Texprint: A New Algorithm to Discriminate Textures Structurally*

Antoni Grau[1], Joan Climent[1], Francesc Serratosa[1], and Alberto Sanfeliu[2]

[1]Dept Automatic Control, Technical University of Catalonia UPC
c/ Pau Gargallo, 5 E-08028 Barcelona, Spain
{agrau,climent}@esaii.upc.es
Universitiy Rovira i Virgili Tarragona
[2] Institute for Robotics, UPC/CSIC
c/ Lloren s i Artigues, 4-6 E-08028 Barcelona, Spain
asanfeliu@iri.upc.es

Abstract. In this work a new algorithm for texture analysis is presented. Over a region with size NxN in the image, a texture print is found by means of counting the number of changes in the sign of the derivative in the gray level intensity function by rows and by columns. These two histograms (**Hx** and **Hy**) are represented as a unique string R of symbols. In order to discriminate different texture regions a distance measure on strings based on minimum-cost sequences of edit operations is computed.

1 Introduction

Texture provides an important feature to characterize and discriminate regions. In general, textural features extraction methods can be classified in statistical and structural methods [5]. In statistical approach, texture is characterized by a set of statistics that are extracted from large ensemble of local properties representing interpixel relationships. On the other hand, structural methods are based on the model that texture is made of a set of elements arranged with some regular placement rule. In [10] sets of connected pixels with similar gray level as elements are extracted and characterized by size, directionality and shape. In [11] texture elements by a region growing procedure are analyzed and the spatial arrangement among them by regularity vectors is described. For texture discrimination, in [7] a syntactic model for the generation of structured elements is proposed. In this work we present a new algorithm to generate the texture print (*texprint*) over regions in an image. This texture print will be the basis for a comparison between texture images and a further discrimination step among texture regions. Because texture is not a perfect pattern repeated along images with similar texture, it is not possible to use an exact matching algorithm to compare *texprints*. To perform such comparison the Levenshtein distance

* This work has been funded by Spanish Department of Science & Technology, DPI2001-2223.

between *texprints*, which are represented by strings, will be found. If the distance is short enough, these *texprints* correspond to similar textures. We propose the use of string-to-string correction problem as placement rules of elements (primitives) obtained statistically. This technique can be applied to pattern recognition [6], [9] and [11]. This paper is organized as follows: in Section 2 we present the algorithm to extract the *texprint*. In Section 3 we propose the use of Levenshtein distance as a measure to discriminate textures with different *texprint*. Experimental results are shown in Section 4.

2 Algorithm for Extracting the *Texprint*

In this Section the algorithm used to compute and generate the *texprint* from a textured image is described. First, the original image is normalized in order to make further steps invariant to illumination changes. Then, the input image is divided into regions, or windows, with *N*x*N*-pixels size. For each region *W*, the algorithm finds two histograms. First histogram, **Hx**, is calculated from pixels in the *W* region in X axis. Second histogram, **Hy**, is calculated in Y axis. Each position in both histograms is incremented after the evaluation of a condition over a row (for **Hx**) or over a column (for **Hy**). The condition is defined as: there will be an increment in a position of the histogram, **Hx**(i) or **Hy**(i), if there is a change in the sign of the first derivative of the gray level intensity in its row or in its column, respectively. This is a measure of the texture appearing in the image. We have seen that images with no texture (smooth texture) present some non-null histograms, for this reason a second condition is defined: the difference between pixels with different derivative has to be greater than a certain threshold *T*. The use of a threshold is due to the fluctuation in the image during the acquisition process.

The algorithm, for **Hx**, can be formalized as follows.

```
For each row, i
  For all the pixels in this row, k
    Evaluate condition 1
    Evaluate condition 2
    If (condition 1 AND condition 2) then Hx(i)++
  endfor
endfor
```

Here, *Condition* 1 is "(S = Sign ($k+1$) * Sign ($k+2$)) < 0"
where Sign ($k+1$) = I(i, k) - I($i,k+1$); and
Sign ($k+2$) = I($i, k+1$) - I($i, k+2$).

The image function is represented by I, indexed by rows and colums (r,c). Following, we can define *Condition* 2: "ABS(S)>Threshold *T*", where ABS is the absolute value.

If *condition* 1 is false there is no change in the increasing or decreasing of the gray level values. For the **Hy** histogram, conditions are similar with the unique difference that the pixels to be evaluated are taken by columns (Y axis). These histograms represent the *texprint* of the evaluated window *W*.

3 Discrimination Step

Since we propose an structural approach, the prints represented by **Hx** and **Hy** can be considered as two strings of symbols (characters) (R_{Hx} and R_{Hy}). If these strings are concatenated ($\oplus$) a new string R with double length is obtained ($R = R_{Hx} \oplus R_{Hy}$). Therefore, the characteristic that defines the texture print over a region is the string R (the new texture print). The problem in the discrimination step is now reduced and it is defined as follows: two texture regions p and q are similar iif their texture prints R_p and R_q approximate match.

The problem of string-matching can generally be classified into exact matching and approximate matching. For exact matching, a single string is matched against a set of strings and this is not the purpose of our work. For approximate string matching, given a string v of some set V of possible strings, we want to know if a string u approximately matches this string, where u belongs to a subset U of V. In our case, V is the global set of texture prints and u and v are texture prints obtained from different texture images. Approximate string matching is based on the string distances that are computed by using the editing operations: substitution, insertion and deletion [12].

Let Σ be a set of symbols and let Σ^* be the set of all finite strings over Σ. Let Λ denote the *null string*. For a string $A = a_1a_2...a_n \in \Sigma^*$, and for all $i, j \in \{1, 2,..., n\}$, let $A<i, j>$ denote the string $a_ia_{i+1}...a_j$, where, by convention $A<i, j> = \Lambda$ if $i > j$.
An *edit operation* s is an ordered pair $(a, b) \neq (\Lambda,\Lambda)$ of strings, each of length less than or equal to 1, denoted by $a \rightarrow b$. An edit operation $a \rightarrow b$ will be called an *insert* operation if $a = \Lambda$, a *delete* operation if $b = \Lambda$, and a substitution operation otherwise.
We say that a string B results from a string A by the edit operation $s = (a \rightarrow b)$, denoted by $A \rightarrow B$ via s, if there are strings C and D such that $A = CaD$ and $B= CbD$.
An *edit sequence* $S:= s_1s_2...s_k$ is a sequence of edit operations. We say that S *takes* A *to* B if there are strings $A_0, A_1, ..., A_k$ such that $A_0 = A$, $A_k = B$ and $A_{i-1} \rightarrow A_i$ via s_i for all $i \in \{1, 2, ..., k\}$.
Now let γ be a cost function that assigns a nonnegative real number $\gamma(s)$ to each edit operation s. For an edit sequence S as above, we define the cost $\gamma(S)$ by $\gamma(S):= \Sigma_{i=1,..,k}\gamma(s_i)$. The *edit distance* $\delta(A, B)$ from string A to string B is now defined by $\delta(A, B):= \min\{\gamma(S) \mid S$ is an edit sequence taking A to $B\}$. We will assume that $\gamma(a \rightarrow b)= \delta(A, B)$ for all edit operations $a \rightarrow b$. The key operation for string matching is the computation of edit distance. Let A and B be strings, and $D(i,j)= \delta(A(1, i), B(1, j))$, $0 \leq i \leq m$, $0 \leq j \leq n$, where m and n are the lengths of A and B respectively, then:

$$D(i,j)= \min\{ D(i\text{-}1,j\text{-}1) + \gamma(A(i) \rightarrow B(j)), D(i\text{-}1,j) + \gamma(A(i) \rightarrow \Lambda), D(i,j\text{-}1) + \gamma(\Lambda \rightarrow B(j)) \} \quad (1)$$

for all $1 \leq i \leq m$, $1 \leq j \leq n$. Determining $\delta(A, B)$ in this way can in fact be seen as determining a minimum weighted path in a weighted directed graph. Note that the arcs of the graph correspond to insertions, deletions and substitutions. The Levenshtein distance (metric) is the minimum-cost edit sequence taking A to B from vertices $v(0,0)$ to $v(n,m)$. In our case both strings have the same length (N) and the algorithm used is $O(N^2)$, [2].

4 Results

Free parameters *N* and *T* (size of the region and threshold, respectively) can not be found in a formal way and it is only from experiments and empirical proofs that a set of optimal values we can obtained to best discriminate textures. First, we show the texture images used in this experiment. These images have been obtained from [1] representing an universal accepted set of textures in many works. We used 20 Brodatz images, figure 1, highly representative form natural textures.

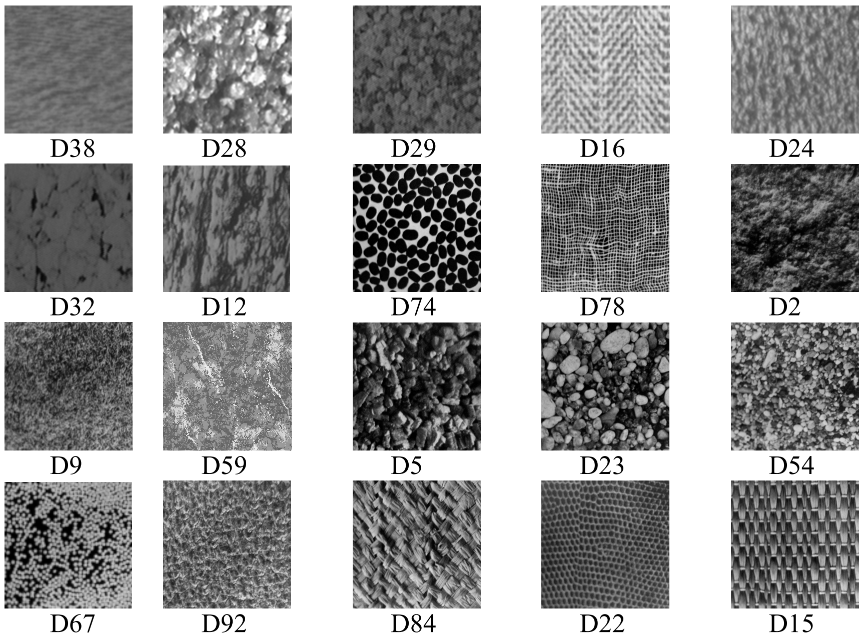

Fig. 1. Brodatz's texture images used in this experiment and their indexes, from [1]

Due to the uncertain parameters *N* and *T*, firstly we will find visually the *texprints* searching for some information 'at a glance'. In figure 2, the shape of the *texprint* can be observed for texture D67, pellets. We choose three values for the threshold *T* (0, 8 and 16, every column in figure 2) and five values for the regions of exploration (8, 16, 32, 48 and 64, every row in figure 2). In each plot, histograms **Hx** and **Hy** have been already concatenated. Their shape was predictable: the bigger the region of exploration, the higher the histogram values, that is, there are more changes of sign in the derivative. Respect the consequences of the threshold, it can be seen its attenuative effect, reducing the number of changes. In such a situation, it is not necessary to normalize the histogram values because, once a region size is chosn, this will be the unique size for all the experiments.

Visually, it is not possible to discriminate any texture from the shape of the *texprint*, but intuitively it contains some outstanding information about the texture of the image. Therefore, a numerical method is needed in order to evaluate the differences between *texprints*. The Levenshtein distance will be the measure of how different the prints are.

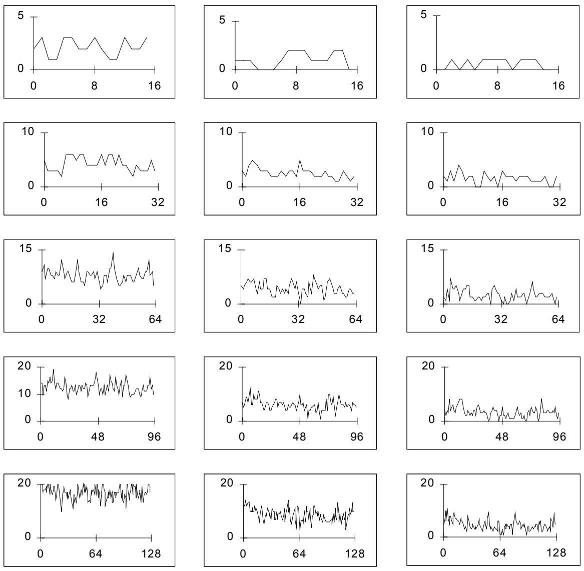
Fig. 2. Some texprints for texture "pellets", D67

As the visual observation of plots do not supply rellevant information, the next step is: for each texture image we compute the distances among *texprints* in the same image. For this proof, different region sizes have been evaluated (*N* ranging from 5 to 64), taken randomly over the image. In a similar way, different values for the threshold *T* have been evaluated (*T* ranging from 1 to 20). The computed distances are an average of 50 distances for a given *N* and *T*.

The distances between *texprints* obtained from the same texture are low and they follow a pattern: when the region of exploration grows and the threshold is low, the distance between *texprints* is higher. This result was predictable: the accumulated number of variations depends directly on the size of the region, while the attenuative effect of the threshold *T* disappears when its value is low. A sample of this proof can be seen in figure 3. For different values N and T a surface map indicates the distance between texture regions in D38 texture (left) and D15 texture, right of figure 3. The costs for insertion and deletion are constants, while the cost of substitution is the difference between the symbols of the string to be substituted. The maximum distances can be found in the upper right corner of each map indicating, with with values for the threshold *T* and low size of the regions, *N*. In the rest of the *T* and *N* values the distance is lower than 2.

The next step in texture discrimination is to observe the distances between different textures and to evaluate whether they are significative enough. We compute the distances among the whole set of textures by pairs of textures. Once more, we use different values for the region of exploration (*N*=5 to 64) and for the threshold (*T*=0 to 20). Each distance is averaged for 50 distances with *N* and *T* fixed.

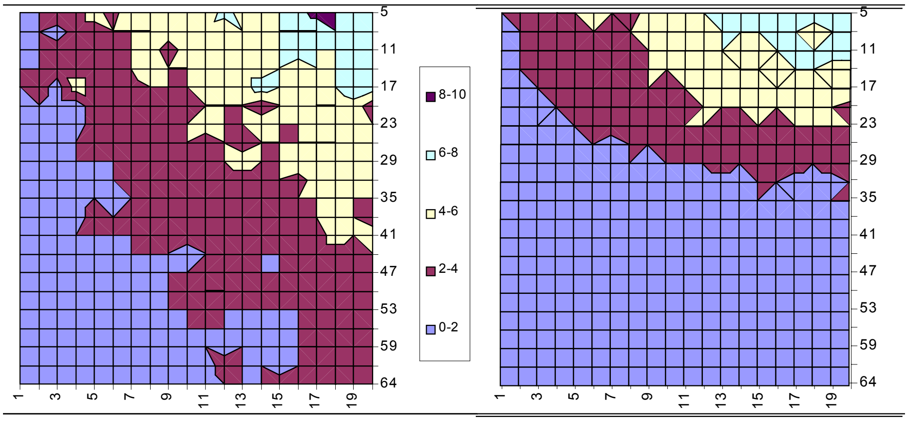

Fig. 3. Left. Distance between texture D38 and D38. Right. Distance between D15 and D15

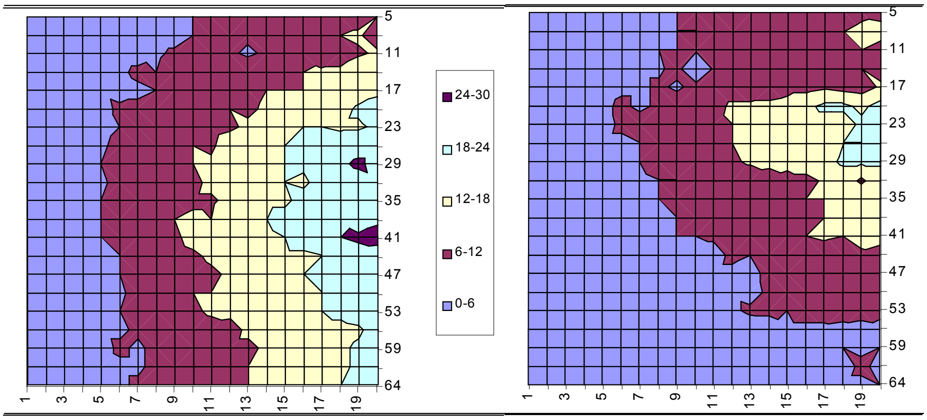

Fig. 4. Left. Distance between texture D38 and D78. Right. Distance between D74 and D92

In figure 4, the distance map of different textures can be seen. The maximum distances has moved to the right center of the map, with high values of the threshold and a medium size for the exploration region. The distances range from 10 to 30 in a big area of the surface map.

Comparing the results between figure 3 and figure 4, the distances between different textures are higher than the distances between a same texture and this effect indicates that the discrimination can be possible. But, a further step is needed in order to find an better value for the size of the regions and the value of the threshold. These values are not always the same for any texture and it is necessary to find the values that best discriminate. For this reason, the best (suboptimal) values can be found by equation (2).

$$\text{Best}\,(N, T) \;=\; \max \frac{\text{Dist}(W_{TexA}, W_{TexB})}{\text{Dist}(W_{TexA}, W_{TexA}) + 1} \tag{2}$$

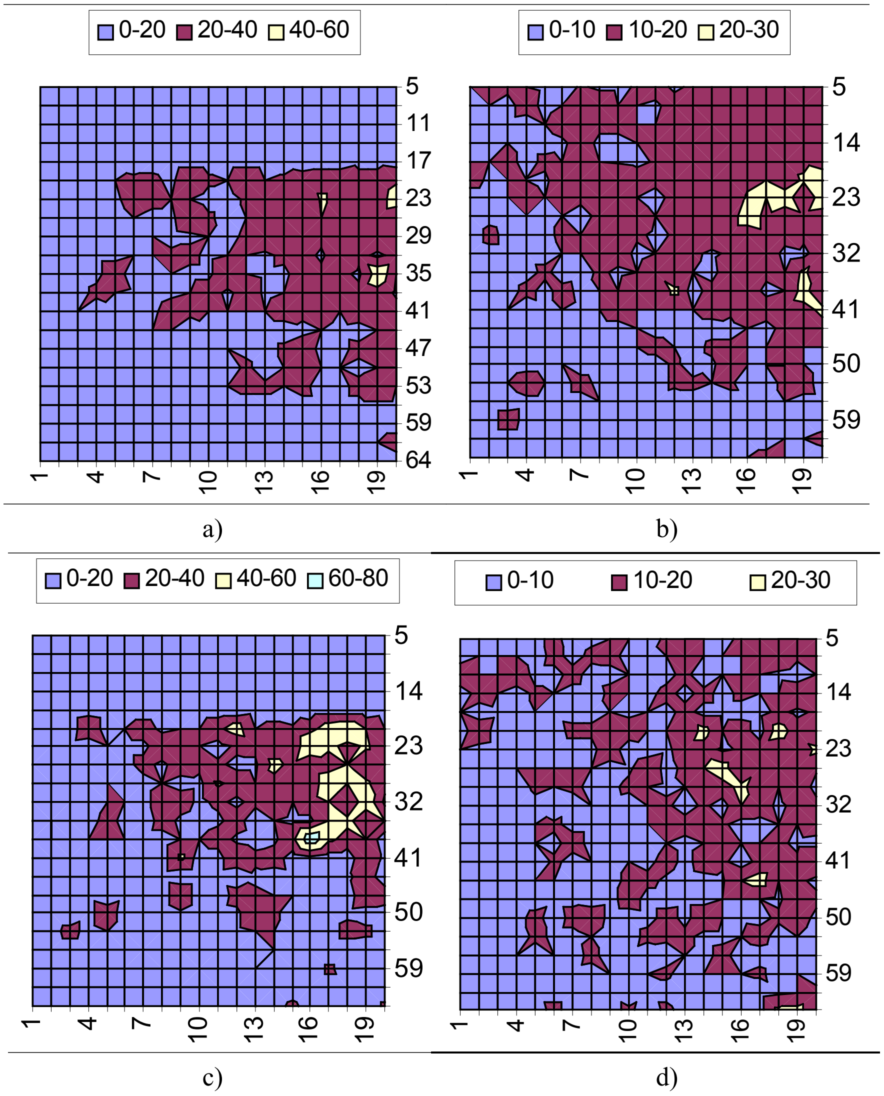

Fig. 5. Plots for the Best N and T parameters, using textures a) D74 and D15; b) D74 and D24; c) D9 and D15; d) D67 and D15

The values that most contribute to the discrimination between texture are those that maximize the relation in equation (2). This quotient is the distance between different texture divided by the distance between similar textures.

In figure 5, a few combinations for best finding the *N* and *T* parametres are shown. The values that maximize the equation (2) are located at the right center of the plot, that means, values ranging from 24 to 40 for the *N* (size of the exploration region) and values from 14 to 19 for *T*, the threshold value. Therefore, we choose *N*=32 and *T*=16, both values inside the intervals and also powers of 2, thinking about present and future hardware implementations for reducing the algorithm cost-time, [3] and [4].

Thus, there is only a last important experiment. Once the values for *N* and *T* are already fixed, we compute the distances among the whole set of available textures. In table 1, these distances are shown and they are normalized respect the size of the region, *N*, because the distance depends on the amount of nodes in the graph.

Table 1. Distances among textures

D38	2																			
D28	16	7																		
D29	33	29	14																	
D16	19	14	23	5																
D24	14	11	27	12	3															
D32	8	15	32	15	12	4														
D12	18	12	30	12	10	13	5													
D74	5	41	30	16	11	12	13	1												
D78	23	46	21	14	18	22	18	21	10											
D2	11	41	27	11	10	10	12	96	17	4										
D9	18	40	27	15	12	16	10	13	17	12	6									
D59	9	42	30	13	11	9	15	7	20	8	14	5								
D5	10	42	29	15	10	7	12	7	20	10	12	8	5							
D23	12	44	28	13	11	10	13	8	19	9	13	8	9	5						
D54	15	41	27	13	11	12	11	11	17	12	11	10	11	10	7					
D67	14	39	29	11	11	16	11	11	17	10	13	14	10	12	10	6				
D92	19	40	23	10	12	14	12	16	15	14	12	16	15	15	12	11	8			
D84	16	41	26	19	11	16	12	15	20	11	11	12	12	11	10	10	11	7		
D22	8	37	29	12	10	7	15	7	20	9	11	6	7	8	10	11	10	11	3	
D15	16	42	29	14	11	12	11	12	18	12	10	11	13	13	12	12	12	11	12	5
Textures	D38	D28	D29	D16	D24	D32	D12	D74	D78	D2	D9	D59	D5	D23	D54	D67	D92	D84	D22	D15

The indexes of table 1 correspond to all the textures used in this experiment. The distances between similar textures (the diagonal of table 1) are, in the whole cases, lower than the distance between different textures. For this reason, we can affirm that a discrimination between textures is achieved using, as a characteristics, histograms **Hx** and **Hy** treated structurally. The quadratic order of the algorithm is not problematic because the length of the strings are short enough to achieve good discrimation results in less than 1 second in a 500-Mhz Pentium PC, for 512x512-pixel input images.

5 Conclusions

We have seen that it is possible to demonstrate empirically certain questions when their formalization is difficult to carry out. Texture has something to do with variations of gray levels in the image and, it is under this assumption that we propose an algorithm for generating a texture print of a region into a image. This print is related to the changes in the sign of the derivative-gray-level intensity function by rows and by columns. These accountings are represented by a string R that will be approximate matched with strings obtained from other texture images. The result of this matching will be measured as a distance based on minimum-cost sequences of edit operations. This approximate matching is translation invariant. Through the results presented above, we verify that texture is an implicit characteristic in the images represented by its gray levels and, moreover, it is possible to discriminate regions with different textures. As a future work, we can consider the cyclic string-to-string correction problem as approximate matching reaching an important improve: the comparison between texture prints will be invariant to rotations but, in the other hand, the algorithm is $O(N^2 \log N)$. Another challenge is to implement this algorithm with a specific architecture to be used in, i.e., robot navigation.

References

1. P. Brodatz, *Textures: A Photographic Album for Artist and Designers*, Dover Publishing Co., New York, 1966.
2. H. Bunke and A. Sanfeliu, *Syntactic and Structural Pattern Recognition Theory and Applications*, Series in Computer Science, Vol. 7, World Scientific Publ., 1990.
3. J. Climent, A. Grau, J. Aranda and A. Sanfeliu, "Low Cost Architecture for Structure Measure Distance Computation", *ICPR'98*, Australia, pp. 1592-1594, August 1998.
4. J. Climent, A. Grau, J. Aranda and A. Sanfeliu ,"Clique-to-Clique Distance Computation Using a Specific Architecture", *SSPR'98*, Sydney, Australia, pp. 405-412, August 1998.
5. R.M. Haralick, "Statistical and Structural Approaches to Texture", *Proc. of the IEEE* 67, No. 5, pp. 786-804, 1979.
6. H.-C. Liu and M.D. Srinath, "Classification of partial shapes using string-to-string matching", *Intell. Robots and Comput. Vision*, SPIE Proc. Vol. 1002, pp. 92-98, 1989.
7. S.Y. Lu and K.S. Fu, "A Syntactic Approach to Texture Analysis", *Computer Graphics & Im. Proc.*, Vol. **7**, No. 3, 1978.
8. T. Matsuyama, K. Saburi and M. Nagao, "A Structural Analyzer for Regularly Arranged Textures", *Computer Graphics and Image Processing*, Vol. **18**, pp. 259-278, 1982.
9. D. Sankoff and J.B. Kruskal, eds, Time Warps, String Edit and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, Reading, MA, 1983.

10. F. Tomita and S. Tsuji, *Computer Analysis of Visual Textures*, Kluwer Academic Publishers, 1990.
11. W.H. Tsai and S.S. Yu, "Attributed string matching with merging for shape recognition", *IEEE Trans. Patt. Anal. Mach. Intell.* **7**, No. 4, pp. 453-462, 1985.
12. R.A. Wagner et al., "The string-to-string correction problem", *J. Ass. Comput. Mach.* 21, No 1, pp. 168-173, 1974.