only holds if we consider the whole set $\mathcal{C}$. If more information about the curves are given, e.g. if fiducial points are given, then it might be possible to construct invariants which are non-constant and continuous.

Thus the euclidean nature of image distorsion and the projective nature of camera geometry do not interact well. It is possible that one could construct projective invariants which are continuous with respect to some other metric, but would this metric be relevant?

ACKNOWLEDGEMENTS

I would like to thank my supervisor Gunnar Sparr for inspiration and guidance. I would also like to thank my fellow students Anders Heyden and Carl-Gustav Werner for their help.

## REFERENCES

[1] A. Blake and C. Marinos, "Shape from Texture: Estimation, Isotropy and Moments," *Artificial Intelligence*, vol 45, pp. 332-380, 1990.

[2] A. Blake and D. Sinclair"Isoperimetric Normalization of Planar Curves," *IEEE PAMI*, vol. 16, no. 8, pp. 769-777, August 1994.

[3] M. Brady and A. Yuille, "An Extremum Principle for Shape from Contour," *PAMI-6*, no. 3, pp. 288-301, June 1984.

[4] J. B. Burns, R. S. Weiss and E. M. Riseman, "The Non-existence of General-case View-Invariants," in *Geometrical Invariance in Computer Vision Mundy, J. L. and Zisserman, A. editors, MIT Press*, 1992.

[5] S. Carlsson, "Projectively Invariant Decomposition and Recognition of Planar Shapes," *Proc. 4th ICCV, Berlin*, pp. 471-475, May 1993.

[6] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Affine Invariant Model-based Object Recognition," *IEEE Journal of Robotics and Automation*, 6, pp. 578-589, 1990.

[7] J. L. Mundy, and A. Zisserman (editors), *Geometric Invariance in Computer Vision*, MIT Press, Cambridge Ma, USA,, 1992.

[8] C. A. Rothwell, A. Zisserman, D. A. Forsyth and J. L. Mundy, "Canonical Frames for Planar Object Recognition," *Proc. 2nd ECCV*, Genova, Italy, pp. 757-772, 1992.

[9] K. Åström, "A Correspondence Problem in Laser-Guided Navigation," *Proc. Swedish Society for Automated Image Analysis*, Uppsala, Sweden, pp. 141-144, 1992.

[10] K. Åström, "Affine Invariants of Planar Sets," *Proc. 8th Scandinavian Conference on Image Analysis*, Tromsö, Norway, pp. 769-776}, 1993.

[11] K. Åström, "Affine and projective normalization of planar curves and regions," *Proc. 3rd ECCV*, Stockholm, Sweden, vol. II, pp. 439-449, 1994.

# An Evaluation of Intrinsic Dimensionality Estimators

Peter J. Verveer and Robert P.W. Duin

***Abstract* – The intrinsic dimensionality of a data set may be useful for understanding the properties of classifiers applied to it and thereby for the selection of an optimal classifier.**

**In this paper we compare the algorithms for two estimators of the intrinsic dimensionality of a given data set and extend their capabilities. One algorithm is based on the local eigenvalues of the covariance matrix in several small regions in the feature space. The other estimates the intrinsic dimensionality from the distribution of the distances from an arbitrary data vector to a selection of its neighbors.**

**The characteristics of the two estimators are investigated and the results are compared. It is found that both can be applied successfully, but that they might fail in certain cases. The estimators are compared and illustrated using data generated from chromosome banding profiles.**

## I. INTRODUCTION

A traditional practice in the field of pattern recognition is to select a small set of features before training a classifier. Neural network applications, however, have shown many examples where networks are trained successfully having large numbers of inputs, sometimes even larger than the number of training objects, see for example [1–4]. One way to understand this is to assume that in these applications the data is located in some low–dimensional, possibly non–linear subspace of the feature space, see Duin [5]. Due to its non–linear mapping properties a neural network might be able to approximate this low–dimensional subspace by its first layers and to perform the classification in this subspace by its output layer.

Before investigating this hypothesis, the tools to analyze data in non–linear subspaces have to be defined and evaluated. In this paper we report on such an evaluation for estimators of the intrinsic dimensionality. This can be defined as the smallest number of independent parameters that is needed to generate the given data set, see Bennett [6] and Trunk [7].

Consider a set of $L$–dimensional vectors with intrinsic dimensionality $K$. The usual interpretation is that the vectors lie on a possibly non–linear surface with topological dimensionality $K$: they lie in a $K$–dimensional subspace of the $L$–dimensional feature space. We represent a vector $\mathbf{x}$ in this set using the following model:

$$\mathbf{x}=\mathbf{f}(\phi)+\mathbf{u}, \tag{1}$$

where $\mathbf{f}(\phi)$ is an $L$–dimensional (possibly non–linear) function of the $K$–dimensional parameter vector $\phi$. The $L$–dimensional variable $\mathbf{u}$ denotes the noise. If $\mathbf{u}$ is equal to zero the function $\mathbf{f}$ defines a $K$–dimensional surface or sheet $S$ containing the vectors.

If $\mathbf{u}$ is not zero, then $\mathbf{x}$ does not lie exactly on $S$ but will have an offset perpendicular to $S$. Thus the effect of noise on the data set can

Manuscript received October 4, 1993; Revised July 15, 1994. Recommended for acceptance by Dr. Anil K. Jain.

At the time this work was carried out, P.J. Verveer was with the Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands. Currently he is with the Department of Molecular Biology, Max Planck Institute for Biophysical Chemistry, Am Faßberg 11, D–37077 Göttingen, Germany.

R.P.W. Duin is with the Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands.

IEEE Log Number P95005.

be understood by modeling the data set as a $K$–dimensional surface that is not infinitely thin.

To gain better understanding of this effect we examine the set in a small region around a point $\mathbf{y}$ on $S$. We approximate $S$ at $\mathbf{y}$ by a $K$–dimensional linear surface. This surface is spanned by $K$ orthonormal vectors $\mathbf{s}_i$, $i = 1,\ldots,K$. Furthermore we have $L$–$K$ vectors, $\mathbf{n}_j$, pointing in the directions that are perpendicular to $S$, defined by:

$$\mathbf{n}_j \cdot \mathbf{s}_j = 0.$$

So the vectors $\mathbf{s}_i$, and $\mathbf{n}_j$ constitute local descriptions for the function $\mathbf{f}(\phi)$ and the noise $\mathbf{u}$ in (1). Within this small region, we will only be able to see that the vectors are approximately in a linear surface if the largest variance in the directions perpendicular to $S$ is much smaller than the smallest variance in the directions of $S$:

$$\frac{\underset{i}{Min}\left\{Var(\mathbf{s}_i)\right\}}{\underset{j}{Max}\left\{Var(\mathbf{n}_j)\right\}} > \alpha, \quad i = 1,\ldots,K, j = 1,\ldots,L-K \text{ and } \alpha >> 1 \qquad (2)$$

This should be true for all valid choices of $\mathbf{s}_i$ and $\mathbf{n}_j$. A typical value for $\alpha$ used by us is 10. $\mathrm{Var}(\mathbf{s}_i)$ depends on the size of the region that is chosen. $\mathrm{Var}(\mathbf{n}_j)$ depends on the variance caused by the noise (the "thickness" of the surface) and the variance caused by the fact that $S$ cannot be exactly represented by a linear surface. If the region is chosen too large then $\mathrm{Var}(\mathbf{n}_j)$ might be high due to the non–linear nature of $S$. If there is noise then $\mathrm{Var}(\mathbf{n}_j)$ will still have a non zero value, even if $S$ can be perfectly represented by a linear surface. In that case, if the region is chosen too small, equation (2) will not be satisfied. Thus the noise in the data set defines a lower bound to the size of the region that can be chosen. If the noise in the set is large this region should be large, possibly conflicting with the non–linearities in $S$. In that case the intrinsic dimensionality cannot be observed at $\mathbf{y}$.

This analysis shows that an algorithm, that tries to determine the intrinsic dimensionality from local information, may have to deal with the problem of noise. We will see this with the two algorithms that are discussed in this paper.

Another problem arises if the number of vectors in the set is too small to represent the surface $S$. The shape of this surface then will not be retrievable from the vectors and it will be impossible to find the correct value of the intrinsic dimensionality. The sample size has to be sufficiently large to represent the non–linearity in $S$ and to filter out the influence of noise. Even for infinite sample sizes, however, the conflict between noise and non–linearities remains.

In the last few decades, several methods to estimate the intrinsic dimensionality of a given data set have been published. Dubes and Jain [8] give a summary of the literature. Wyse *et al.* [9] compare the existing algorithms that compute the intrinsic dimensionality.

This paper concentrates on two algorithms, that, according to Wyse *et al.*, can be successfully applied to estimate the intrinsic dimensionality of a data set. In this paper an extension of an algorithm published by Fukunaga and Olsen [10] will be presented that reduces the necessary amount of interaction. Furthermore a slightly different implementation of an algorithm by Pettis *et al.* [11] will be presented. This modified algorithm is not iterative, in contrast to the original algorithm. These two algorithms are compared to each other. As an example, the algorithms are applied to find the intrinsic dimensionality of "real" data sets constructed from chromosome banding profiles.

## II. The Local Eigenvalue Algorithm

The algorithm of Fukunaga and Olsen for computing the intrinsic dimensionality, as described in [10] is based on the well–known Karhunen–Loève expansion [12]. For vectors in a linear subspace, the dimensionality of this subspace is equal to the number of non–zero eigenvalues of the covariance matrix. Fukunaga and Olsen assume that the intrinsic dimensionality of a data set can be computed by dividing the data set in small regions. In each region the surface in which the vectors lie is approximately linear and the eigenvalues of the local covariance matrix are computed. They are normalized by dividing them by the largest eigenvalue. The intrinsic dimensionality is defined as the number of normalized eigenvalues that are larger than some threshold $T$. Choosing a low $T$. means ignoring directions with a normalized variance less than $T$. However, if the number of vectors in each region is small, the eigenvalue estimates will not represent well the true variance. Choosing $T$ is then difficult. We therefore developed the following procedure to estimate the intrinsic dimensionality, using the eigenvalues of the local covariance matrix.

Consider a set of $M$ vectors with dimensionality $L$. The $L$ eigenvalues $\mu_i$ of the local covariance matrix are computed for a region consisting of $N$ neighboring vectors from this data set. The eigenvalues are normalized by dividing them by the largest eigenvalue and ordering the result: $\mu_1 = 1$ and $\mu_i \geq \mu_{i+1}$, for $i = 1, \ldots, L$. In order to find the number of significant eigenvalues we study the hypothesis that the vectors lie within a subspace of dimension $i$ and are locally uniformly distributed. In a Monte Carlo experiment we studied the probability that the smallest normalized eigenvalue $v_i$ computed for $N$ points uniformly distributed in an $i$–dimensional sphere is smaller than some threshold $U_{i,N}$:

$$\Pr\left\{v_i < U_{i,N}\right\} = p, \qquad (3)$$

If for a given $p$ (typically p = 0.05) an observed eigenvalue $\mu_i < U_{i,N}$, we decide that $\mu_i$ is not significant, so $K < i$. Thus $p$ is the probability of an incorrect decision $K < i$. For large $N$, (or small $i$) $U_{i,N}$ will approach to one leading to underestimates of the intrinsic dimensionality for deviations of the assumption of locally uniformly distributed vectors. We therefore introduced another threshold $T_{i,N}$, by bounding $U_{i,N}$ with:

$$T_{i,N} = \mathrm{Min}\left\{T_{max}, U_{i,N}\right\} \qquad (4)$$

A typical value for $T_{max}$, is 0.1 (see (2)) stating that only eigenvalues that are at least 10 times smaller than the largest one are neglected. See also the discussion in the introduction concerning the "thickness" of a surface. We computed the values of $U_{i,N}$ using computer simulations and tabulated them for repeated use. Fig. 1 gives $U_{i,N}$ as a function of $N$ for $i$ = 2, 10 and 25. Note the high values of $U_{i,N}$ for high $N$ and low $i$. An estimation $K_{le}$ for the intrinsic dimensionality can now be defined as the smallest value $i$ for which $\mu_i + 1 < T_{i+1, N}$, if $I = 0, \ldots, L$–1, else $K_{le} = L$. Note that in contrast to the original algorithm of Fukunaga and Olsen the threshold may differ for testing different eigenvalues. We will refer to this estimator as the *local eigenvalue estimator*.

The assumption that the vectors are uniformly distributed in a $K$ dimensional linear subspace indicates that $N$ should be chosen as small as possible. However if there is noise present in the set, $N$ has to be chosen larger. Eigenvalues related to the functional behavior will increase if $N$ increases (as it determines the size of the regions), but eigenvalues caused by the noise will not. Therefore, the eigenvalues indicating the noise will be smaller after normalization. Further-

more $N$ vectors span an $N$–1 dimensional space, therefore $N$ should be chosen larger than $K$ otherwise the estimation will be certainly too low.

This procedure is repeated for several regions yielding the local intrinsic dimensionality in each region. These regions might overlap. In [13] we describe a procedure to choose the regions and to control the amount of overlap between the regions.

To find an estimate for the global intrinsic dimensionality the normalized and sorted eigenvalues can be averaged over all regions. These can then be used to compute the intrinsic dimensionality using the method described above. In that case it is assumed that the normalized and ordered eigenvalues are approximately the same in all regions.

## III. The Near Neighbor Algorithm.

Another approach to estimate the intrinsic dimensionality was taken by Pettis *et al.* [11]. Assuming that the vectors are locally uniformly distributed, they derive the following expression for the intrinsic dimensionality:

$$K = \frac{\bar{r}_k}{(\bar{r}_{k+1} - \bar{r}_k)k}, \tag{5}$$

where $\bar{r}_k$ is the average of the distances from each vector to its $k^{\text{th}}$ nearest neighbor. Pettis *et al.* derived the expected value of this estimation for the special case of three uniformly distributed one–dimensional vectors. They found $E(K) \approx 0.9$. Apparently this estimator is biased even for this simple special case.

Pettis *et al.* also describe an iterative algorithm that gives an estimate of the intrinsic dimensionality that is based on an arbitrary number of neighbors. We found that it often did not iterate to a correct value. We derived a non iterative solution from equation (5). If $\bar{r}_k$ is observed for $k = k_{min}$ to $k_{max}$, a least square regression line can be fit to $\bar{r}_k$ as a function of $(\bar{r}_{k+1} - \bar{r}_k)k$. A non-iterative estimation for the intrinsic dimensionality, that will be called $K_{nn}$, can be obtained [13]:

$$K_{nn} = \left( \sum_{k=k_{\min}}^{k_{\max}-1} (\bar{r}_k + 1 - \bar{r}_k)^2 \right)^{-1} \left( \sum_{k=k_{\min}}^{k_{\max}-1} \frac{(\bar{r}_{k+1} - \bar{r}_k)\bar{r}_k}{k} \right) \tag{6}$$

We will call this the *near neighbor estimator*. Note that the result will generally not be an integer. To interpret the result it has to be rounded to the nearest integer value. It can be expected that using more than two neighbors yields more accurate estimations. The assumption that the vectors are locally uniformly distributed implies that $k_{min}$ and $k_{max}$ should be as small as possible. If there is noise present in the data the distances to the first nearest neighbors should not be used: $k_{min} > 1$.

Pettis *et al.* found empirically that outliers tend to distort the estimation of the intrinsic dimensionality. They describe a scheme to remove outliers. We implemented this scheme in our software.

Another problem is the possible influence of *edge effects*. Vectors on or close to the edge of the cluster cannot be assumed to be uniformly distributed. If the fraction of such vectors is high, it can be expected that the estimation of the intrinsic dimensionality will be distorted. This might be the case if the intrinsic dimensionality of the data set is high and the density of the vectors is low due to small sample size. In the limit ($K \rightarrow \infty$) all data points generated within a $K$–dimensional sphere will be on its surface.

We tried to quantify the "edginess" of the data in the following way. For a data point on the edge it holds that there are no points further away from the data set mean in the direction of the vector pointing from the mean to the data point. Thus, if the hyperplane perpendicular to this vector and containing the data point under consideration has no data points on the outside then this is an edge point. The *edginess* of a data set is now defined as the fraction of data points that are edge points. For $K$–dimensional Gaussian distributions with $N$ data points we found by simulations for the edginess $Q(K,N)$: $Q(1,N) = 2/N$, $Q(2, 10) = 0.38$, $Q(2, 100) = 0.06$, $Q(10, 100) = 0.77$, $Q(20, 100) = 0.94$, $Q(15, 1000) = 0.75$, $Q(20, 1000) = 0.99$, $Q(40, 1000) = 1$. From these numbers can be concluded that for many high dimensional problems large fractions of the data will be on the edge.

## IV. Comparison of the Algorithms.

In this section we compare the properties of the two algorithms that were described in the previous sections. For each property that we discuss we compare the algorithms to each other. The following observations are based on experiments that were done on a large set of artificial problems with known intrinsic dimensionality, using uniform and normal distributions, data on spirals, etc. These experiments are described in [13].

- The accuracy of the near neighbor algorithm increases by using the distances to more than two of the nearest neighbors for a given data point.
- For data sets that are too small, the intrinsic dimensionality can not be found. For both algorithms the number of vectors in the set that is needed depends on the nature of the set. However, the near neighbor algorithm usually needs fewer vectors than the local eigenvalue algorithm. This is because the near neighbor algorithm in principle can compute the intrinsic dimensionality from the average of the two nearest neighbors to each vector (if no noise is present).
- In real data sets, noise will be present. We found that the near neighbor algorithm still can be used to compute the intrinsic dimensionality for moderately noisy data. In that case $k_{min}$ has to be chosen larger than one. Similarly the local eigenvalue estimator can still be used if $N$ can be sufficiently increased. We found that the near neighbor algorithm was less sensitive to noise than the local eigenvalue estimator.
- If a large fraction of the vectors in the data set lies at or close to the edge of the set, then the near neighbor estimate will generally be distorted. The local eigenvalue estimator does not suffer from this problem.
- We found that the near neighbor estimator generally yields an underestimate of the intrinsic dimensionality for sets with high intrinsic dimensionality. This is not necessarily the result of edge effects because we also found that the near neighbor estimator was an underestimate for the intrinsic dimensionality for vectors that are uniformly distributed on the surface of a high dimensional sphere. This set has no edges. The local eigenvalue estimator is not an underestimate of the intrinsic dimensionality. However, the number of neighbors in each region needs to be very high to find the correct value of the intrinsic dimensionality for sets with high intrinsic dimensionality. This is because the thresholds $T_i N$, computed using the procedure, described in Section II, are very low for low $N$ and high $i$ (see Fig. 1).

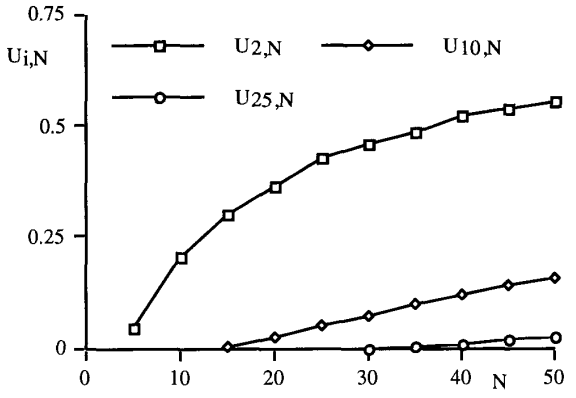


Fig. 1. The upper limits $U_{i,N}$ for the thresholds $T_{i,N}$, as a function of $N$, for $i = 2$, $i = 10$ and $i = 25$.

- Both algorithms that we described are based on the assumption that the vectors in the data set are a uniformly distributed in a local (small) region on the $K$–dimensional surface. We have not detected any problems for sets that had an intentionally non uniform underlying density.
- For the local eigenvalue algorithm, the computer time that is needed depends both on the dimensionality of the data set and the number of regions that is used. The needed time increases quickly if the dimensionality increases, because in each region the eigenvalues of the local covariance matrix need to be computed. For the near–neighbor algorithm considerably less computer time is needed. For this algorithm, the time depends mainly on the number of vectors in the set. Typical CPU times for a 20 dimensional set, containing 1000 vectors are 55 seconds for the local eigenvalue algorithm and 17 seconds for the near neighbor algorithm, on a Silicon Graphics Iris Indigo R4000 computer.

## V. Experiments on "Real" Data Sets

As an example we will study data from sampled chromosome banding profiles. A classical approach is to characterize a banding profile by a fixed set of features. Errington and Graham [14] showed that it is possible to classify chromosomes with a very good performance using the gray values of resampled profiles for the input of a neural network, using as few as 15 samples for each profile. This number is far below the number needed to reconstruct the banding profile for most chromosome classes. So it is likely that the data has some hidden redundancy. Beside the redundancy, the possibility to choose the sampling density freely for the chromosome profiles makes this data type attractive for a study of the intrinsic dimensionality.

We will investigate the intrinsic dimensionality for the sampled profiles of the longest human chromosome. A number of sets have been generated from these profiles, each containing 1000 vectors. Each set is constructed using a different sampling density, yielding different dimensionalities. The sets c20, c40, c60 and c80 have dimensionality of respectively, 20, 40, 60 and 80. Results of experiments on these sets are validated by running the experiments on normally distributed sets. These sets, coded as g20, g40, g60 and g80, have the same dimensionality and size as the chromosome sets. Furthermore, they are generated using the covariance matrix of the corresponding chromosome set.

First the near neighbor algorithm was applied to the sets. The near neighbor algorithm is very useful for a first inspection of the sets because it is so fast. Even for these large sets the edge effect may be important, as discussed in Section III.

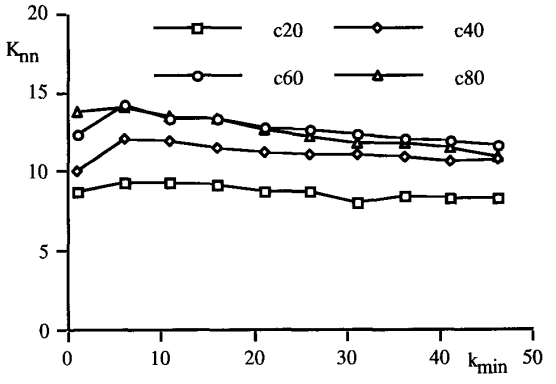


Fig. 2. $K_{nn}$ as a function of the first of five neighbors used to compute it, for sets constructed by sampling chromosome banding profiles at different sampling densities.

Fig. 2 shows the results for the chromosome sets. $K_{nn}$ is plotted against $k_{min}$, the first of five neighbors used to compute $K_{nn}$ using equation (6). Thus $k_{max} = k_{min} + 4$. The results for c60 and c80 are almost identical. Apparently 60 samples are enough to represent the banding profiles. Fig. 3 plots $K_{nn}$ for g60 and g80 along with the results for c60 to show that the results for c60 are considerably lower than for normally distributed sets. Clearly the estimations for c60 are much lower.

According to Fig. 2 the intrinsic dimensionalities of the c60 and c80 sets are approximately 14. Also shown in Fig. 3 are the results for a 18 dimensional normally distributed set with identity covariance matrix. The results are almost equal to the results for c60. Apparently the near neighbor algorithm underestimates the intrinsic dimensionality of a set $K = 18$. The correct value for the intrinsic dimensionality for c60 may be closer to 18 than 14.

The local eigenvalue algorithm has been applied to see if the same values can be found. The sets are divided in slightly overlapping regions. An estimation for the global intrinsic dimensionality is computed from the average of the ordered and normalized eigenvalues over all regions, using the procedure described in Section 2. In this experiment $p = 0.05$ and $T_{max} = 0.1$ were used.

Fig. 4 shows that a large number of vectors are needed in each region, to find an estimation for the chromosome set that is lower than for the corresponding normally distributed set. This is because the threshold $T_{i,N}$ is very low for high $i$ and low $N$. A value of 17 or 18 is found if $N$ is high enough. These are values that we found using the near neighbor estimator. Note, however, that it is not easy to decide which value for $N$ yields the correct estimate, if the local eigenvalue estimator is used.

In [13] we describe similar experiments on a much smaller chromosome. In that case we found a value of six for the intrinsic dimensionality, using both algorithms. We found that the near neighbor algorithm did not underestimate the intrinsic dimensionality of a six dimensional normally distributed set. The results of the experiment that we described here are more difficult to interpret.

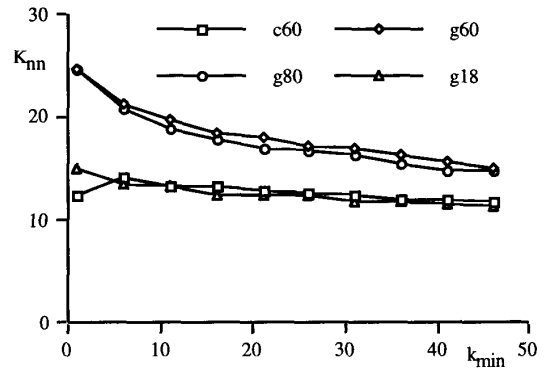


Fig. 3. $K_{nn}$ as a function of the first of five neighbors used to compute it, for two normally distributed sets g60 and g80, generated with the covariance matrices of the sets c60 and c80. The results for an 18 dimensional normally distributed set are also shown. Furthermore the results for c60 (Fig. 2) are repeated here for comparison purposes.

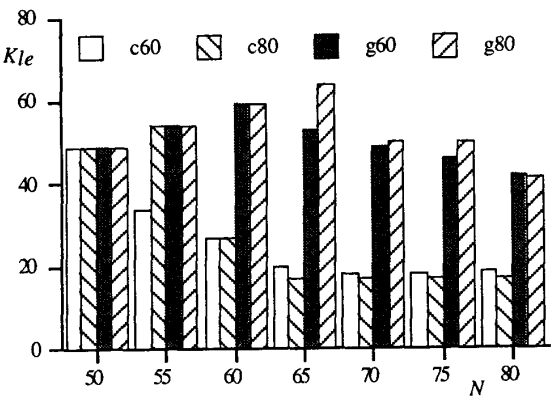


Fig. 4. $K_{le}$ as a function of the number of neighbors in each region for two sets, c60 and c80, constructed by sampling chromosome banding profiles at different sampling densities and for two normally distributed sets g60 and g80.

## VI. Conclusions.

Two existing algorithms to compute the intrinsic dimensionality have been extended and their behavior has been investigated. The use of the local eigenvalue algorithm of Fukunaga and Olsen [10] has been simplified by introducing a method to choose the thresholds that are needed to determine the "significance" of an eigenvalue. An estimator for the global value for the intrinsic dimensionality is defined.

The near neighbor algorithm as published by Pettis *et al.* [11] has been improved by using more than two neighbors in the computation of the intrinsic dimensionality. Moreover, the need for an iterative solution has been removed. This results in more accurate estimates.

The near neighbor algorithm is faster than the local eigenvalue estimator. It is also less sensitive to a low density of the vectors and it is thereby applicable to smaller data sets. However, its estimation of the intrinsic dimensionality will be an underestimate, especially if the intrinsic dimensionality is high. This is made worse by the fact that the algorithm can suffer from edge effects, if the number of vectors is low and the intrinsic dimensionality is high. It is a topic of future research to find a systematic estimate of the bias of the near neighbor technique.

The local eigenvalue algorithm does not suffer from these problems. However, the algorithm requires a higher sampling density, making it less suitable for small data sets. This is especially a problem for sets with high intrinsic dimensionality. Furthermore the local eigenvalue algorithm is more sensitive to noise than the near neighbor algorithm.

For both algorithms, the interpretation of the results is not easy. Choosing $N$ or $k_{min}$ is difficult because the correct choice depends on the nature and the size of the data set. This is illustrated by the experiments on chromosome banding profiles.

These algorithms do not give a final solution for the problem of finding intrinsic dimensionality. The use of these algorithms requires considerable knowledge of the properties of the algorithms. It is helpful to compare the results to those obtained for normally distributed sets that have approximately the same covariance matrix as the data set. Applying both algorithms and comparing the results can clarify the results.

## Acknowledgment

The authors thank Dr. T. Gerdes, Rigshospitalet, Copenhagen, Denmark, for making the chromosome data set, used in Section 5, available to us. Prof. I.T. Young is acknowledged for his continuous support and for proofreading the paper.

## References

[1] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: a parallel network that learns to read aloud," *The John Hopkins University Electrical Engineering and Comp. Science, Technical Report* JHU/EECS–86/01, 1986, Reprinted in J.A. Anderson and E. Rosenfeld, Neurocomputing: Foundations of Research, MIT Press, Cambridge, 1988

[2] D.A. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," *Advances in neural information processing systems, ed Touretzky*, Morgan, Kaufman Publishers, San Mateo, pp. 305–331, 1989

[3] B.A. Golomb, D.T. Lawrence and T.J. Sejnowski, "Sexnet: a neural network identifies sex from human faces," *Advances in neural information processing systems*, ed. Lippman, Moody and Touretzky, Morgan, Kaufman Publishers, San Mateo, pp. 575–577, 1991

[4] S.I. Kamata, R.O. Eason, A. Perez and E. Kawaguchi, "A neural network classifier for LANDSAT image data," *Proceedings 11th IAPR International Conference on Pattern Recognition*, Volume II, Conference B: Pattern Recognition Methodology and Systems (ICPR11, The Hague, The Netherlands, August 30 – September 3, 1992), IEEE Computer Society Press, Los Alamitos, California, pp. 573–576, 1992

[5] R.P.W. Duin, "Superlearning capabilities of neural networks?," *Proc. of the 8th Scandinavian Conference on Image Analysis*, NOBIM, Norwegian Society for Image Processing and Pattern Recognition, Tromsø, Norway, pp. 547–554, 1993

[6] R.S. Bennet, "The intrinsic dimensionality of signal collections," *IEEE Trans. Inform. Theory*, vol. IT–15, pp. 517–525, 1969

[7] G.V. Trunk, "Statistical estimation of the intrinsic dimensionality of a noisy signal collection," *IEEE Trans. Comp.*, vol. C–25, pp. 165–171, 1976

[8] R.C. Dubes and A.K. Jain, *Algorithms for clustering data*, Prentice Hall, 1988

[9] N. Wyse, R. Dubes and A.K. Jain, "A critical evaluation of intrinsic dimensionality algorithms," *Pattern Recognition in Practice*, ed Gelsema, E.S. and Kanal, L.N., North–Holland Publishing Company, pp. 415–425, 1980

[10] K. Fukunaga and D.R. Olsen, "An algorithm for finding intrinsic dimensionality of data," *IEEE Trans. Comp.*, vol. C–20, pp. 176–183, 1971

[11] K.W. Pettis, T.A. Bailey, A.K. Jain and R.C. Dubes, "An intrinsic dimensionality estimator from near–neighbor information," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI–1, pp. 25–37, 1979

[12] K. Fukunaga, *Introduction to statistical pattern recognition*, Academic press, 1972
[13] P.J. Verveer and R.P.W Duin, "Estimators for the intrinsic dimensionality evaluated and applied," Delft University of Technology, Faculty of Applied Physics, Pattern Recognition Group, Technical Report, 1993
[14] Ph.A. Errington and J. Graham, "Application of artificial neural networks to chromosome classification," *Cytometry*, vol. 14, pp. 627–639, 1993

## Automated Evaluation of OCR Zoning

Junichi Kanai, Stephen V. Rice,
Thomas A. Nartker and George Nagy

*Abstract*— **Many current optical character recognition (OCR) systems attempt to decompose printed pages into a set of zones, each containing a single column of text, before converting the characters into coded form. We present a methodology for automatically assessing the accuracy of such decompositions, and demonstrate its use in evaluating six OCR systems.**

*Index Items*—**Document image understanding, page segmentation, layout analysis, performance evaluation metric.**

### I. Introduction

The first step in Optical Character Recognition (OCR) is to locate and order the text to be recognized. Commercial OCR systems allow the user to demarcate the text regions on a page image by drawing boundaries around them. The order of these regions, or zones, is significant. It normally corresponds to the reading order of the page. This process is known as *manual zoning*.

Alternatively, the user can let the OCR system automatically identify text regions and their order. The system finds columns of text and, if they are not part of a table, defines a separate zone for each column so that the generated text will be "de-columnized." This is also known as "galley format." In addition, the system identifies graphic regions in order to exclude them.

Zone representation schemes are not standardized. The following methods are used by commercial OCR systems: bounding rectangles, piecewise rectangles, polygons, and nested rectangles (see Fig. 1.). In some rare instances, zones may overlap. Moreover, deskewing the page alters the zoning. Therefore, geometric comparison of zones is not feasible. A recently proposed alternative to our method is the comparison of the configuration of black pixel sets, but this method precludes testing "black-box" commercial systems [1].

In order to evaluate the accuracy of automatic zoning, we introduce a *zoning metric* based on the number of edit operations required to transform an OCR output to the correct text. The string of characters generated by the system under evaluation on an unzoned page image is compared to the correct text string. All mismatched substrings are identified by a divide-and-conquer string matching algorithm that finds the longest matches first. Then the number of editing operations (deletions, insertions, and moves) required to correct the system output is computed. Finally, the number of editing operations required to correct the output of the same OCR system operating on a manually-zoned version of the page image is subtracted to yield the cost of correcting only zoning errors. All of the above operations, with the exception of producing the "true" string for each page, are performed automatically.

Section II presents the move-counting and string-matching algorithms in greater detail. Section III describes the test data and the experimental protocol. The results obtained by processing 460 printed pages on six commercial systems are discussed in Section IV. Section V points to future research.

Manuscript received July 9, 1993; Revised July 7, 1994. Recommended for acceptance by Dr. Rangachar Kasturi.

J. Kanai, S. V. Rice, and T. A. Nartker are with the Information Science Research Institute at UNLV, Las Vegas, NV 89154-4021, USA.
G. Nagy is with the ECSE Department, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA.

IEEE Log Number P95007

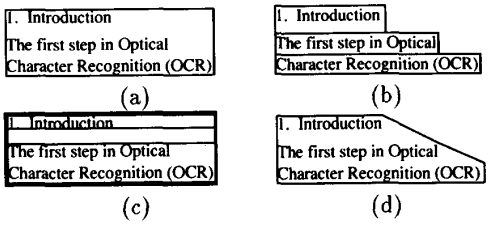


Fig. 1. Zone Representation Schemes. (a) bounding rectangle, (b) piecewise rectangles, (c) nested rectangles, (d) polygon.

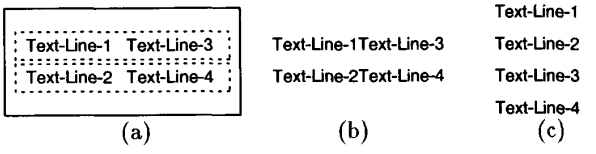


Fig. 2. Example of Zoning Error. (a) incorrectly zoned page, (b) generated text, (c) correct text.

### II. Zoning Metric

A human editor utilizes three kinds of operations to correct OCR-generated text: *insertion, deletion,* and *move.* If a zoning algorithm misclassifies a text region as a graphic region or does not detect a text region, the characters in the text region are not recognized by the OCR algorithm. Thus, the editor must insert (type) the missing characters into the OCR output.

On the other hand, if a graphic region is misclassified as a text region, characters in the graphic region are included in the OCR output. Furthermore, graphic objects could be converted into a set of characters. For example, the vertical axis of a graph might become I's. Such phantom characters must be deleted from the OCR output.

When a multi-column page is incorrectly zoned as shown in Fig. 2., Text-Line-3 must be moved between Text-Line-2 and Text-Line-4. Therefore, the cost of correcting the reading order of an OCR output is an important part of measuring the performance of a zoning algorithm.

A move operation can be performed by either *cut and paste* or *delete and re-type.* The human editor will normally make use of a *cut and paste* capability to move a string of $n$ characters to its correct location. But for $n$ less than some threshold $T$, it is easier (and less costly) to perform $n$ deletions and $n$ insertions to make the correction. The value of $T$ will vary depending on the skills of the human editor and the editing tools at hand, but is most likely to be in the range of 5 to 100.

We propose a two-part algorithm for calculating the cost of correcting the OCR-generated text. First, the minimum number of edit operations is estimated (see Fig. 3.). Next, the total cost is calculated