Document Logical Structure Analysis Based on Perceptive Cycles

Yves Rangoni and Abdel Belaïd

Loria Research Center - Read Group, Vandœuvre-lès-Nancy, France
{rangoni, abelaid}@loria.fr
http://www.loria.fr/∼rangoni/
http://www.loria.fr/∼abelaid/

Abstract. This paper describes a Neural Network (NN) approach for logical document structure extraction. In this NN architecture, called Transparent Neural Network (TNN), the document structure is stretched along the layers, allowing an interpretation decomposition from physical (NN input) to logical (NN output) level. The intermediate layers represent successive interpretation steps. Each neuron is apparent and associated to a logical element. The recognition proceeds by repetitive perceptive cycles propagating the information through the layers. In case of low recognition rate, an enhancement is achieved by error backpropagation leading to correct or pick up a more adapted input feature subset. Several feature subsets are created using a modified filter method. The first experiments performed on scientific documents are encouraging.

1 Introduction

This paper tackles the problem of document logical structure extraction based on physical feature observations within document images. Although this problem has known many solutions, it still remains very challenging for noisy and variable documents.

The literature abounds of structure analysis approaches for different document classes. A survey of the most important approaches can be found in [1]. Most of them are based on formal grammars describing the connection between logical elements. However, these methods have drawbacks because the rules are given by the user and could be not sufficient to handle complex and noisy documents. It is difficult to remove ambiguities and many thresholds must be fixed to process the matching between the physical and the logical structure.

Consequently, a learning-based method seems to be a more adapted solution. Artificial neural network (ANN) approaches allow such a training (rules are learnt) and are known to be more robust to noise and deformation. However, ANN like the classical Multi Layer Perceptron (MLP) is considered as a black box and does not explicit the relationships between the neurons. In the same time, domain-specific knowledge appears essential for document interpretation as mentioned in [2] and it seems useful to keep a part of knowledge in a Document Image Analysis (DIA) system.

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 117–128, 2006.

In order to take into account theses two aspects (knowledge and learning), we propose a new ANN approach that use a Transparent Neural Network (TNN) architecture. This method has the same MLP capacities and can act, in the same time, on the reasoning by introducing knowledge. The recognition task is done progressively by propagation of the inputs (local vision) towards the outputs (global vision). Back-propagation movements, during recognition step, are used for an input correction process as the human perception acts. These successive "perceptive cycles" (vision-interpretation) bring a context return which is very helpful for the input improvement.

This paper is organized as follows. In the first section, the TNN architecture is described. The second section details an input feature clustering method to speed up the perceptive cycles. Finally, in the last section experimental results and discussions are reported.

2 The TNN Architecture Description

The proposed TNN architecture is described in Fig. 1. The first layer receives physical features where each element corresponds to a neuron. The following layers represent the logical structure at three different levels, from fine to coarse (see Fig.8 for the whole input and output names).

All the layers are fully connected and all the neurons carry interpretable concepts. This modeling integrates common knowledge on "general" document

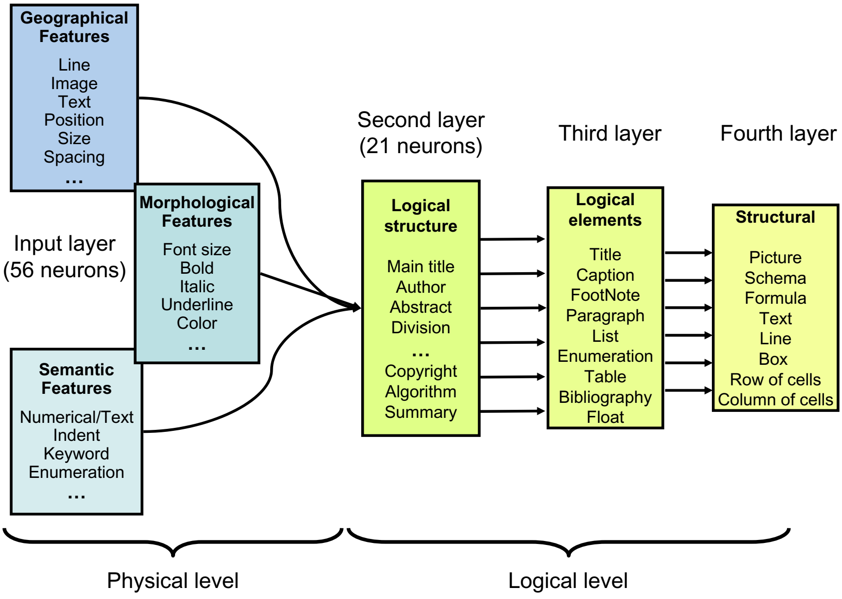


Fig. 1. Neuron semantic for document analysis

structure. It can be more precise if a DTD (Document Type Definition) is given as the DTD organizes the logical element in hierarchy. The real TNN output is the first logical level (the second layer) while the last layers represent the global context (third and fourth layers). They are used to precise the context which is needed for logical structure identification during the perceptive cycles.

This system can be considered as a hybrid method set between a model-driven (DTD integration) and a data-driven approach (training phase). As for a classical MLP, a database is used to train the links between physical and final logical structures. The list of input and output elements is given in Fig. 8.

As the model is transparent and errors can be known for each output layers, the training of the whole network is achieved locally for each consecutive layer pairs. The weight modifications are carried out by an error correction principle. The first stage consists in initializing the weights with random values, then for each couple (Input, Output) of the training database, a predicted value is computed by propagation. The error between the computed output P and the desired output O is then determined. The second stage back-propagates this error in the previous layers. If the activation function is a sigmoid, the value to add to a weight $w_{i,j}$ is $\alpha(O_j - P_i)P_i(1 - P_j)I_i$.

The Fig. 2 shows a small example of a trained TNN that classifies RGB colors in rainbow colors. The green links are used for a positive contribution and the red ones are used for a negative one. The line thickness is proportional to the link weight.

Contrary to a MLP, the recognition process is more complicated. The MLP looks at the maximum output layer component $O_i = argmax\{O\}$ and deduces that the input pattern belongs the i^{th} class. In a TNN system, the outputs are analyzed and two decisions can be chosen (Fig. 3.):

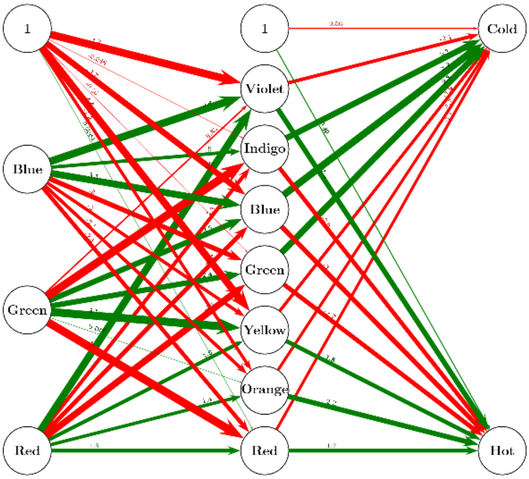


Fig. 2. A TNN classifying RGB pixels in hot and cold colors. It uses rainbow color decomposition in its intermediary layer.

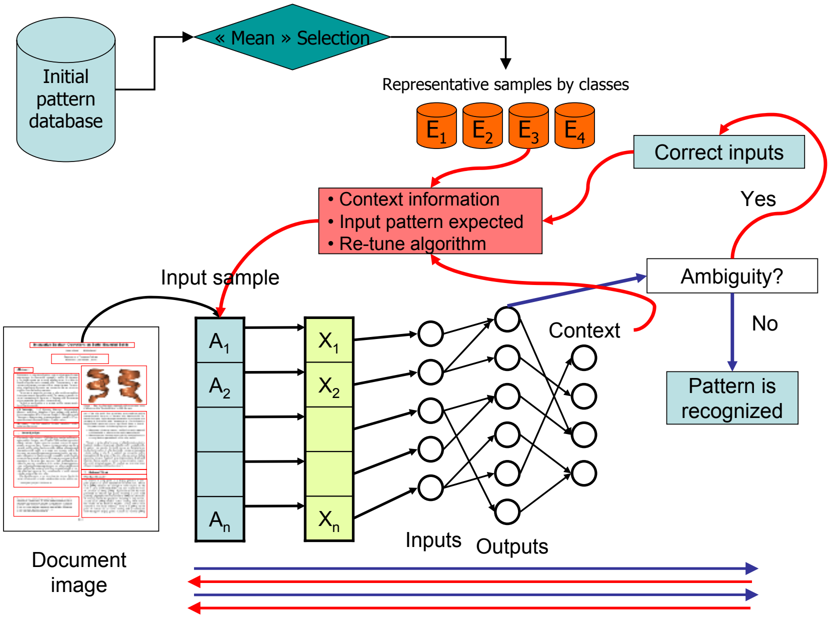


Fig. 3. Perceptive cycles: propagation, analysis, context return, and correction

- the first decision concerns the output when it is close to a unit vector. Thus, the system gives a ruling on a "good" pattern. This means that a class has a sufficient score $\|O\|_\infty \geqslant \varepsilon$ with $0 \ll \varepsilon < 1$ (acceptable class) and this winning class has a score greater than the others $\Gamma(O) = \frac{n((\sum O_i)^2 - \sum O_i^2)}{(n-1)(\sum O_i)^2} \leqslant \eta$ with $0 < \eta \ll 1$ (superior class). If such an output satisfies these rules, the system stops and the pattern is classified.
- the alternative decision occurs when the system reports an ambiguity (i.e. the pattern is confused among several classes). In that case, the latest TNN layers react and propose a context. Thanks to the known neuron semantic, information from upper layers are used to determine the possible or unlikely classes. A hypothesis is created about the possible pattern class and then the input is analyzed in order to find the wrong component values.

As the input physical features (e.g. bounding box, font style, text, etc.) are determined by specific algorithms, it is possible to operate on their precision (or quality) by reconsidering the algorithm parameters, or by changing totally the algorithm method. An example of "re-tuning" can be the OCR settings that give the text. It is possible in an OCR engine to change the amount of computation but changing consequently the recognition quality. The "High Speed" mode is chosen when it is needed to separate text and image whereas "High Quality" mode is preferred if a precise word (a key-word for example) is searched in the text block.

Another example of algorithm "swapping" is the evaluation of word number in a text block. Two solutions can be chosen. The first solution uses RLSA and

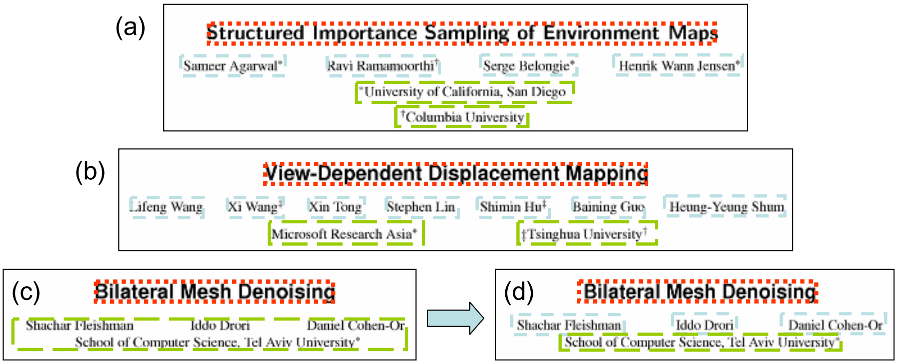


Fig. 4. In (a) and (b), the segmentation and labeling results are correct. Picture (c) shows the result before a context return: the authors are not found. In (d) is the recognition process of (c) after an input correction: title, authors, and locality are found now.

evaluates the number of connected components. The second solution uses an OCR and simply counts the number of words. The first solution is the fastest but gives approximate results whereas the second solution is more time-consuming but more accurate.

With the use of context, new information coming from the training database can be added during the correction. For example, if a segmentation problem occurs, the system finds the "mean" awaited bounding box and corrects the previous bounding box dimensions. This example is not insignificant, because segmentation errors are frequent and penalize the whole physical extraction. The context returns allow often a better segmentation and contribute to better recognition accuracy (see Fig. 4).

3 Input Feature Clustering

In the previous section, the TNN is showed to be able to analyze its outputs and to improve its inputs accordingly. Better results than a MLP can be obtained thanks to the perceptive cycles. However, "high-level" feature extraction is needed for each cycle what remains a time-consuming procedure.

In order to speed-up the global process, the input features are categorized and classified in subsets. The feature subsets are used progressively as TNN inputs. A first feature set is chosen and then if the recognition rate is too low, another set (containing the first and additional features) is selected and so on until reaching the final solution. As the whole features are not necessary needed to classify many patterns the computation is reduced consequently.

Two criteria are used for feature classification: "quality" and "velocity". The "velocity" corresponds to the algorithm execution time given either by experiments or formally by studying the algorithm complexity. For "quality", there is no straightforward measurement method. A specific method based on feature

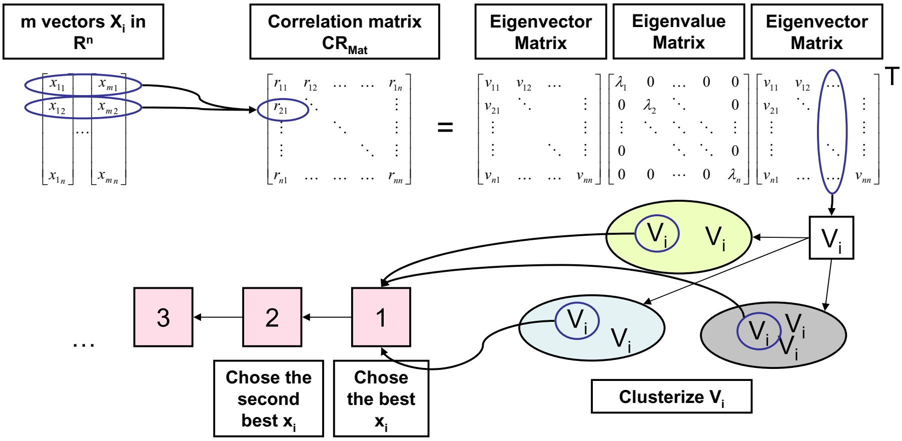


Fig. 5. Data categorization according to predictive capacity

subset selection is proposed. The objective is to determine the best feature combination to feed a pattern classification system. This method is used to create a feature partitioning.

The literature mentions two main feature selection methods: filter and wrapper [3]. The first one selects variables by ranking them with correlation coefficients (it is usually suboptimal for building a predictor, particularly if the variables are redundant). The second one assesses variable subsets according to their usefulness to a given predictor (but the predictor is needed to construct the subsets). As the subsets are needed to construct the TNN architecture, a filter method has been considered. The filter method is also adapted to exclude many redundant variables in the same subset and to keep the most relevant ones.

The Karhunen-Loeve transform is used as a first step in the filter selection method. In [4] we used an extension of the Principal Component Analysis (PCA) in order to build subsets of initial features and not rewrite the features in another base, as the PCA is originally designed.

The eigenvectors V (in absolute value) of the data correlation matrix $CR_{Mat} = (cor(X_i, X_j))$ are computed. The vectors are then clusterized using a Self Organizing Map (SOM) with an Euclidian distance (see Fig. 5.).

The obtained clusters contain similar eigenvectors (i.e. redundant variables). The feature corresponding to the nearest eigenvector from each cluster center is chosen for the creation of a new subset. This subset contains high predictive features which are the least correlated. By fixing the neuron number of the SOM, the number of desired subsets can be chosen.

An important phase of this clustering process is to determine the lower-space dimension q (i.e. the variance to be kept). As no optimal solution exists, some heuristics proposed by the literature have been tested:

- fixed number q: this is a straightforward method where cutting level is imposed by the user.
- fixed percentage: similarly to the previous case, but here the user chooses the first $p\%$ of the eigenvalues.
- cumulated percentage: the number q is determined when the sum of the first variance (eigenvalue) is greater than a given fixed percentage.

These three methods are usual but their choice assumes that the user overcomes its application and can appreciate the dimension to use. These methods are often used in social sciences because it is easier to interpret the data. Two other methods, which are more general and more robust, are based on the shape of the eigenvalue sequence:

- Kaiser method: the average of all the variances is calculated. The space dimension q is determined when the sum of the first variance is greater than this average. Of a wide spread employment, it can be put at fault.
- Cattell method: [5] suggests to find the place where the smooth decrease of eigenvalues appears to level off to the right of the plot (the scree-test). This heuristic is often considered as the most powerful [6]

4 Experimental Results and Discussion

Before introducing results on document image analysis, experiments about "low-level" and highly correlated features are presented.

A first experimentation procedure is employed to illustrate the variable subset creation method. For this purpose, the MNIST database [7] is used. A MultiLayer Perceptron is used to evaluate the group validity. This classifier has the same settings along all the experiments (topology, initial random weights, etc.). Two experiments have been made on this database. The first uses the whole initial pixels of each image (digits are 28×28 pixel images). The second experiment uses resampled images (in a 7×7 format). Thus, we have for the first experiment 784 variables are considered in the first experiment and 49 for the second one.

The MLP is trained with different variable subsets and the recognition rate is chosen as a quality measurement. The subsets are compared to randomly created ones. One thousand of random subsets have been generated and evaluated by the MLP. Then the best one is retained for the comparison. This procedure will be the same for the following experiments about document analysis.

Table 1 shows normalized comparison results between the subset obtained by our method and the best of the random subsets for the initial MNIST database. The Fig. 6 shows the position of the selected pixels in the image. The first picture represents the "mean" digit coming from the whole database ($\frac{1}{n}\sum_{i=1}^{n} I_i$) and in the next three pictures, chosen pixels can be seen. The Table 2 is similar to the Table 1 but here 7×7 pixel images are used for test.

The approach gives good results in spite of the strong influence of each pixel (expecting those on the border) on the classifier. The method keeps the two thirds of the information by keeping less than 4% of features (Table 1: with 25 of the 784 variables, 67.6% of the information is kept).

Table 1. MNIST digit classification accuracy while decreasing the number of features

	Method	
# features	Random	Our selection
784(max)	100%	100%
500	98.4%	99.2%
300	95.9%	98.4%
150	90.5%	96.5%
100	84.2%	94.2%
50	70.9%	87.8%
25	47.1%	67.6%

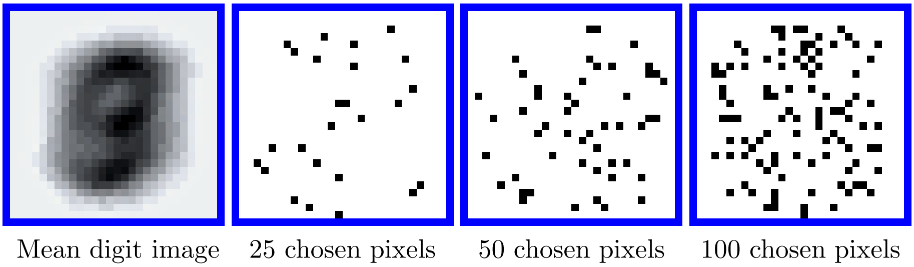


Fig. 6. Feature subsets created for MNIST database

Experiments concerning the document logical structure analysis are presented below. We have chosen as a main database 74 Siggraph 2003 conference papers [8]. The documents are scientific articles having many and diversified logical structure elements (see Fig. 7 for two examples).

In these 74 documents, 21 logical structures are labeled that represents more than 2000 patterns. The input and output features are presented in Fig. 8. Note that all the physical inputs (geometrical, morphological, and semantic) are numerical values between 0 and 1 after possibly normalization. In general, the number represents a percentage (e.g. the percentage of bold characters in a text block) and for other features that represent a number (e.g. the number k of keywords in a text block) we use the series $\sum_{n=1}^{k} 1/(n+1)$ to have a number between 0 and 1.

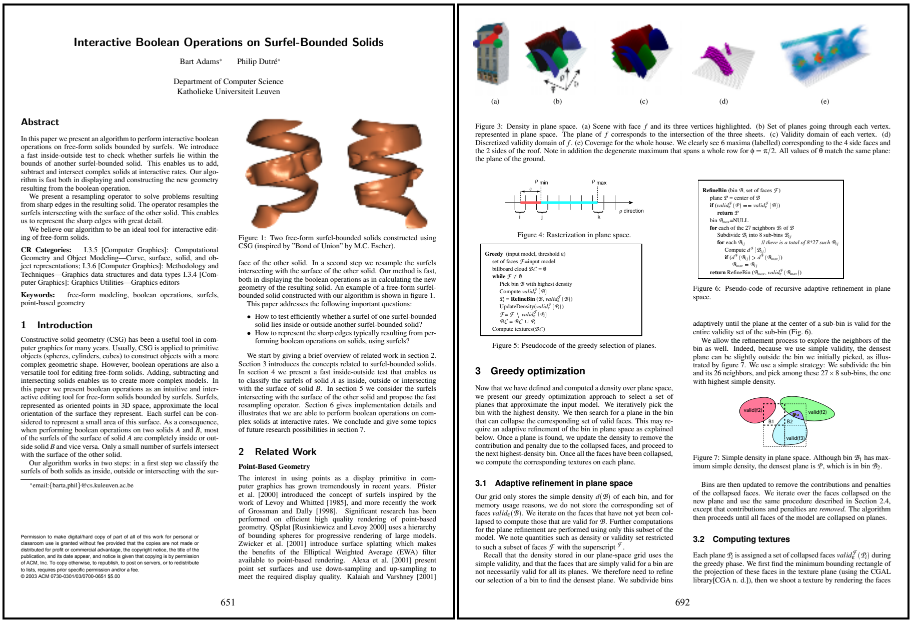
As previously, the same protocol for input feature selection is experimented on this document database. We extracted physical information from the document layout. There are 56 features composed of geometrical, typographical, and morphological information (see Fig. 8) and we use once again a MLP as classifier.

Table 3 synthesizes some results of logical structures recognition accuracy according to the eigenvalue choice methods as mentioned at the end of the previous section. The five methods have been tested on different subset sizes.

The choice of the space dimension influences the results quality. Even if the MLP is a classifier able to give good results with few features, choosing too low

Table 2. Resampled MNIST digit classification accuracy while decreasing the number of features

# features	Method	
	Random	Our selection
49(max)	100%	100%
35	94.2%	99.3%
25	81.2%	88.6%
15	56.2%	70.5%
10	43.9%	55.2%



Interactive Boolean Operations on Surfel-Bounded Solids

Bart Adams* Philip Dutré*

Department of Computer Science
Katholieke Universiteit Leuven

Abstract

In this paper we present an algorithm to perform interactive boolean operations on free-form solids bounded by surfels. We introduce a fast inside-outside test to check whether surfels lie within the bounds of another surfel-bounded solid. This enables us to add, subtract and intersect complex solids at interactive rates. Our algorithm is fast both in displaying and constructing the new geometry resulting from the boolean operation.

We present a resampling operator to solve problems resulting from sharp edges in the resulting solid. The operator resamples the surfels intersecting with the surface of the other solid. This enables us to represent the sharp edges with great detail.

We believe our algorithm to be an ideal tool for interactive editing of free-form solids.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types I.3.4 [Computer Graphics]: Graphics Utilities—Graphics editors

Keywords: free-form modeling, boolean operations, surfels, point-based geometry

1 Introduction

Constructive solid geometry (CSG) has been a useful tool in computer graphics for many years. Usually, CSG is applied to primitive objects (spheres, cylinders, cubes) to construct objects with a more complex geometric shape. However, boolean operations are also a versatile tool for editing free-form solids. Adding, subtracting and intersecting solids enables us to create more complex models. In this paper we present boolean operations as an intuitive and interactive editing tool for free-form solids bounded by surfels. Surfels, represented as oriented points in 3D space, approximate the local orientation of the surface they represent. Each surfel can be considered to represent a small area of this surface. As a consequence, when performing boolean operations on two solids *A* and *B*, most of the surfels of the surface of solid *A* are completely inside or outside solid *B* and vice versa. Only a small number of surfels intersect with the surface of the other solid.

Our algorithm works in two steps: in a first step we classify the surfels of both solids as inside, outside or intersecting with the surface of the other solid. In a second step we resample the surfels intersecting with the surface of the other solid. Our method is fast, both in displaying the boolean operations as in calculating the new geometry of the resulting solid. An example of a free-form surfel-bounded solid constructed with our algorithm is shown in figure 1.

This paper addresses the following important questions:

- How to test efficiently whether a surfel of one surfel-bounded solid lies inside or outside another surfel-bounded solid?
- How to represent the sharp edges typically resulting from performing boolean operations on solids, using surfels?

We start by giving a brief overview of related work in section 2. Section 3 introduces the concepts related to surfel-bounded solids. In section 4 we present a fast inside-outside test that enables us to classify the surfels of solid *A* as inside, outside or intersecting with the surface of solid *B*. In section 5 we consider the surfels intersecting with the surface of the other solid and propose the fast resampling operator. Section 6 gives implementation details and illustrates that we are able to perform boolean operations on complex solids at interactive rates. We conclude and give some topics of future research possibilities in section 7.

Figure 1: Two free-form surfel-bounded solids constructed using CSG (inspired by ”Bond of Union” by M.C. Escher).

2 Related Work

Point-Based Geometry

The interest in using points as a display primitive in computer graphics has grown tremendously in recent years. Pfister et al. [2000] introduced the concept of surfels inspired by the work of Levoy and Whitted [1985], and more recently the work of Grossman and Dally [1998]. Significant research has been performed on efficient high quality rendering of point-based geometry. QSplat [Rusinkiewicz and Levoy 2000] uses a hierarchy of bounding spheres for progressive rendering of large models. Zwicker et al. [2001] introduce surface splatting which makes the benefits of the Elliptical Weighted Average (EWA) filter available to point-based rendering. Alexa et al. [2001] present point set surfaces and use down-sampling and up-sampling to meet the required display quality. Kalaiah and Varshney [2001]

*email:{barta,phil}@cs.kuleuven.ac.be

651

Figure 3: Density in plane space. (a) Scene with face f and its three vertices highlighted. (b) Set of planes going through each vertex. represented in plane space. The plane of f corresponds to the intersection of the three sheets. (c) Validity domain of each vertex. (d) Discretized validity domain of f. (e) Coverage for the whole house. We clearly see 6 maxima (labelled) corresponding to the 4 side faces and the 2 sides of the roof. Note in addition the degenerate maximum that spans a whole row for $\phi = \pi/2$. All values of θ match the same plane: the plane of the ground.

Figure 4: Rasterization in plane space.

```
Greedy (input model, threshold $\varepsilon$)
    set of faces $\mathcal{F}$=input model
    billboard cloud $\mathcal{BC} = \emptyset$
    while $\mathcal{F} \neq \emptyset$
        Pick bin $\mathcal{B}$ with highest density
        Compute $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$
        $\mathcal{P}_i$ = RefineBin ($\mathcal{B}$, $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$)
        UpdateDensity($valid_\varepsilon^{\mathcal{F}}(\mathcal{P}_i)$)
        $\mathcal{F} = \mathcal{F} \setminus valid_\varepsilon^{\mathcal{F}}(\mathcal{P}_i)$
        $\mathcal{BC} = \mathcal{BC} \cup \mathcal{P}_i$
    Compute textures($\mathcal{BC}$)
```

Figure 5: Pseudocode of the greedy selection of planes.

3 Greedy optimization

Now that we have defined and computed a density over plane space, we present our greedy optimization approach to select a set of planes that approximate the input model. We iteratively pick the bin with the highest density. We then search for a plane in the bin that can collapse the corresponding set of valid faces. This may require an adaptive refinement of the bin in plane space as explained below. Once a plane is found, we update the density to remove the contribution and penalty due to the collapsed faces, and proceed to the next highest-density bin. Once all the faces have been collapsed, we compute the corresponding textures on each plane.

3.1 Adaptive refinement in plane space

Our grid only stores the simple density $d(\mathcal{B})$ of each bin, and for memory usage reasons, we do not store the corresponding set of faces $valid_\varepsilon(\mathcal{B})$. We iterate on the faces that have not yet been collapsed to compute those that are valid for $\mathcal{B}$. Further computations for the plane refinement are performed using only this subset of the model. We note quantities such as density or validity set restricted to such a subset of faces $\mathcal{F}$ with the superscript $^{\mathcal{F}}$.

Recall that the density stored in our plane-space grid uses the simple validity, and that the faces that are simply valid for a bin are not necessarily valid for all its planes. We therefore need to refine our selection of a bin to find the densest plane. We subdivide bins adaptively until the plane at the center of a sub-bin is valid for the entire validity set of the sub-bin (Fig. 6).

We allow the refinement process to explore the neighbors of the bin as well. Indeed, because we use simple validity, the densest plane can be slightly outside the bin we initially picked, as illustrated by figure 7. We use a simple strategy: We subdivide the bin and its 26 neighbors, and pick among these 27 × 8 sub-bins, the one with highest simple density.

```
RefineBin (bin $\mathcal{B}$, set of faces $\mathcal{F}$)
    plane $\mathcal{P}$ = center of $\mathcal{B}$
    if ($valid_\varepsilon^{\mathcal{F}}(\mathcal{P}) == valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$)
        return $\mathcal{P}$
    bin $\mathcal{B}_{max}$=NULL
    for each of the 27 neighbors $\mathcal{B}_i$ of $\mathcal{B}$
        Subdivide $\mathcal{B}_i$ into 8 sub-bins $\mathcal{B}_{ij}$
        for each $\mathcal{B}_{ij}$        // there is a total of 8*27 such $\mathcal{B}_{ij}$
            Compute $d^{\mathcal{F}}(\mathcal{B}_{ij})$
            if ($d^{\mathcal{F}}(\mathcal{B}_{ij}) > d^{\mathcal{F}}(\mathcal{B}_{max})$)
                $\mathcal{B}_{max} = \mathcal{B}_{ij}$
    return RefineBin ($\mathcal{B}_{max}$, $valid_\varepsilon^{\mathcal{F}}(\mathcal{B}_{max})$)
```

Figure 6: Pseudo-code of recursive adaptive refinement in plane space.

Figure 7: Simple density in plane space. Although bin $\mathcal{B}_1$ has maximum simple density, the densest plane is $\mathcal{P}$, which is in bin $\mathcal{B}_2$.

Bins are then updated to remove the contributions and penalties of the collapsed faces. We iterate over the faces collapsed on the new plane and use the same procedure described in Section 2.4, except that contributions and penalties are *removed*. The algorithm then proceeds until all faces of the model are collapsed on planes.

3.2 Computing textures

Each plane $\mathcal{P}_i$ is assigned a set of collapsed faces $valid_\varepsilon^{\mathcal{F}}(\mathcal{P}_i)$ during the greedy phase. We first find the minimum bounding rectangle of the projection of these faces in the texture plane (using the CGAL library[CGA n. d.]), then we shoot a texture by rendering the faces

692

Fig. 7. Two scientific document database samples

or too high eigenvector dimension can be bad for the input feature clustering and consequently for the classifier.

It seems here (and for other tests that have be done on MNIST) that the Cattell method (that set $q = 19$) is most of the time better than Kaiser (with $q = 14$). The two methods, which automatically find the number q, give the same or better results than the classical ones where the user must fix this number. We will retain for the following tests the Cattell method that seems to be the most robust on many experimentations.

As expected and confirmed in Table 4, these “high-level” features lends well to this selection.

In this case, the choice of a small set of features is more difficult. The feature clustering method seems to be appropriate when the number of features is rather

Logical	Physical		
	Geometrical	Morphological	Semantic
Title	Text	Bold	IsNumeric
Author	Image	Italic	KeyWords
Email	Table	Underlined	%KnownWords
Locality	Other	Strikethrough	%Punctuation
Abstract	x position	UpperCase	Bullet
Key words	y position	Small Capitals	Enum
CR Categories	Width	Subscript	Language
Introduction	Height	Superscript	Baseline
Paragraph	NumPage	Font Name	
Section	UpSpace	Font Size	
SubSection	BottomSpace	Scaling	
SubSubSection	LeftSpace	Spacing	
List	RightSpace	Alignment	
Enumeration		LeftIndent	
Float		RightIndent	
Conclusion		FirstIndent	
Bibliography		NumLines	
Algorithms		Boxed	
Copyright		Red/Green/Blue	
Acknowledgments			
Page number			

Fig. 8. Logical outputs and physical inputs for documents

Table 3. Logical structure rate accuracy (in %) according to dimensionality q reducing method

Feature number	Fixed number		Fixed %		% Variance		Kaiser	Cattell
	Num	Accur.	F%	Accur.	%V	Accur.	(q=14)	(q=19)
5	2	64.4	2%	61.6	10%	66.5	69.2	68.1
	5	64.3	5%	72.1	20%	67.9		
	10	60.3	10%	64.3	40%	61.4		
	15	59.2	20%	57.8	60%	63.6		
10	2	78.4		79.7		81.1	77.7	82.3
	5	79.8		82.7		73.5		
	10	72.9		77.1		78.4		
	15	70.0		76.6		72.6		
20	2	85.4		82.1		82.3	85.7	86.1
	5	84.9		82.8		86.1		
	10	83.6		83.3		82.3		
	15	82.6		83.3		78.8		
30	2	85.2		82.9		84.2	87.4	88.0
	5	86.8		85.6		85.8		
	10	86.6		86.5		86.7		
	15	86.3		85.4		87.7		

small and can be very powerful in this case (more than 83% of information is kept by dividing the variable number by 5).

Leaving side input features selection, results about complete DIA system are presented. Three input features subsets are created with the previous method.

Table 4. Logical elements classification accuracy while decreasing the number of features

	Method	
# features	Random	Our selection
56(max)	100%	100%
35	86.9%	99.3%
25	65.0%	79.6%
15	51.8%	80.1%
10	35.1%	83.8%
5	17.9%	44.9%

Table 5. Logical classification by MLP and TNN with perceptive cycles

Recognition		TNN			
rates	MLP	C_1	C_2	C_3	C_4
All elements	81.6%	45.2	78.9	90.2	91.7%
Best class	86.9%	66.7	85.3	85.3	99.3%
Worst class	0.0%	0.0	0.0	4.0	28.6%
Recognition time (MLP as reference)	100%	70%	145%	185%	240%

Extraction tools, which can be configured, are used to extract the physical layout. During the recognition phase, the system can choose between the feature subsets and act on extraction tools as mentioned in Section 3. The training stage uses 44 documents and 30 for the test. Test results between a MLP and the TNN at the end of four perceptive cycles are presented in Fig.5.

The perceptive cycles increase the recognition rates. After 4 cycles, the classifier reaches 91.7%. A TNN without perceptive cycles is worse than a MLP (45.2% instead of 81.6%) because TNN does not have many constraints in its intermediate layers. With perceptive cycles, the context returns make it possible to gain in precision while the algorithm complexity increases with a 2.5 factor.

5 Conclusion

We presented in this article a neural network architecture for document logical structure analysis. The method uses a Transparent Neural Network that makes it possible to introduce knowledge in each neuron and to organize in hierarchy the neurons in order to create a "vision" decomposition. The topology can simulate a decomposition hierarchy from fine (the patterns to recognized) to coarse (the global context).

Thanks to this system, we can adapt the computation time according to the pattern granularity and complexity. These“perceptive cycles” as named in cognitive psychology allow simulating in the same system a recognition process that uses automatic and fixed knowledge rules, a hierarchical view, and an interpretation-correction process thanks to hypothesis creation. An input feature clustering was done to speed-up the perceptive cycles.

The TNN gives encouraging results. Although some improvements are in hand, tests are already better than a simple MLP, without adding too heavy computation. In our future works, we will propose a genetic-method to choose representative samples in the database during the context return. Another work will be done to improve the feature subset creation and a method to deal with the final cases of rejected patterns will be presented.

References

1. Mao, S., Rosenfelda, A., Kanungo, T.: Document structure analysis algorithms: A literature survey. SPIE Electronic Imaging (2003)
2. Nagy, G.: Twenty years of document image analysis in pami. PAMI (2000)
3. Guyon, I., Elisseeff, A.: An introduction to variable and feature extraction. Journal of Machine Learning Research (2003)
4. Rangoni, Y., Belaïd, A.: Data categorization for a context return applied to logical document structure recognition. ICDAR (2005)
5. Cattell, R.: The scree test for the number of factors. Multivariate Behavioral Research (1966)
6. Zwick, W.R., Velicer, W.F.: Comparison of five rules for determining the number of components to retain. Psychological Bulletin (1986)
7. LeCun, Y.: (http://yann.lecun.com/exdb/mnist/)
8. Siggraph: http://www.siggraph.org/s2003/. (2003)