


Figure 3. An example of oriental languages classification

4.2 Classification of Oriental Language Using Fractal Signature

The fractal method is applied to classify the different languages, such as shown in Figure 2, with three samples. The basic idea of the experiment is that an image of a document printed in a kind of language can be mapped onto a gray-level function. Furthermore, this function can be mapped on to a surface which can be used to approximate its fractal dimension. In order to extract the structure information of a document, *fractal signature* will be used. From the definition of the fractal signature, i.e. Eq. (9), it is clear that the fractal signature is completely determined by the area of the surface. The surface is a mapping of the gray-level function which represents a document image of oriental language. Consequently, the fractal signature reflects certain characteristics of the document image of oriental language. We can obtain the results of classification of three scripts from the fractal signature trend in Figure 3. These preliminary results indicate that discrimination based on the fractal signature of document images may well represent a viable approach to utilizing computers to assist in multifarious language classification.

5. Conclusions

In this work, we presented the results which aimed at showing that, within the field of image analysis, it is possible to use fractal feature based on the estimating fractal dimension to extract information that is relevant in object recognition tasks. The experiment results show that this approach allows us to obtain new and interesting descriptions of complex patterns. In some situations, it has even already yielded better results than "classical" methods. For all the cases, differences in fractal dimension can yield the significative values.

References

[1] Matteo Baldoni, Cristina Baroglio, Dacide Cavagnino, and Lorenza Saitta. "Towards automatic fractal feature extraction for image recognition," in *Feature extraction, construction and selection: a data mining perspective*, (Huan Liu and Hiroshi Motoda, editors), pp. 356-373, Kluwer Academic Publishers, 1998.

[2] B. B. Chaudhuri and Nirupam Sarkar. "Texture segmentation using fractal dimension," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 17, pp.72-77, 1995.

[3] Gerald A. Edgar. *Measure, Topology and Fractal Geometry*, New York: Springer-Verlag, 1990.

[4] S. Peleg, J. Naor, R. Hartley, and D. Avnir. "Multiple resolution texture analysis and classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 4, pp. 518-523, July 1984.

[5] Antoine Saucier and Jiri Muller. "Multifractal approach to textural analysis," in *Fractals and Beyond Complexities in the Sciences*, World Scientific Publishing Company, pp. 161-171, 1998.

[6] Y. Y. Tang, Hong Ma, Jiming Liu, Bingfa Li, and Dihua Xi. "Multiresolution analysis in extraction of reference lines documents with gray-level backgroung," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19, pp. 921-926, 1997.

Invariant Grey-scale Features for 3D Sensor-data

Marc Schael and Sven Siggelkow
Institute for Pattern Recognition and Image Processing
Computer Science Department
University of Freiburg, D-79110 Freiburg i. Br., Germany
{schael,siggelkow}@informatik.uni-freiburg.de

Abstract

In this paper a technique for the construction of invariant features of 3D sensor-data is proposed. Invariant grey-scale features are characteristics of grey-scale sensor-data which remain constant if the sensor-data is transformed according to the action of a transformation group. The proposed features are capable of recognizing 3D objects independent of their orientation and position, which can be used e. g. in medical image analysis. The computation of the proposed invariants needs no preprocessing like filtering, segmentation, or registration. After the introduction of the general theory for the construction of invariant features for 3D sensor-data, the paper focuses on the special case of 3D Euclidean motion which is typical for rigid 3D objects. Due to the fact that we use functions of local support the calculated invariants are also robust with respect to independent Euclidean motion, articulated objects, and even topological deformations. The complexity of the method is linear in the data-set size which may be to high for large 3D objects. Therefore approaches for the acceleration of the computation are given. First experimental results for artificial 3D objects are presented in the paper to demonstrate the invariant properties of the proposed features.

1. Introduction

In many areas of research 3D datasets become increasingly important [5]. All applications which use 3D sensor-data (acquired by sensors which scan objects of the real world, e. g. medical imaging or process tomography) must cope with undesirable transformations. These transformations result from the different properties of the sensor or the scanning method. All these transformations act as geometrical and/or grey-scale based transformation on the data. To apply methods of digital image processing and pattern recognition it is helpful to construct features which are invariant with respect to the transformations mentioned above.

Different approaches for the construction of invariant features have been developed during the last decade. They can be divided into the following three main categories [2]:

Normalization methods extract salient features of an object (like center point, main axes) and normalize the object with respect to these. Their robustness, however, is limited by the quality of determining the salient features.

Differential approaches are based on Lie groups. Invariant features must be insensitive to infinitesimal variations of the parameters of the transformation group. Therefore invariants can be constructed by solving differential equations that have been obtained by setting partial derivatives with respect to the transformation parameters to zero. However, solving the partial differential equations often is a quite complex task.

Integral approach: The equivalence class of an object forms an orbit in object space. The idea is to average arbitrary functions evaluated on the orbit (Haar integrals) [6, 9, 12]. It is clear that the integral over the entire orbit is invariant to the transformation group.

A second categorization of methods can be done into *grey-scale based approaches* and *geometry based approaches*:

Grey-scale based approaches use the full sensor information for describing objects and deriving invariant features. An example for a grey-scale based method for the recognition of 3D objects are moments [3].

Geometry based approaches reduce an object's representation to its geometrical primitives (points, lines, ellipses etc.) instead, thus neglecting the object's texture information. An overview of different geometry based approaches for the recognition of 3D objects can be found in [8, 4].

In this paper an *integral, grey-scale based approach* for the construction of invariant features for the recognition of 3D objects is proposed. It is based on an averaging operation over the Euclidean transformation group [13]. We would like to point out that the proposed method is not restricted to grey-scale objects but can be applied to other sensor-data like color, multi-band etc. as well.

0-7695-0750-6/00 $10.00 © 2000 IEEE

The remainder of this paper is organized as follows: First we introduce the terminology used in this paper. In section 3 we then propose our method for the construction of invariant 3D grey-scale features. This is treated not only theoretically but we also discuss practical aspects like the efficient implementation. Section 4 then presents first results applying the method to artificial 3D data. Finally we conclude our paper in section 5.

2. Terminology

We model a 3D dataset as a mapping $\mathbf{M} : \mathcal{P}^3 \subset I\!N^3 \mapsto [0,\ldots,V], \mathbf{x} \mapsto \mathbf{M}[\mathbf{x}]$, with $\mathbf{x} = (x,y,z)^T$, and $x,y,z \in [0,...,N-1]$, $V \in I\!R$, $N \in I\!N$. Possible 3D datasets can be 3D volume images, 3D geometric data or 3D depth images [15]. The action of geometrical transformations on the 3D sensor-data may be described by the action of a transformation group G. If $\widetilde{\mathbf{M}}$ denotes the transformed dataset of $\mathbf{M}$ we can write $\widetilde{\mathbf{M}} = g\mathbf{M}$, $g \in G$. This means we have to transform the entire dataset $\mathbf{M}$ by the group element g. Alternatively we can compute the coordinates $\widetilde{\mathbf{x}}$ of the transformed pattern $g\mathbf{M}$ with $\widetilde{\mathbf{x}} = g^{-1}\mathbf{x}$. We can write $\widetilde{\mathbf{M}}[\mathbf{x}] = \mathbf{M}[\widetilde{\mathbf{x}}]$. To put it differently, either we transform the whole dataset $\mathbf{M}$ by g or we transform the coordinate $\mathbf{x}$ by the inverse group element g^{-1}. Two patterns $\widetilde{\mathbf{M}}$ and $\mathbf{M}$ are called equivalent, $\widetilde{\mathbf{M}} \overset{G}{\sim} \mathbf{M}$, if one pattern is the result of the action of an element g of the transformation group G on the other, i.e. $\widetilde{\mathbf{M}} \overset{G}{\sim} \mathbf{M} \Leftrightarrow \exists g \in G : \widetilde{\mathbf{M}} = g\mathbf{M}$. In this paper the proposed features map the pattern space to $I\!R$. A feature F is invariant with respect to the action of a transformation group G if $F(g\mathbf{M}) = F(\mathbf{M})\, \forall g \in G$.

3. Constructing invariant features for 3D

In [13] a method for the construction of invariant features for 2D grey-scale images by averaging over the transformation group is presented. Averaging over a group G can be written as

$$A[f](\mathbf{M}) := \frac{1}{|G|} \int_G f(g\mathbf{M})\, dg, \qquad (1)$$

where the fraction in front of the integral is used to normalize the averaging result by the volume of the group G. Equation (1) is also called invariant integration. For compact and finite groups we define the volume $|\cdot|$ of a group G as $|G| := \int_G dg$. It is evident that the result is invariant to any transformation of $g \in G$. The existence of a complete feature set can be shown for compact and finite groups [12]. In this paper we extend this approach to 3D and develop a method for the construction of invariant features for 3D sensor-data.

To evaluate (1) one has to choose a parameterization $\boldsymbol{\xi}$ of the group G. Since the parameterization can be arbitrarily chosen, the results of the integrations over the same pattern $\mathbf{M}$ with different parameterizations should be equal. This is in general not the case. Each parameterization has its own distribution in parameter space. To make the integral independent of the chosen parameterization, one has to determine a measure $\rho(\boldsymbol{\xi})$ which weights the volume elements dg of the integral. The measure $\rho(\boldsymbol{\xi})$ is called invariant measure and guarantees the independence of the integral from the parameterization used.

Due to the fact that we want to construct invariant features for the rotation and translation in $I\!R^3$ (Euclidean motion) we will now focus on the Euclidean transformation group. The Euclidean group can be defined by the rotation group $SO(3)$ and the translation group G_T as the Cartesian product $G_E := SO(3) \times G_T$. For both groups we must determine the invariant measure to derive a parameterized invariant integration formula. The translation group G_T can be parameterized by $(t_x,t_y,t_z)^T$, which defines the translation vector $\mathbf{t}$. To obtain a finite translation group G_T all translation parameters are understood modulo N. For the translation group G_T the invariant measure amounts to unity. Now we have to choose a suitable parameterization for the rotation group $SO(3)$. By using Cayley-Klein parameters [7] and the rotation angle and axis parameterization (Ω,Θ,φ) for $SO(3)$ the invariant measure can be calculated as shown in [7] to $\rho(\Omega,\Theta,\varphi) = 0.5\sin^2(0.5\Omega)\sin(\Theta)\, d\varphi\, d\Theta\, d\Omega$. The group volume of the Euclidean group G_E for the selected parameterization (Ω,Θ,φ) can be determined to $|G_E| = \pi^2 N^3$. Remember that the invariant measure of the translation group amounts to unity, therefore the volume of the translation group amounts to N^3 (the number of voxels of the 3D dataset). The final formula for the invariant integration over the Euclidean group G_E with rotation angle and axis parameterization $\boldsymbol{\xi} = (t_x,t_y,t_z,\Omega,\Theta,\varphi)$ can be given now as:

$$A[f](\mathbf{M}) = \alpha \int_{G_E} f\left(g_{\boldsymbol{\xi}}\mathbf{M}\right) \sin^2(\tfrac{1}{2}\Omega)\sin(\Theta)\, d\boldsymbol{\xi}, \qquad (2)$$

where α is a constant factor defined as $\alpha := (2\pi^2 N^3)^{-1}$.

3.1. Efficient evaluation

The interpretation of the invariant integration in equation (1) shows that the straightforward calculation for more complex functions of f requires high computation power. For each group element g the pattern $\mathbf{M}$ is transformed and a function f is calculated from the resulting pattern $g\mathbf{M}$. The result of f must be weighted with the invariant measure which depends on the chosen parameterization. Finally the total integral is determined from all results of f. The sum is normalized by the group volume $|G_E|$.

Using the rotation axis and angle parameterization the action of the group element $g_{\boldsymbol{\xi}}$ can be formulated as

$$g_{\boldsymbol{\xi}}\mathbf{M} = \mathbf{M}\left[\mathbf{R}_{\Omega\Theta\varphi}^{-1}\mathbf{x} - \mathbf{t}\right]. \tag{3}$$

Examination of this equation reveals the following: Assume a constant vector of translation $\mathbf{t}$. We know that the rotation matrix $\mathbf{R}$ is an element of $SO(3)$, so the transformed point $\widehat{\mathbf{x}} = \mathbf{R}_{\Omega\Theta\varphi}^{-1}\mathbf{x}$ moves around $-\mathbf{t}$ on a sphere with radius $|\mathbf{x} - \mathbf{t}|$ if we vary the parameters $(\Omega, \Theta, \varphi)$.

What happens if we introduce the function f. Let us select monomials $f(\mathbf{M})=\prod_{i=1}^{d} a_i\mathbf{M}(\mathbf{x}_i)^{p_i}$. We define the corresponding radius r_i by $r_i = |\mathbf{x}_i - \mathbf{t}|$. Inserting equation (3) into the monomial f results in:

$$f(g_{\boldsymbol{\xi}}\mathbf{M}) = \prod_{i=1}^{d} a_i\mathbf{M}\left[\mathbf{R}_{\Omega\Theta\varphi}^{-1}\mathbf{x}_i - \mathbf{t}\right]^{p_i}, \tag{4}$$

where d denotes the number of product terms in the monomial f. The radii r_i define the sizes of the spheres on which the points $\widehat{\mathbf{x}}_i$ move. Evaluating the expression (4) for a given $(\Omega, \Theta, \varphi)$ reveals the following: We have to build the product of d terms. For each term we must calculate the power p_i of the grey-scale value $\mathbf{M}$ at the position $\widehat{\mathbf{x}}_i - \mathbf{t}$ multiplied with a_i. This is a local operation in the neighborhood of point $\mathbf{t}$. Since we have to integrate over all translations $\mathbf{t}$ the local computation must be done for all elements of the pattern $\mathbf{M}$. This interpretation leads to the following two step strategy for efficient evaluation of formula (2). First, for every point of the 3D pattern a local function f is evaluated. In the second step the total integral of all local results is calculated.

3.2. Sampling of the spheres

We now discuss the problem of sampling for an implementation of the method. To ensure a sampling of uniform density on the sphere we use the following approach: In the case of 2D the sampling of the circle with radius R can be defined over a maximum arc length between two adjacent sample points. The maximum arc length defines an upper limit for the sampling. For radius $R = 1$ it should be sufficient to choose a maximum arc length of at least $s_{\max} = \pi/4$. Based on the arc length we can determine the offset angle $\Delta\Theta$ between two adjacent sample points: $\Delta\Theta(R) = \pi/\lceil 4R \rceil$. Sample points which do not lie on the grid must be interpolated, e. g. by trilinear interpolation. For a given angle Θ the radius of the equator circle is calculated as $R_\Phi = R\sin(\Theta)$. Evaluating the same condition with the arc length for the offset angle $\Delta\Phi$ leads to $\Delta\Theta(R) = \pi/\lceil 4R\sin(\Theta) \rceil$. The radius R is defined by the maximum radius of the spheres which must be sampled. On the right side of figure 1 a correct sampling of a sphere is shown. Our proposed method for the sampling of a sphere is only an approximation. To increase the quality of the sampling it might be necessary to decrease the maximum arc length $s_{\max}$. A detailed derivation of the algorithm for the computation of 3D invariants can be found in [11].

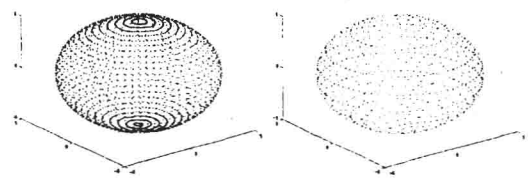


Figure 1. On the left side an incorrect sampling method is shown, the right side shows a correct sampling.

3.3. Computational Complexity

As mentioned above the algorithm consists of a two step strategy. For each voxel of the 3D volume data we have to evaluate a kernel function f. As we use functions of local support the calculation time is independent from the data-set size. After all local evaluations we have to sum all local results. Thus, the computational complexity is $\mathcal{O}(N^3)$ which is linear in the data-set size. For large data-set sizes this results in long processing time. We give two possibilities for acceleration of the method: parallel/distributed processing and stochastic sampling.

As the major amount of calculation time is needed within local computations the method can be easily implemented on parallel or distributed hardware without high communication overhead. E. g. in [1] the parallelization of the algorithm is described for 2D grey-scale features.

Another possibility for high speedup is to estimate the features by a Monte-Carlo method instead of calculating them deterministically [14]. The basic idea is not to evaluate f on all samples of figure 1 but to evaluate f only on n random uniformly distributed samples: $A[f](\mathbf{M}) \approx \frac{1}{n}\sum_i f(g_{\boldsymbol{\xi}_i}\mathbf{M})$,with random $\boldsymbol{\xi}$. Thus, one obtains an error. This, however, can be estimated with the following Gaussian distribution accuracy estimation formula:

$$\Phi\left(\frac{\epsilon\sqrt{n}}{\sqrt{V(f(g_{\boldsymbol{\xi}_i}\mathbf{M}))}}\right) \geq 1 - \frac{\delta}{2}, \tag{5}$$

with ϵ being the error bound, δ being the probability of exceeding this error, $V(.)$ being the variance and $\Phi(.)$ being the integrated standard normal distribution. To give a concrete example: Set $\epsilon = 0.01$ and $\delta = 5\%$. Choosing kernel

functions $f(g\mathbf{M}) \in [0,1]$ we obtain

$$V(f(g\mathbf{M})) = \underbrace{E(f^2(g\mathbf{M}))}_{\leq 1} - \underbrace{E(f(g\mathbf{M}))^2}_{\geq 0} \leq 1 \quad (6)$$

and therefore $n \geq 38416$, i.e. constant complexity. This, however, requires that the application allows for some uncertainty in the features, e. g. when the class distance is big compared to ϵ.

4 Experiments

First experiments with artificial 3D objects were done to show the invariant properties of the proposed features. The results are based on [11]. We have created binary volume images with a cuboid, a sphere and a pyramid as objects. The dimensions of the volume images were $64 \times 64 \times 64$. Voxels belonging to the objects were set to value 255. The remaining voxels which surround the objects were set to zero. All objects have the same object volume. The mean grey-scale value[1], which defines a trivial invariant feature with respect to the Euclidean motion, is not able to discriminate between these objects. Three more objects were defined by cutting each object into two parts. Thus, the total number of classes which were defined for the experiments was six. In figure 2 rendered images of the unseparated objects are shown. Rendered images of the separated volume objects are shown in figure 3.

Figure 2. Unseparated artificial 3D objects used in the experiments.

Figure 3. Separated artificial 3D objects used in the experiments.

Each of the reference objects was transformed by a translation or rotation. The parameters of the Euclidean transformations are denoted by $\kappa = 1,...,5$, where $g_\kappa = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)$. The applied transformations were $g_1 = (0,0,0,0,0,0)$, $g_2 = (0,0,0,10,5,7)$, $g_3 = (0,0,\frac{\pi}{2},0,0,0)$, $g_4 = (0,-\frac{\pi}{2},0,0,0,0)$, and $g_5 = (\frac{\pi}{2},0,0,0,0,0)$. We have defined the following classes corresponding to $c = 1...6$: cuboid, sphere, pyramid, separated cuboid, separated sphere, and separated pyramid. Including the reference objects we generated five objects per class. Thus, the total number of objects used was 30. For each object we calculated 40 invariant features with monomials of second and third degree. Nearly all invariant features were able to discriminate all classes individually. Table 1 presents the results for a weighted Euclidean distance of all features. The class "cuboid" can be discriminated well from the other classes. The distance between the class "separated sphere" and "separated cuboid" is smaller by several orders of magnitude. The classes with the greatest distance between each other are the class "sphere" and the class "separated sphere".

c	1	2	3	4	5	6
1	0.0	40990.1	9373.5	42632.7	7491.0	27545.7
2	40990.1	0.0	54226.8	160.6	32418.1	10100.7
3	9373.5	54226.8	0.0	57469.5	17268.6	37640.6
4	42632.7	160.6	57469.5	0.0	33495.4	10237.0
5	7491.0	32418.1	17268.6	33495.4	0.0	20153.8
6	27545.7	10100.7	37640.6	10237.0	20153.8	0.0

Table 1. Weighted Euclidean distances between the mean values of the features of the object classes c.

Further experiments were made in [10] to analyze the robustness of the proposed invariant features: the volume images including the surrounding background were disturbed by additive Gaussian noise of different signal-to-noise ratios. Different classes could still be discriminated for reasonable SNR. In first experiments we used real MR images to show the functionality of the method [10].

5 Conclusion

In this paper we proposed a technique for the construction of invariant grey-scale features for 3D sensor-data. The proposed features are capable of recognizing 3D objects independent of their orientation and position and can be used e. g. in medical image analysis. The construction of the proposed invariant features does not depend on any preprocessing, e. g. filtering, segmentation, or registration. We first derived the underlying theory and then developed a two step strategy for the efficient computation: The first step consists of the local evaluation of a nonlinear function f for each element of the 3D dataset. In the second step the total integral is build from all local results. The com-

[1]The mean grey-scale value can be constructed using the monomial $f(\mathbf{M}) = \mathbf{M}[0,0,0]$.

putational complexity is linear in the data-set size, which may be too high for large 3D objects. Therefore possibilities for acceleration by parallel/distributed processing or by stochastic sampling have been discussed. First experimental results with artificial 3D objects were presented in the paper to demonstrate the invariant properties of the features. A total number of six classes were defined for the experiments. The classes were composed by the reference object and transformed objects. Five objects were used per class. All classes could be discriminated well by the invariant features. Intended areas of applications are medical volume images, e. g. magnetic resonance images. We are also looking into the possibility to use this algorithm for content based volume image retrieval.

References

[1] T. Andreae, M. Nölle, and G. Schreiber. Embedding cartesian products of graphs into de Bruijn graphs. *Journal of Parallel and Distributed Computing*, 46(2):194–200, Nov. 1997.

[2] H. Burkhardt and S. Siggelkow. Invariant features for discriminating between equivalence classes. In I. Pitas, editor, *Nonlinear Model-Based Image/Video Processing and Analysis*. John Wiley & Sons, to appear.

[3] N. Canterakis. Complete moment invariants and pose determination for orthogonal transformations of 3D objects. Internal Report 1/96, Technische Informatik I, Technische Universität Hamburg-Harburg, 1996.

[4] O. Faugeras. *Three-Dimensional Computer Vision*. The MIT Press, Cambridge, Massachusetts, 1. edition, 1993.

[5] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume 2. Addison-Wesley, 1993.

[6] A. Hurwitz. Über die Erzeugung der Invarianten durch Integration. In *Nachr. Akad. Wiss. Göttingen*, pages 71–89. 1897.

[7] K. Kanatani. *Group-Theoretical Methods in Image Understanding*. Springer Series in Information Sciences. Springer-Verlag, 1990.

[8] J. L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. Massachusetts Institute of Technology, 1992.

[9] T. G. Newman. A group theoretic approach to invariance in pattern recognition. In *Pattern*, pages 407–412, Chicago, 1979.

[10] M. Schael. Invariantenbasierte Objekterkennung aus dreidimensionalen Sensordaten. Master's thesis, AB Technische Informatik I, Technische Universität Hamburg-Harburg, 1996.

[11] M. Schael. Invariant greyscale features for 3d sensordata. Internal Report 9/98, Albert-Ludwigs-Universität, Freiburg, Institut für Informatik, December 1998.

[12] H. Schulz-Mirbach. *Anwendung von Invarianzprinzipien zur Merkmalgewinnung in der Mustererkennung*. PhD thesis, Technische Universität Hamburg-Harburg, Feb. 1995. Reihe 10, Nr. 372, VDI-Verlag.

[13] H. Schulz-Mirbach. Invariant features for gray scale images. In G. Sagerer, S. Posch, and F. Kummert, editors, *17. DAGM - Symposium "Mustererkennung"*, pages 1–14, Bielefeld, 1995. Reihe Informatik aktuell, Springer.

[14] S. Siggelkow and M. Schael. Fast estimation of invariant features. In W. Förstner, J. Buhmann, A. Faber, and P. Faber, editors, *Mustererkennung, DAGM 1999*, Informatik aktuell, Bonn, Sept. 1999. Springer.

[15] Y. F. Wang and A. Pandey. Interpretation of 3D structure and motion using structured lighting. In *Proceedings, Workshop on Interpretation of 3D Scenes (Austin, TX, November 27–29, 1989)*, pages 84–90, Washington, DC., 1989. Computer Society Press, Computer Society Press.

Novelty detection in airframe strain data

Simon J. Hickinbotham and James Austin
Department of Computer Science
University of York, UK
sjh@cs.york.ac.uk

Abstract

The structural health of airframes is often monitored by analysis of the frequency of occurrence matrix (FOOM) produced after each flight. Each cell in the matrix records a stress event of a particular severity. These matrices are used to determine how much of the aircraft's life has been used up in each flight. Unfortunately, the sensors that produce this data are subject to degradation themselves, resulting in corruption of FOOMs. This paper reports a method of automating detection of sensor faults. It is the only known method that is capable of detecting such faults. The method is in essence a dimensionality reduction algorithm coupled to a novelty detection algorithm that produce measures of unusual counts of stress events at the level of the individual cell and unusual distributions of counts over the entire FOOM. Cell-level error is detected using a probability threshold and a sum of standard deviations. FOOM-level error is detected using a novel application of the Eigenface algorithm. Novelty is measured using a mixture of Gaussian model of the data, fitted using the Expectation-Maximisation algorithm.

1 Introduction

The structural health of airframe structures is commonly monitored by a series of strain gauges. During a flight, any deformation and return to a static state is called a stress cycle. Each cycle is classified as a particular event using the mean and extreme loads of the stress cycle. Counts of stress events during a flight are stored in frequency of occurrence matrices, or FOOMs. By examining FOOMs from each flight, the structural health of the aircraft can be monitored [4]. Corrupted FOOMs need to be identified to ensure that the structural health of the aircraft is monitored accurately.

There are two classes of sensor fault. The first is a random addition of counts that can be caused by (for example) electromagnetic current faults. These spurious counts are distributed either randomly throughout the cells in the FOOM, or accumulate in a single (spike) cell in the matrix. The second type of fault (response shift) is caused by a shift in the response of a sensor to the loads it monitors. This change distorts the distribution of counts that should arise as a result of the manoeuvres accomplished during the flight. It has the effect of distorting the distribution of genuine counts in the FOOMs and is rarely visually apparent.

Our approach to the detection of sensor fault is to examine the variability of FOOMs from normal flights both at the level of the individual cell and at the level of the whole FOOM. Our strategy is to apply a number of statistical measures to sets of example FOOMs and use these as inputs to a novelty detector. The measures include a Gaussian model of each cell in the FOOM coupled with an Eigenface representation of data developed by Turk and Pentland [5]. These measurements provide a compact representation of FOOM data without making any assumptions about the underlying distributions of stress cycle counts. In an earlier paper [3], the authors examined the use of a Multi-Layer perceptron (MLP) for the classification of sensor faults. This scheme required data from faulty sensors to be available for training. Here, we circumvent this problem by using a novelty detection scheme as our flight classifier. The novelty detector algorithm uses the Expectation-Maximization algorithm of Dempster, Laird and Rubin [2] to fit a Gaussian basis function model to the parameterised data. The advantage of using a novelty measure is that *any* unusual flight will be detected, providing changes are evident in the measurements we are using.

Figure 1. Graphical representation of a FOOM. Counts are clustered around the low-stress region of the FOOM, shown here as the bright region along the left hand side of the image.

2 The Frequency of Occurrence matrix

During a flight, a stress cycle is a deformation and return to a constant load of the airframe. The FOOM is a two-dimensional triangular histogram of these stress cycles [4]. The mean of the stress cycle determines the horizontal position in the FOOM of the cell that will be incremented. The difference between the maximum and minimum load of the cycle is called the alternating stress, and determines the ver-

0-7695-0750-6/00 $10.00 © 2000 IEEE