European Journal of Computer Science and Information Technology

Vol.3, No.1, pp.87-107, March 2015

Published by European Centre for Research Training and Development UK (www.eajournals.org)

ADAPTIVE TECHNIQUES APPLIED TO DOMINANT POINT DETECTION

Barros Neto, L.C. de 1; Diré, G.F. 2,3; Hirakawa, A. R. 4

[1]High Performance Computing Laboratory, Sector Center for Computing and Mathematics (CCMAT), Foundation State University Center of the West Zone (UEZO). Avenue Manuel Caldeira de Alvarenga, 1203, Campo Grande, Rio de Janeiro, 23070-200, Brazil.[2]Laboratory of Chemical and Biological Analysis (LAQB), Foundation State University Center of the West Zone (UEZO).
[3]Estácio de Sá University (UNESA/Estácio), Rio de Janeiro, Brazil.
[4]Associate Professor Researcher at the department of computer and electrical engineering. Polytechnic School, University of São Paulo (USP).

ABSTRACT: *Utilizing Adaptive Finite Automaton (AFA) to implement Adaptive Digitized Straight Line Segments (ADSLS) actuating as exploration automaton of a boundary, we propose an alternative for the available researches on dominant point detection in which primitives are composed by ADSLS. Consequently, this method is shown by simulations to be effective to represent adaptive regions of support and adequate for the complexities of real world scenarios like a shape classifier. Furthermore, even being based in the simple underlying mechanism of Finite Automaton (FA), ADSLS is able to adapt, reacting to circumstance stimuli in a single pass, also presenting learning capability.*

KEYWORDS: Adaptive Systems, Automata, Computational Geometry, Pattern Recognition, Error Correction.

INTRODUCTION

The points where the shape curvature function changes significantly presenting "high curvature values" are called dominant points whose detection is an important step in many shape classification, image processing, computer vision applications or robot navigation because they convey valuable information on the outline of the shape (Prasad & Quek, December, 2013) (Shwetha & Ramya, 2014). Historically, the problem of detecting points of high curvature in 2D shapes has been researched since the early 1970's (Ruberto & Morgera, 2009). In particular, the syntactic method was used to describe the structure of a two-dimensional shape by grammatical rules and the local details by primitives, composing a boundary chain of *m* vectors (You & Fu, 1979). Four attributes were proposed to describe an open curve segment, and the angle between two consecutive curve segments was used to describe the connection. Thus, a curve or arc segment C may be characterized by its curvature function *f(l)*, such as given a point *l* belonging to C, and considering σ as the angle between tangent lines to the arc segment on the points $l - \frac{\Delta l}{2}$ and $l + \frac{\Delta l}{2}$, the curvature function *f(l)* is given by Expression 1, corresponding to the derivative of σ along the segment, relative to the arc length Δl:

$$f(l) = \lim_{\Delta l \to 0} \frac{\sigma}{\Delta l} \qquad (1)$$

A digital arc S is understood as a set of interconnected pixels belonging to a digital image, positioned on a grid such that each point of the set has exactly two neighbors, except two of these points, known as extremes, which have only one neighbor in S (Klette & Rosenfeld, 2004). The terminology "high curvature values" is feasible in a relative sense to the points belonging to an established neighborhood of the dominant point due to that there exist no mathematical definition of curvature for the digital curve. In fact, this neighborhood is called the Support Region of a point of interest, usually predefined (Wu, 2003). Noticing that there are other names in the literature for the dominant points in question, for example, peaks and valleys or corners meaning points on the curve associated with identifiable discontinuities in the mean curvature of the curves. As a matter of fact, usually corner detection takes into account the magnitude of the discontinuity related to the degree of curvature near the corner and the regions of the curve on both sides of the corner.

A drawback encountered in this research is the definition and processing of primitives owing to the extraction of primitives to define shape local properties and the construction of production rules to describe the global structure. By (You & Fu, 1979), if primitives were very simple curve segments with fixed lengths, then it would be necessary to use context-sensitive grammars to take care of the scaling, impeding the use simple devices like the Finite Automaton (FA).This problem attracts significant attention even today in the research community as (Prasad, D. K., August, 2013) described state of art methods showing that in most the duality of precision/reliability and local/global of a fit haunts most fitting algorithms and the majority of modern approaches use local fit to detect dominant points neglecting the global aspect.

By the algorithm proposed by (Gao & Leung, 2002), given a boundary represented by *k* primitives $P_{i-k+1}P_{i-k+2}\ldots P_i$, mergers occur between primitives (i.e. two or more primitives are associated or grouped in a given primitive representing a class) considering l_k and θ_k, the length and orientation angle of the primitive reference respectively. In the opinion of (Gao & Leung, 2002) the angle contribution from a specific primitive to the merger process depends on the length of the contour related to this length primitive, meaning that its contribution to the class is proportional to the amount of primitives of the same type.

A well used technique to represent a shape efficiently is by using chain code primitives in which, originally, the prominence of a corner (the "cornerity" of a point) may be the length product of uniform substrings and the angle discontinuity at that particular point (Abbasi, Olyaee, & Ghafari, 2013). Such substrings are called arms of the chain code on both sides, that is to say, from the left and right of the point traversing the boundary in clockwise and anticlockwise direction.

However, there are disadvantages in chain code schemas as high sensitivity to noise (Feschet, 2008) causing that the oscillation incurvature which cause peaks and valleys in the arches being corrupted requiring primitives with "corner tolerances". This is confirmed by the comments of

(Ruberto & Morgera, 2009) that algorithms are not robust in noisy contours, due to fact that the local maximum curvature may be caused by noisy variations on the curve. In this way, (Ruberto & Morgera, 2009) utilized a final process of refinement to reduce the uncertainties in the application of attributes like angle and local arm lengths in determining corners.

Relatively to the syntactic approach utilizing automaton, very little is found in the literature, an exception is (Dinesh & Guru, 2007) that described a method of dominant point detection useful for its curvature estimation in which a FA is devised to determine an adaptive region of support of a point. However, among the computational costs involved there was the construction of the FA a priory with 120 states, and thus boundary parametric information was not taken in account. Chain code was introduced by Freeman in 1970 (Freeman, 1970) as a one-pixel-thick boundary descriptor in a grid, and digital straightness was conjectured as well. In this model, given a pixel, the main and immediate neighborhoods of this pixel are shown by symbols in Fig. 1.

Understanding the problem from the syntactic point of view, this study involves the concepts of language, grammar and types of grammars (Ramos, Neto, & Ítalo Santiago Vega, 2009). According to Noam Chomsky hierarchy dating back to 1956, languages are classified into four different classes: Recursive Languages (or type 0), Context Sensitive Languages (or Type 1), Context-Free Languages (or type 2) and Regular Languages (or Type 3). There are degrees of complexity related to the classes mentioned since class 3 type is a subset of class Type 2, Type 2 class is a subset of a class type 1, class and type 1 is a subset of Class 0.

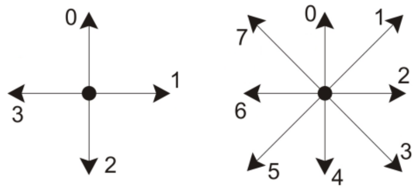


Figure 1: On the left is a graphical representation of the chain code symbols 0-3 of neighborhood-4. On the right, the chain code symbols 0-7 of neighborhood-8.

One reason for the lack of proposals utilizing automata is caused by the computational power required for dominant point detection that inhibits the use of simple devices as FA. This study proposes an alternative for the existing methods through the Adaptive Digitized Straight Line Segment device (ADSLS) (Barros Neto & Hirakawa, Oct. 2014) implemented by Adaptive Finite Automaton (AFA), in which the chain code is applied to represent contour shapes using the facility of adaptive scales such that the number of automaton states, or required memory, vary adaptively according with the curves.

Fig. 2 shows an example of the string of DSLS, in the first quadrant and thus codified by symbols *a* and *b*, always composed of a symbol that is repeated in runs and of an isolated one as in this case of *b* and *a* respectively.

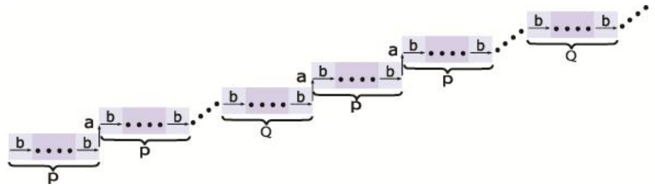


Figure 2: A generic DSLS in the first quadrant, composed of runs of P and Q symbols b, as spaced as possible between codes of a, with P and Q constant integers.

For situations of arcs with local and unreal peaks or valleys, the detection process has to leave out of account these local irregularities, and consider the overall arc to get a panoramic view of the entire outline of the shape including the possibility to compare arc lengths.
However, DSLS of variable lengths and angles to represent the regions of support, which are not necessarily symmetric, are required with powerful recursive languages that inhibit the application of simple formalism for syntactic analysis, such as FA (Barros Neto & Hirakawa, Oct. 2014) - remarks that a regular language is specified by a regular grammar. The concepts of regular language and FA are equivalent in the sense that for every regular language there is at least one FA that recognizes it and vice versa (You & Fu, 1979).

(Barros Neto & Hirakawa, Oct. 2014) presented capabilities of adaptive techniques related to the formalism of ADSLS in computational geometry, we advance further proposing this formalism applied to corner detection utilizing the same AFA, a Turing-powerful device (Neto, 2007).This continuity of this paper is organized as follows. In Section 2 the problem is detailed. In Section 3, the underlying principles necessary for understanding this research are presented. The method involved in this study is described in Section 4, regarding automaton implementation and adaptive grids. Simulations are performed in Section 5 corresponding to proofs of concept aiming at the analysis of this proposal. In Section 6, results are discussed and the position of this research within the state of the art is indicated in Section 7. In Section 8, final considerations are drawn.

PROBLEM STATEMENT

(Dinesh & Guru, 2007), utilized here didactically as the reference paper since it uses an exploration automaton, a pre built FA with 120 states actuating in a kind of backtracking schema to cross the border in clockwise or counterclockwise, which leads it to any final state or reject the input strings representing the outline injected into the FA as input. The number of transitions that the FA takes on the input chain code sequence defines the size of the corresponding arm, fixed to the maximum of 120 states, because this input defines the set of points which come before the point of interest P_i as the left arm and the set of points which come after the point of interest is regarded as right arm.

In the opinion of (Feschet, 2008) the Freeman model tends to generate too much short segments even with partial adaptive solutions like the one described by (Bhowmick&Bhattacharya, 2007). Understanding the region of support of a point P_i as the point itself plus the left and right arms of lengths A and B respectively, the angleα_i made at that point due to its left and right arms may becalculated based in some existing algorithm, for instance the oneproposed by the reference paper:

$$\alpha_i = \arccos\left[\frac{A^2 + B^2 - C^2}{2AB}\right] \qquad (2)$$

, where C is the lengths of a third side, opposed to P_i, connecting the extremes of the left and right arms of a triangle with sides of length A, B and C. A basic case is a square like indicated in Fig. 3where P_i is the reference point and the lengths $A = \overline{P_{i-1} \ P_i}$ is the left arm, $B = \overline{P_i \ P_{i+1}}$ is the right arm, and $C = \overline{P_{i-1} \ P_{i+1}}$.

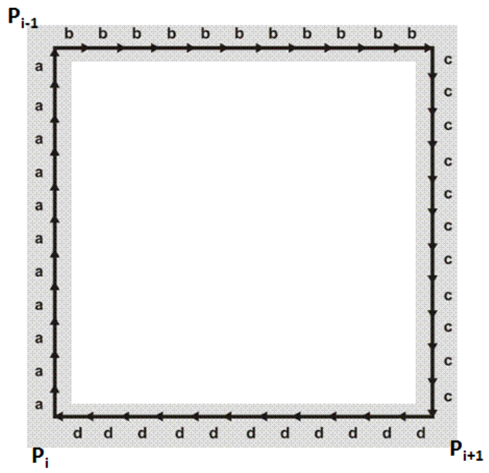


Figure 3: Shape in form of square showing dominant points P_i, P_{i-1} and P_{i+1}.

It followed that the curvature ρ_i at the point P_i may be given by$\rho_i = \frac{1}{\alpha_i}$, selecting as dominant points those that bear local maxima curvature, whose calculus depends on the detection results from the FA. Even without any noise effect, for the Freeman model, the uncertainties described in section 1 are critical also due to change of directions, meaning changing the isolated symbol as result of angle ambiguities in digital arcs conducting to errors in Expression 2.Such uncertainties occur by the very nature of the digitization process, associated with effects depending of the fineness of the sampling grid, and theoretical model implementations exemplified by approximations of the derivatives of the curvature with respect to the length of Expression 1 in digital geometry.

(Lebedev, 2004) performs a coordinate precision analysis concluding that there is an intrinsic error in the Freeman model which causes an inaccuracy in the measurements in this model depending of the thickness of a β zone. (Lebedev, 2004) estimated measurement errors of the coordinates for the corners of a rectangle arbitrarily oriented in a grid, as well as other objects characterized by corresponding segments mutually perpendicular. The noise influence was not presented in more detail, only commented that it is a function of orientation angle, the length of the sides and the relationship between the sides of a rectangle. (Veelaert, 2005) concluded about the process of extracting geometric primitives from an image that has to deal with different types of uncertainty in different stages of the process confirming the need to consider the intrinsic inaccuracies mentioned in this item.

For these reasons (Largeteau-Skapin & Andres, 2006) described transformation operations between the Euclidean and discrete formalisms proposing a discrete smooth scaling operation in order to represent objects in a finer discrete grid (Woo & Park, 2011) also takes into account studies about the length of primitive as a factor in their grouping into classes in the digital case demonstrating that that the determination of the various attributes of arcs involve approximations of theoretical models and compromises between accuracy and allowable tolerances. For example, consider images containing a number of short segments generated by the output of some edge detection algorithm. These segments are considered as individual structures defined by their geometric properties like the corresponding orientation angle or segment length. This idea has the effect of treating such short segments as primitives, which are then called as tokens, considering that normally these tokens lie along straight lines; the lines become the most important structures in an image. In consequence, the region of support detected by pre built FA tends to increase in consuming memory, preventing shape scaling and defective error correction caused by approximations of the regions of support due to the digitization process, as well as the restrictions implicit to the sensitiveness of the Freeman model. The adaptive scales proposed by the present work study aims to use the concept of discrete geometrical simplification based operation, changing the quantization errors adaptively depending on the lengths and curvatures of the arches. With this procedure, the unnecessary details detected for a given grid spacing will not be detected in the other, and vice verse.

FUNDAMENTS

An adaptive device changes its behavior dynamically in response to input stimuli without interferences from other external agents, including users (Neto, 2007). Normally, they are made of two layers comprising a non-adaptive underlying mechanism ND_0, associated to an adaptive layer AM, using the same formalism of the first. This growth in complexity profits not only from the notable increment in expressive power of the combination, but also in versatility, as one can choose any consolidated mechanism as the non-adaptive device. An Adaptive Finite Automaton (AFA) is represented by Expression3 with FA as ND_0.

$$AFA = (ND_0, AM) \qquad (3)$$

The next topic presents a brief review of AFA.

Adaptive Finite Automaton (AFA)

From Expression 3, the adaptive layer AM comprises adaptive actions that work on the original set of rules (ND_0; AM). ND_0 characterizes AFA initial configuration.

Adaptive actions are calls to parametric adaptive functions (ADF) responsible for self modification procedures. Depending on the stimulus *i* from input string, linked to an operational step *i*, AFA configuration ND_{i-1} is modified by adaptive actions, resulting that the FA $ND_{i-1}$is changed to another FA ND_i belonging to the set $\{ND_0, ND_1, ND_2, \ldots, ND_i \ldots : i \geq 0\}$.Furthermore, the AFA formalism regards elementary adaptive actions to be applied to the transition set of the automaton, so that sets of elementary adaptive actions are abstracted in ADF which interconnects the adaptive layer to ND_i, as presented in Fig. 4 through generic ADF R and S.

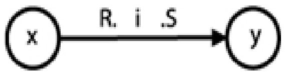


Figure 4: A generic AFA transition where R and S are optional adaptive functions (ADF) responsible for self modification procedures.

Fig. 4 shows the static graphic representation of a generic AFA transition *(x, i): R→y: S*, where *x* is the current state before the transition; *y* is the state after the transition; *i* is the input stimulus before the transition; R is an ADF executed before applying the transition; and, finally, S is an ADF executed after applying the transition. Graphically, any ADF R is portrayed by R. (R followed by point) in case it is of the before type; likewise, any ADF S is an after type if it happens to be denoted by .S (point followed by S). When it comes to formats, there are three modalities of elementary adaptive actions shown in Table 1, specified by a prefix symbol. Given a certain pattern transition enclosed in brackets shown in column SYMBOLOLGY of Table 1 associated to the prefix symbol, where [*(x, i) : R→y :S*] is the pattern to be specified and R and S are optional ADF, the inspection kind searches the current state set for this pattern. The deletion one erases the pattern from the current state set; and the insertion kind adds the pattern to the current set of transitions.

TABLE I: ELEMENTARY ADAPTIVE ACTION FORMAT.

MODALITIES	PREFIX SYMBOL	SIMBOLOGY
INSPECTION	?	?[*(x, i): R→y :S*]
REMOTION	-	-[*(x, i): R→y :S*]
INSERTION	+	+[*(x, i): R→y: S*]

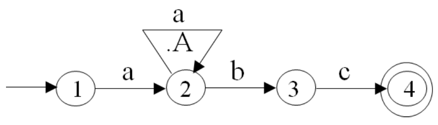
A provision is made so that the inspection type is executed first, next the deletion, and finally the insertion kind; adding that null transitions have the lowest priority. About ADF format, in the general case, it has a heading composed of parameters, generators and variables and a body constituted of elementary adaptive actions. All of them are optional; however, if parameters are specified, they have to be supplied to activate the corresponding ADF. Variables are used in place of any of the components of the elementary adaptive action, further assigned the actual corresponding values in the matching process with the pattern given. Then, after the matching process, variables may be undefined (in case no match is found) or defined (otherwise). Generators are used to assign names to newly created states. Roughly speaking, generators are also like special variables, which are automatically assigned unique values as soon as an ADF is activated. In the activation of an ADF, the assignment of argument values to the parameters occurs, too. Neither generators nor parameters are allowed to change any longer, once assigned.

To differ from variables, generators receive the symbol * as exponent. Fig. 5 shows an example of AFA that recognizes strings of the kind $a^nb^nc^n$, with $n \geq 1$, so that ADF A inserts a new transition that consumes tokens *b* and *c* in the automaton each time token *a* is consumed.

DSLS revisited

From the introduction in Section 1, in neighborhood-4 or neighborhood-8, the chain code is a sequence of elements in which each element is a symbol from Fig. 1that represents the vector joining two neighboring pixels of a digital arch, aiming to represent the digital arch in question. In his model, Freeman stated that strings representing straight lines must obey three properties in neighborhood-8: (Prop1) At most two types of symbols, representing directions in the chain code, can be present, and these can differ by unity module eight. (Prop2) For one of these directions, the run length must be 1. One of the two symbols always occurs singly. (Prop3) Successive occurrences of the single symbol are as uniformly spaced as possible among codes of the other value, which occurs in groups. The meaning of Prop1 to Prop3 is to represent the straight line by a sequence of vectors with multiple slopes of 45^0 and the lengths of which are either 1 (when horizontal or vertical) or $\sqrt{2}$ (whendiagonal).

As the third property Prop3 was considered somewhat unclear, researches proved that the straightness of a digital arc can be determined by the absence of unevenness in its chain code, necessary and sufficient for meeting the chord property, extracted from the brief historical presented by (Barros Neto, Hirakawa, & Massola, 2011):
A digital arc A is said to have the chord property if for every two digital points *c* and *d* in A, and for each point p = (x, y) on $\overline{cd}$, there is a point e = (h, k) of A such that max$\max\{|x-h|,|y-h|\} <$ where $\overline{cd}$ is the line segment between *c* and *d*.



```
A{
var1,var2,
ger1*,ger2*:
-[(var1,b)->3]
+[(var1,b)->ger1]
+[(ger1,b)->3]

-[(var2,c)->4]
+[(var2,c)->ger2]
+[(ger2,c)->4]
}
```

Figure 5: Example of AFA that recognizes string $a^nb^nc^n$: $\boldsymbol{n \geq 1}$.

The chord property implied establishing a hierarchical structure composed of consecutive numbers corresponding to the runs and runs of runs of the symbols specified by Prop1 and Prop2. This structure of consecutive numbers is expressed by an additional property Prop4.
In sequence, it was demonstrated that there can be only two possible lengths of these different runs, which are two consecutive integers (for example, P and P+1). On the other hand, studies showed examples of DSLS that violate the regularity implicit in the chord property, commenting that, in practice, Prop3 and Prop4 are enviable in digital arcs (Klette & Rosenfeld, 2004). However, it is more reasonable to expect a slight variation in the runs, within a tolerance level, but always keeping the overall slope, thus defining an approximate DSLS.

Therefore, a criterion used concentrated on strings that satisfied the first two properties of the Freeman conjecture, called monotonic codes, as they represent digital arcs that are either ascending or descending, with reference to coordinate axis *x* and *y*.

In order to keep the slope of a digital line, the smallest segment of a DSLS is called the Unit of the Straight Line Segment (USLS), resulting in mathematical models. (Barros Neto & Hirakawa, Oct. 2014) stated an enhanced method taking into account that the adaptive representation can express changes in the scales of segments. Therefore, an irregular trajectory may be detected as DSLS after it is reviewed in a compatible scale, using metrics.
In summary, adaptivity can be an alternative to incorporating the fundamentals of arithmetic discrete geometry to Freeman's in corner detection.

Codification

If nothing else is specified, without loss of generality, in this paper neighborhood-4 (see Fig. 1) is the default, so that the symbols of property Prop1 must be consecutive, module four. More precisely, the symbols that make up strings belong to $\Sigma = \{a, b, c, d\}$. To satisfy Prop1, just consider module 4 along with *a=0, b=1, c=2, d=3*, for neighborhood-4 of Fig. 1 and Fig. 2. Any string $S = s_1 \dots s_n$ may be represented by its symbol, followed by the indication of its *i*-th element s_i giving Expression4:

$S\colon s_i; i = 1,2, \dots, n.$ (4)

In Expression 4, *n* denotes the length of string S which means $|S| = n$. Symbols $s_i \in \Sigma$ may be called token, chain code elements or stimuli, too. The null string *n*=0 is represented by ε. If all symbols of S are identical, $S = s_1 = s_2 = \cdots = s_{n-1} = s_n = s$ a compact representation is $S = s^n$. Note that null transition causes automaton non determinism. A deterministic automaton may be redesigned using markers such as $\Delta \notin \Sigma$ in place of ε.

Adaptive DSLS (ADSLS)

ADSLS uses a modified chord property for models of higher orders (order *n*) incorporating tolerances in angle and in length of DSLS. The modified chord property changes neighborhood of chord property into a variable neighborhood function such asmax$\{|x-h|, |y-h|\} < n$, where *n* is the order of the model that depends on the momentary situation and the length of the segment, to sum up, of the stimuli. That is to say, the neighborhood function of DSLS must have a relatively large width, proportional to the measured length towards the overall linear structure. Regarding techniques for error recovery in this study, it is often convenient to represent the real numbers in a given circumference and not in a straight line, as usual. Especially, from the circumference of unit length, when defining an arbitrary origin point, we represent any point T by its measured distance around the circle in a counterclockwise direction (this by definition). The division of the circle can be from the Farey series in the form of spyrographs described on page 326 of (Klette & Rosenfeld, 2004).

The techniques of error recovery of syntactic analysis of DSLS employ an approach similar to spyrographs in the form of adaptive loops, such that, by these loops, the circumference is built by states of the AFA, which moves cyclically and continuously through the closed loop. In effect, adaptive loops have their total number of states according to tolerance levels. In order to simplify the description of automata, take in to account that the abbreviation HTST means a sequence head-to-toe of transitions that consume the same symbol; besides, each state belonging to the sequence may be specified by the first state followed by its respective sequential index. The extremes of a hypothetic DSLS may be truncated or completely out of the global structural model. In the former case, the sequence should be accepted; in the latter, rejected. Fig. 6 exposes an AFA which tests the first USLS of a DSLS $(a^4b)^n$. Parameter r4 is the last state of the HTST starting in r. From this arrangement, the AFA removes up to four Δ transitions by ADF RA. Elucidating, each time RA is activated by token *a*, it removes from the automaton one of the Δ transitions that constitutesthe HTST. Furthermore, any token *b* received conducts the AFA to the final state; if more than four tokens *a* are received, the sequence is rejected. The analysis of the other extreme is quite similar.

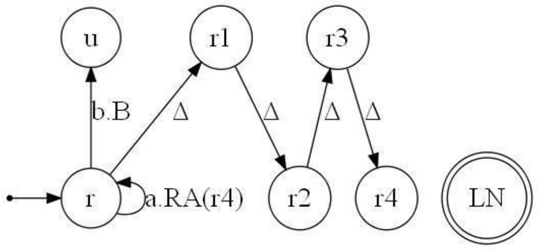


Figure 6: AFA initial configuration to detect a DSLS considering slope errors, including testing the first USLS of a DSLS from $(a^4b)^n$ model.

Recognition of DSLS subjected to slope errors is exemplified by $\{a^n b: n = 3, 4, 5\}$. Fig. 6 shows the initial configuration of the automaton prepared to accept truncated USLS₁ similar to the last item. With the first token *b* consumed, ADF B is activated, which removes transitions of the initial configuration, changing the automaton topology to that of Fig. 7.

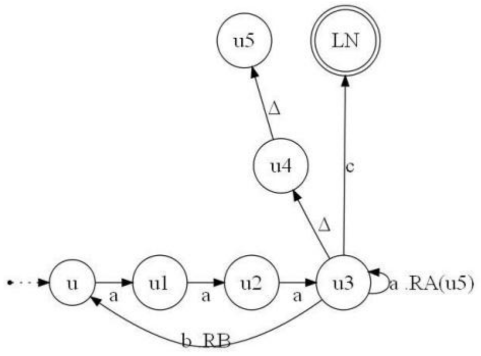


Figure 7: Configuration of AFA of Fig. 6 after the activation of ADF B.

Afterwards, the AFA starts to consume the succeeding USLS*i*: *i*>1 until the input stream is exhausted. A token *c* is included just to signalize the end of the DSLS, when the automaton reaches the final state if the process is successful. On the other hand, if more than 5 tokens *a* are received, RA removes transition of marker *c* to the final state, rejecting the sequence.
Strings of Fig. 8 show the performance of the AFA. These strings follow the model $USLS_i = \{a^n b: n = 3, 4, 5\}$, truncating USLS₁ in some strings, too. Strings out of this model are rejected. Note that the AFA performance does not depend on the length of the input DSLS.

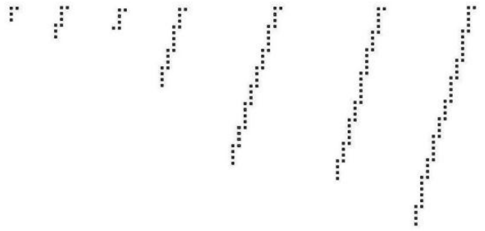


Figure 8: Examples of DSLS accepted by AFA.

***DSLS Length Similarity*.** The method to represent and to apply tolerances is by a graph, or a loop such that the number of states of the loop (that is, its size) is changed adaptively in function, for example, of angle θ_s related to axis x; besides, θ_s gives the main direction of DSLS S, obtaining a syntactic measurement parameter $(1-\psi)$ relative to S detailed in (Barros Neto & Hirakawa, Oct. 2014). Fig. 9 shows a loop containing t_o states, ranging from L_1 to L_{to}.

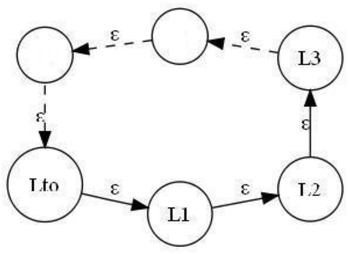


Figure 9: A generic cyclic loop.

Considering that this graph is only for consultation by the automaton, the specific symbol of the transitions between its states becomes irrelevant. The reasoning with this procedure is that, for each symbol belonging to the DSLS S such that $|S| = n$, the automaton has to access the loop, advancing counterclockwise through the cycle, circulating through the graph as many times as the value of n. Factor $0 < \psi < 1$ is an error rate meaning a small percentage of n, inasmuch as a simple case in which the length is estimated by the number of symbols in the neighborhood-4, given by Expression 5; such that given a string S with length $|S| = n$, the corresponding SLRD of length $l_F = n$, this loop allows that the AFA corrects l_F to l_E , where l_E is the corrected estimated length by the Expression 5.

$$l_E \approx \psi l_F \qquad (5)$$

Since n is a variable depending on lengths, the loop is adapted by changing the amount of states t_o of Fig. 9, and the main angle of the DSLS θ_s.

$$t_0 \approx \lfloor {}^{1}/_{(1-\psi)} \rfloor \qquad (6)$$

For a given value of n, the AFA pumps a primitive (for instance, any symbol not belonging to Σ) for each circle on the loop thereby obtaining a syntactic measurement parameter $(1-\psi)$ relative to S. It follows that, in the end, $\lfloor {}^{n}/_{t0} \rfloor$ symbols would have been pumped with the last symbol of S. Therefore, $\lfloor {}^{n}/_{t0} \rfloor$ symbols can be excluded from the total n to obtain the corrected estimated length l_E. It is also possible to implement inequalities in ADSLS by which to judge lengths of two segments $|S_1| = n$ and $|S_2| = m$, in a range of values of Expression 7:

$$n - \lfloor {}^{n}/_{to} \rfloor \leq |S_2| = m \leq n + \lfloor {}^{n}/_{to} \rfloor \qquad (7)$$

PROPOSED METHOD

This study aims to investigate the use of ADSLS as exploration automaton responsible by adaptive techniques. Again, the reference paper bases its implementation and algorithmic in a pre built FA with 120 states actuating as exploration automaton of contours represented by strings that are injected in the automaton as input, to transverse the boundary in clockwise and anticlockwise, which leads it to either the final state or to reject the sequence. Supposing a string $S: s_i; i =$

$1,2, \ldots, n,$ with symbols $s_i \in \Sigma$ that defines a boundary and to determine the left arm of a point $s_j \in S$, the symbols from the point s_j are extended in clockwise direction whereas to determine the right arm, the points are extended in counter clockwise direction in a kind of backtracking.

Notwithstanding, the method proposed here apply the ADSLS in place of the FA with angle and length tolerances adjusted according with the local and global parameters without backtracking to detect the arms since properties of DSLS are used to guide the automaton. Flexibility of extension of this method may be seen by previous works like (Sousa &Hirakawa, 2005) that showed the feasibility of implementing robot navigation systems composed of a navigation automaton operating to work in conjunction with a manager system and a navigation automaton, but restricted to horizontal and vertical trajectories of robots.

The contribution in corner detection here is summarized by the following: *i*) Bringing ADSLS into play to use adaptivity to represent the errors, the tolerances and parameters involved. *ii*) Concurrently, using adaptive grid resolution resulting in adaptable scales. *iii*) The number of states, or required memory, varies according with the curve because the region of support of each point is determined by their local and global properties. Strings resulting from adaptive scale factors that change the length of primitives are recognized by the same set of automata: spatial variability of grid by adaptivity does not require changing the automata. As a result, analysis of situations of region of support depends on the scale involved. If a path segment $\overline{AB}$ is relatively too short compared with the overall trajectory, it is advantageous to use primitive dimensions large enough to be relatively compatible with the size of segment $\overline{AB}$. If segment $\overline{AB}$ is relatively large, primitive dimensions have to be short enough to detect shorter segments. Therefore, the adaptive scale proposed by this research reuses, in a new context, a technique of (You & Fu, 1979) known since the 1970s actualized by improvements in computational power obtained by adaptivity.

The adaptive neighborhood.

The purpose of this research, in which arcs and straight line segments can be in various scales delimited by adaptive boundary conditions, is such that these conditions define the envelope of a digital region on the variable sampling grid depending of the adaptive scale and of the stimuli (input string and parameters like the length of input SLRD).The representation of the different instances of the ideal Freeman's model affected by the angle errors requires the SLRDA act in a range of angles to be changed depending on the stimuli, following a modified chord property. Among these stimuli to affect the adaptive neighborhood, emphasizes is given to the length of the input SLRD. A digital arc C is said to present the modified chord property if, for every two digital points c and d belonging to C, and for each point p = (x, y) on $\overline{cd}$, there is a point e = (h, k) belonging to C such that $\max\{|x-h|, |y-h|\} < n$, with $n \geq 1$. $\overline{cd}$ is the line segment between c and d, and n considers the length of the digitized segment adaptively

In this definition of modified chord property stands out the following: *i*) The neighborhood $\max\{|x-h|, |y-h|\} < n$; *ii*) n is the model order dependent on the momentary situation, the stimulus and the segment length; *iii*) The neighborhood function of a DSLS must

have a relatively large width (proportional to the measured length) in the direction of the overall linear structure.
Fig. 10 outline an example of proposed models by this definition where P depends on the segment length and its type is variable: $P(a, b^m): 0 \leq m < \infty$.

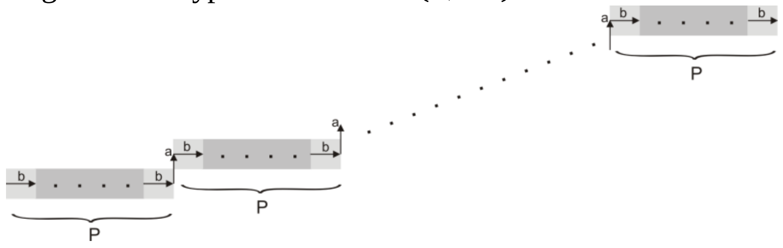


Figure10: Superior order models (order n), with P variable.

Relating to the concept of curvature for the digital case, the topics that follow show experiments of discrimination of basic shapes through SLRDA with intent to change adaptively the corresponding neighborhood related to P variable.

EXPERIMENTS

Dominant point detection is intrinsically related to shape. Due this fact we use basic shapes for the experiments because the exploration automaton has to compare the shape sides (the arms) represented by the input string. Examples of AFA for dominant point detection are described for two cases: free of errors and considering errors. Errors may occur in angle and length. For simplicity, ADF codifications are not shown.

Region of Support Similarity: slope errors.

The AFA of Fig.11 recognizes an input stream W = $S_1S_2S_3S_4$, concatenation of four DSLS: S_1, S_2, S_3, S_4. Each DSLS is an arm.

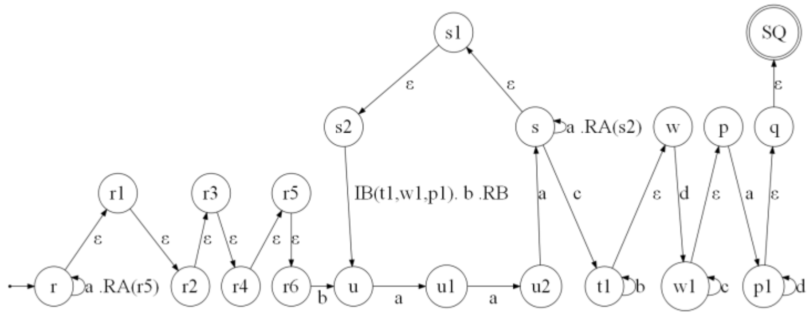


Figure 11: Initial Configuration of AA that Recognizes W = $S_1S_2S_3S_4$ without Length Errors.

To demonstrate the effect of slope errors in classification, the automaton of Fig. 11was fed with strings W represented in Fig.12 varying the slope of individual DSLS. Albeit constituting by DSLS with variable slopes, the string $[(ba^4)^4b(cb^3)(cb^4)(cb^3)(cb^4)c(dc^3)^4d(ad^3)^3(ad^4)a]$ of Fig. 12-awas accepted by the AFA, even so combined slope variations from sides distorts the square. Even though the AA is capable of capture the global geometric property, accepting the sequence that could be rejected by a method more local. String $[(ba^3)(ba^4)(ba^3)(ba^4)b(cb^3)(cb^4)(cb^3)(cb_4)c(dc^3)^4d(ad^3)^4a]$ of Fig.12-b was accepted too, but, comparing with the previous string, this one introduces slope variations in the first shape side, so that total combination of slope variation from all sides diminishes shape distortions visually.

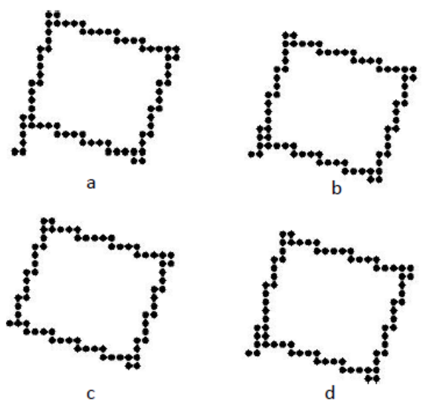


Figure 12: Shapes composed of DSLS with variable slopes

String [$(ba^3)^4b(cb^3)(cb^4)(cb^3)(cb^4)c(dc^3)^4d(ad^3)^4a$] of Fig.12-c, was accepted resulting in a shape with less distortions than those that occurred before. The first, third and fourth shape sides do not have much slope variations. Slope variations from the third side are difficult to visualize, yet not an ideal DSLS; conversely the square is visually identified with easy.

In Fig.12-d, the string was rejected because its USLS (ba^5) is out of the defined range. USLS (ba^5) is easily identified visually in the overall formation:

[$(ba^3)(ba^5)(ba^3)2b(cb^3)(cb^4)(cb^3)(cb^4)c(dc^3)^4d(ad^3)^4a$].

About string of Fig. 12-d, elementary changes in some ADF, like RA, would permit accepting the sequence, adjusting the classifier if required by specifications, otherwise segmentation algorithms should be more precise (Shwetha & Ramya, 2014).

Region of Support Similarity: scaling.

AFA of Fig.13, constructed by modifications of previous ADF, recognizes the triangle fed by $W = S_1S_2S_3$.

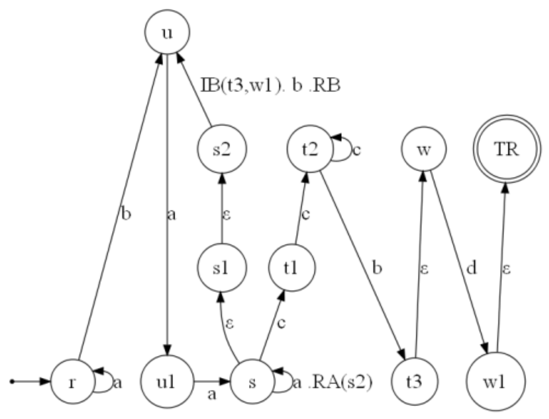


Figure 13: Example of AFA for Triangle Classification.

Strings of Fig. 14 represent the same shape in two different scales recognized by the AFA of Fig. 13.

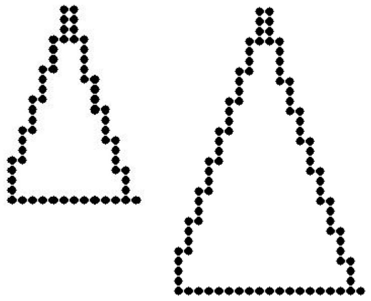


Figure 14: Shape scaling.

Region of Support Similarity: length errors.

Approximate length is informed to the AFA by Ψ in W =Ψ$S_1S_2S_3$, where: *i*) Ψ is such that Ψ=y^{to} is a string composed by an auxiliary token *y*, in order to inform the error length tolerance to be employed related to the angle θ_S of some string S; *ii*) S_2 and S_3 are observed region of support strings to be compared by Expression 7 with S_1within the tolerance (1-ψ). The number of states on the loop is constructed by the AFA of Fig. 15 obtained by Expression 6 from the string Ψ, according with the number of symbols *y*.

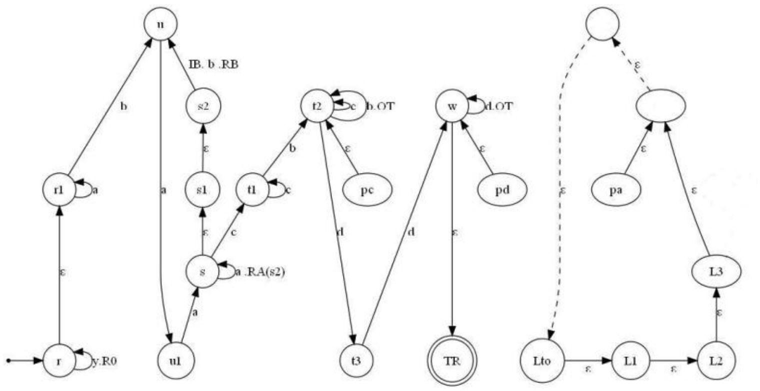


Figure 15: Initial Configuration of AFA for Classifications of Triangle Affected by Length Errors.

A brief explanation of this automaton is as follows. Refer to the AFA of Fig.15 showing the loop of Fig.9 to the right and 3 pointers, *pa*, *pc*, *pd*. Clarifying, pointers are elementary null transitions, for instance, pointer *pd* is pointing to *w* by the transition [(pd, ε) → w], called simply by its fixed state pd.

In case tolerance is informed to the AFA by tokens *y* from Expression 6, adaptive function RO must consume tokens *y*, constructing the loop, introducing the 3 pointers, too. Transition [(r1, a) → r1] is a simplification to consume possible starting symbols *a* of $USLS_1$. With first symbol *b*, the AFA begin to consume subsequent $USLS_i$. As already described, each time an $USLS_i$ from the first side of the triangle is consumed, ADF IB and RB are activated. Besides constructing the templates of the next 2 sides between states t1 to t2:second side; t3 to t4:third side; ADF IB has now the new task to turn the loop by pointer *pa*.

Each time *pa* makes a complete turn through the loop, ADF IB executes the following: *i*) Includes one new USLS in the loop [(t2, b) → t2 : OT] and another in [(w, d) → w : OT]; *ii*) Inserts a transition from the last state of each USLS included before: in the first loop, one that consumes *d* to *t3*, in the second, a null transition to TR ; *iii*) Moves *pc* to a state just one USLS below in direction to*t1*, inserting one transition that consume *d* from the state pointed by *pc* to *t3*; *iv*) Moves *pd* to point just one USLS "below" in direction to *t3*, inserting one null transition from the state pointed by *pd* toTR.

Nonetheless, in case ADF OT is activated, it just removes the transition [(w, ε) → TR] to the final state, so as to shape is too big, out of tolerance, rejecting the input string. In the same way, too small shapes are rejected because the AFA would not find either a *d* transition to *t3*, or a null transition to final state TR.

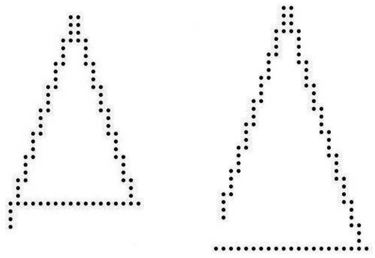


Figure 16: Example of shapes classified correctly with 20% tolerance. Left $(a^3b)^9(c3b)^8d^{17}$; Right:$(a^3b)^9(c^3b)^{11}d^{20}$

We give only two examples of DSLS length errors, since any shape attending the tolerance defined by tokens *y* is accepted, no matter its scale; otherwise it is rejected. Thus, shapes of Fig.16 were accepted by the AFA of Fig. 15 with *y* defining a length tolerance of 20%. Both shapes are easily identified visually, though the shape to the right does not close, splitting the contour because of length errors. On the contrary, the one to the left presented a smaller contour than the ideal triangle, leaving outside the $USLS_1$ of first side. This contour distortion between the two shapes occurred because, first, length of the second side changed from $8USLS_i$ to $11USLS_i$; second, length of third side changed from 17d to 20d.

Implementation and tests of adaptive neighborhoods.

The adaptive neighborhood associates a set of arcs to a corresponding SLRDA which recognizes that set. This topic presents implementation and testing of SLRDA models attending the proposal of adaptive neighborhood associated with the regions delimited by arches responsible for their contours. There are two main types of these delimiters: concave and convex arches, being possible to have a combination of these two types. This text shows only the case of boundaries neighborhood delimited by concave arcs because the convex case presents similarities to the concave. Adaptive loop of Fig. 9 was the structure used to obtain length correction. In this topic the same structure is applied having the number of loop states used by the automaton the meaning of reference length for changing its neighborhood adaptively.

The AFA of Fig. 17 recognizes SLRD whose neighborhoods are changed adaptively according to their lengths. In place of empty transitions that introduce non-determinism, it is used the markerΔ.In previous sections, the adaptive loop of Fig. 9was the structure used to obtain the corrected lengths l_E by Expression 5. In this topic, the same structure is applied considering to of the loop the meaning of length reference to the AFA to change the neighborhood adaptively.

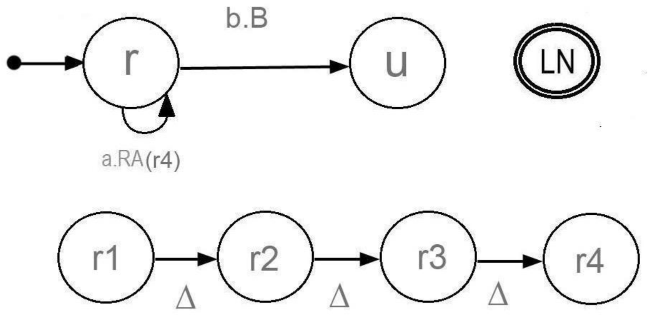


Figure17: AFA initial configuration that implements SLRDA with adaptive neighborhoods variable depending of the length of the input SLRD. Each time RA is activated, a ***Δ*** transition is removed.

For simplification, the ADF of the AFA of Fig. 17 responsible to build the loop of tostates is not shown. In Fig. 17, ADF B is activated with the first symbol b changing the AFA configuration to Fig. 18, to the left, continuing the AFA to consume subsequent USLR in such a way that each USLR from S activates both ADF IB e RB.

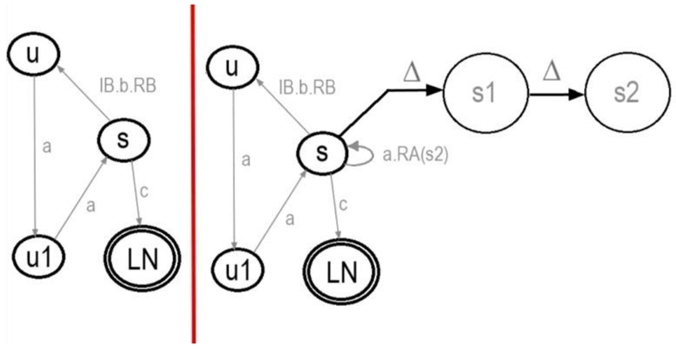


Figure 18: AFA configuration after activation of ADF B of Fig. 17 is to the left. Digital neighborhood is adjusted depending on t_o of the adaptive loop by the number of Δ transitions. Configuration on the beginning of the third loop circle with state s_2 integrated in the topology is to the right.

For an input string S composed of λ USLR U_i $S: U_i$; $i = 1,2, \dots, \lambda$, the AFA configuration shown to the left of Fig. 18 is maintained while$1 \leq i < to$ accepting USLR of the type $\{a^l b: l = 2\}$; but when $i = to$, ADF IB is activated inserting a new Δ transition in the interconnected sequence of state s composed of Δ transitions of the AFA. For $t_0 \leq i < 2to$, accepted USLR are changed to the range $\{a^l b: l = 2{,}3\}$. For $i = 2to$, ADF IB is activated inserting a new Δ transition in the AFA changing its configuration to the one of Fig. 18-right. Accepted USLR changes to the range$\{a^l b: l = 2{,}3{,}4\}$; these process continues in such a way that each$i$ multiple of t_0, ADF IB is activated inserting one Δ transition to alter the neighborhood.

Summarizing the process, each USLR of S activates ADF IB and RB such that: i) The transitions of interconnected sequence of Δ transitions s_1, s_2, s_3... s_v that exist in the automata are removed by

ADF RB (this because the previous USLR cannot have the maximum amount of tokens a related to the AFA neighborhood); *ii*) ADF IB inserts again the same automaton transitions s_1, s_2, s_3... s_v removed by ADF RB in order to enable their consume by the following USLR; *iii*) The ADF IB moves a pointer one position ahead in the adaptive loop; i*v*) Each full turn in the adaptive loop made by the pointer, by activation of ADF IB, the following actions are performed: *a*) Change of SLRDA neighborhood by including a transition (s_v, Δ)$\rightarrow s_{v+1}$ incrementing the interconnected sequence s_1, s_2, s_3... s_v with the state s_{v+1}; *b*) Changing the ADF RB to remove from the AFA, in step *i*) above also transitions connected to state s_{v+1}, included in the previous step; *c*) exclusion (s, a) $\rightarrow$s.RA(s_v) transition and insertion of the transition (s, a) $\rightarrow$s.RA(s_{v+1}).

The neighborhood is delimitated by concave arc due to the fact that the maximum angle of each USLR increases gradually following the angles arctan(2), arctang(3), arctang(4)....arctan($\lfloor \lambda/to \rfloor$)according with the number of states transitions with markerΔ.

RESULTS AND SUGGESTIONS FOR FURTHER WORK

Confronting this working with the traditional models without adaptivity, a first aspect is the question of the flexibility of the models. The traditional models without adaptivity have to satisfy the chord property, rejecting the arches of Fig. 19 characterized by USLR U range of $U(b, a^m)$: $2 \leq m < 5$, since to be in agreement with a chord property there must be a pattern of consecutive numbers on the runs without presenting irregularities. In Fig. 19, the left arc is encoded by
$(a^3\, b)^6 (a^4\, b)^5 (a^5\, b)^5 (a^6\, b)^2 a^4 b\, (a^3\, b)^6$. The arch to the right in Fig .19 is encoded by$(a^3\, b)^6 (a^4\, b)^5 (a^5\, b)^5 (a^6\, b)^5$.

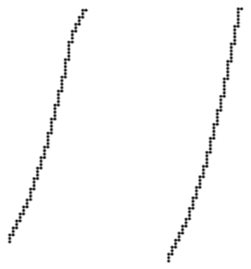


Figure19. Examples of SLRD accepted by the AFA of Fig. 17 with $to = 5$. However they are not recognized by traditional methods without adaptivity.

About the symbol that occurs isolated on the runs, note that *b* is maintained on the ache of Fig. 19, beginning with USLS of angle *arctang*(3) related to the *x* axis, but this angle decrease gradually in the subsequent USLR in function of the quantity of tokens a, until a minimum, representing a convex arch. Consequently, characterized by invariable behavior traditional methods tend to segment digital arches in great quantity of short segments (Feschet, 2008). As for the methods with preset values in a range, the number of segmented short segments is reduced, but they lose the overall characteristics of the arcs, rejecting the ADSLS of Fig. 19, segregated in many ADSLS. Therefore, the dynamic changes in neighborhood of this study, which alters the functionality of the algorithms depending on the stimulus, allow an overall better characterization of the arcs.

Concerning the computational complexity due to limitations of invariable behavior in a predetermined range, the traditional methods tend to divide the digital arcs in some set composed of large number of short segments. By significantly reducing the elements of this set, adaptivity minimizes the data structures required to represent the arcs. Furthermore, the introduction of adaptive tolerances simplifies algorithms compared the one presented by invariable behavior. Therefore, the representation of arcs by a set consisting of even smaller number of segments minimizes the complexity in time of the adaptive algorithm, besides makes it linear.

It follows that a shape can be represented by a set of ADSLS. The points on ADSLS cannot be considered as the dominant points corresponding to first order curvature function. Intersection points between ADSLS are candidates for dominant points presenting higher order curvature function corresponding to points related to changes of directions. These higher order changes are reflected in the changes of the isolated occurring symbol, increasing the curvature of the arch on these points. Moreover higher order curvature arc contains abrupt changes in its curvature in relation to the total length of the arch; therefore, its pattern of curvature variation differs from a first order arch.

Accordingly with arcs presented in the experiments, the SLRDA acting as exploration automaton operates properly allowing to implement, as continuation of this work, a structure in which the exploration automaton should be reconfigured by an adaptive process manager system and a navigation automaton with responsibility for analysis of arches with higher order curvature function to optimize the global nature of analysis. In this way the local and global aspects of the detection process will be optimized simultaneously. With this implementation, backtracking will be reduced to a minimum, restricted only to the break points interconnecting SLRDA. Therefore, a future scope of the idea involves implementation of a shape classifier incorporating the combination of an exploration and a navigation automaton.

POSITIONING OF THIS RESEARCH ON THE STATE OF THE ART

This study introduces a method related to the concept of adaptive linearity implemented by ADSLS, wherein, for example, irregular arcs can be actually straight when viewed in the proper range scale.

One of the features of this research was to emphasize algorithms without backtracking, improving the representation of SLRD and their parameters by an adaptive neighborhood. By the generalization of this work with the reusing of classic algorithms, as well as applying the known syntactic background, adaptivity can integrate other modern existing methods.

Taking into account the work (Neto, 2007); this research involves devices guided by rules with multilevel hierarchical adaptivity whose set of rules is variable presenting modifiable adaptive functions. Therefore results in an entire evolutionary potential of this work based on the reuse and generalizations of formalisms that have traditionally been used.

FINAL CONSIDERATIONS

Considering variable angles and minute errors of DSLS, to our knowledge, this is the first attempt to introduce AFA in dominant point detection. By traditional techniques, automaton would have to be implemented a-priori, with high level of complexity to treat errors that cause imprecise models or imprecise scale of DSLS in different angles.

Compared with other methods of digital line representation in dominant point detection, this work showed that the formalism presented in (Barros Neto, Junho 2011) incorporates the main advantages of the ADSLS formalism: *i*) Simplicity and relative ease of modeling and implementation, associated with high computational power; *ii*) Models are easy to understand, relatively simple to program and flexible to accept changes in their behavior; *iii*) Longer trajectories will dynamically increase the storage mapping memory used; *iv*) Storage memory required to map unknown environments may be reduced drastically by adaptive straight line segmentation of trajectories because just two grid coordinates are necessary to describe a specific digitized straight line path.

Models of DSLS strings were also presented, associated to the corresponding automaton with experiments that demonstrated the simplicity and efficiency of the method, allowing the use of traditional syntactic tooling. The expressive power of the ADSLS formalism incorporates parameters of DSLS such as angle, length and tolerances to represent adaptive regions of support. Basic shapes were used in the experiments that have immediate applications in some areas like robotics (Jokesch, Bdiwi, &Suchy, 2014).

REFERENCES

Abbasi, H., Olyaee, M., &Ghafari, H. R. (2013, May). Rectifying reverse polygonization of digital curves for dominant point detection. IJCSI International Journal of Computer Science Issues, 10

Barros Neto, L. C. de & Hirakawa, A. R. (2014, October). An approach by straight line segment adaptive techniques in robot navigation. IEEE Latin America Transactions, 12.

Barros Neto, L. de, Hirakawa, A., & Massola, A. (2011, October). An adaptive model applied to digital geometry to enhance segment straightness. Latin America Transactions, IEEE Latin America Transactions, 9.

Barros Neto, L. C. de (2011, Junho). Modelagem em geometria digital aprimorada por técnicas adaptativas de segmentos de retas (Doctoral Dissertation, Escola Politécnica da Universidade de São Paulo (USP)).

Bhowmick, P., & Bhattacharya, B. B. (2007, September). Fast polygonal approximation of digital curves using relaxed straightness properties. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29, 1590–1602.

Dinesh, R., & Guru, D. (2007). Finite automata inspired model for dominant point detection: A non-parametric approach. International Conference on Computing: Theory and Applications, p. 579-583.

Feschet, F. (2008, December). The lattice width and quasi straightness in digital spaces. ICPR 2008, 19th International Conference on Pattern Recognition. In Anais... (p. 1-4). Tampa, FL.

Freeman, H. (1970). Boundary encoding and processing. Picture Processing and Psychopictorics, 241-266. (B.S. Lipkin and A. Rosenfeld, editors, New York, Academic Press, 1970)

Gao, Y., & Leung, M. K. (2002, February). Human face profile recognition using attributed string. Pattern Recognition, 35(2), 353-360.

Jokesch, M., Bdiwi, M., & Suchy, J. (2014, October). Integration of vision/force robot control for transporting different shaped/colored objects from moving circular conveyor. In IEEE international symposium on robotic and sensors environments (ROSE) (p. 78-82).

Klette, R., & Rosenfeld, A. (2004). Digital geometry: geometric methods for digital picture analysis (Elsevier, Ed.). Morgan Kaufmann.

Largeteau-Skapin, G., & Andres, E. (2006). Two discrete Euclidean operations based on the scaling transform. In S. Verlag (Ed.), Anais... (Vol. 4245 LNCS, p. 41-52).

Szeged. Lebedev, V. I. (2004, July). The accuracy of the objects position measuring in an image. XXth ISPRS Congress. In Anais... (Vol. 35; Part B1, p. 45 -47). Istanbul.

Neto, J. J. (2007, Novembro.). A small survey of the evolution of adaptivity and adaptive technology. IEEE Latin America Transactions, 5(7), 496-505.

Prasad, D. K, & Quek, C. (2013, December). Comparison of error bounds for non-parametric dominant point detection. In 2013 9th international conference on Information, communications and signal processing (ICICS) (p. 1-5).

Prasad, D. K. (2013, August). PRO: A novel approach to precision and reliability optimization based dominant point detection. Journal of Optimization, 2013, 15.

Ramos, M. V. M., Neto, J. J. & Ítalo Santiago Vega. (2009). Linguagens Formais. Bookman.

Shwetha, D., & Ramya, S. (2014, August). Comparison of smoothing techniques and recognition methods for online kannada character recognition system. In IEEE international conference on advances in engineering & technology research (ICAETR - 2014).

Sousa, M. A. A., &Hirakawa, A. R. (2005). Robotic mapping and navigation in unknown environments using adaptive automata. In international conference on adaptive and natural computing algorithms, Coimbra, Portugal, 2005.

Veelaert, P. (2005). Uncertain geometry in computer vision. In Discrete Geometry for Computer Imagery, Lecture Notes in Computer Science. (Vol. 3429, p. 359-370). Springer Berlin Heidelberg.

Woo, D.-M., & Park, D.-C. (2011, May). Stereoscopic building reconstruction using high-resolution satellite image data. In computer and information science (ICIS), 2011 IEEE/ACIS 10th international conference on computer and information science (ICIS), (p. 194-198).

Wu, W.-Y. (2003). Dominant point detection using adaptive bending value. Image and Vision Computing, Elsevier, 21.

You, K. C., & Fu, K. S. (1979, June). A syntactic approach to shape recognition using attributed grammars. IEEE transactions on systems, man, and cybernetics, 9(6), 334–345.