DESIGN OF A TOOLBOX TO ASSIST THE CARTOON PRODUCTION

Joaquim Madeira[1], André Stork
Computer Graphics Center
Wilhelminenstr. 7
D–64283 Darmstadt
Germany
{jmadeira, stork}@igd.fhg.de

Abstract

The production of 2D animation films (i.e. cartoons) using traditional techniques is a labor– and cost–intensive process. If some of its tasks could be efficiently supported by appropriate computer tools, production costs would certainly decrease. A set of tools to assist the painting and inbetweening stages of the traditional cartoon production is currently being developed. The reasoning behind the toolbox structure and the fundamental aspects of the developed assisted painting approach are presented.

Keywords: computer–supported cartooning, image processing, assisted painting.

1 – Introduction

The production of 2D animation films (i.e. cartoons) using traditional techniques is a labor– and cost–intensive process. Considering the number of manually drawn and painted layers required just for a few minutes of cartoon film, the amount of time dedicated to these tasks turns out to be a crucial factor in the overall production process: the workload associated to inbetweening (i.e. drawing of intermediate layers) and layer painting represents about 60% of all labor required [4]. If the drawing of intermediate layers (i.e. inbetweens) and the layer painting could be sufficiently supported by computer tools, production costs would certainly decrease. Note that, taking into account the overall production costs, even relatively modest savings are enough to warrant the acquisition of computing equipment and software tools, as well as the training of the prospective users.

Another important aspect that should not be neglected is that to avoid radically eliminating existing practices computer–supported tools for cartoon production should be able to accept hand–drawn images as input. The cartoon designer may continue to use favorite tools to express his/her creativity, and the traditional creation process – with its rough and cleaned–up drawings –, which does not encourage the use of direct input devices (e.g. stylus and tablet), is not disturbed.

1. Grupo de Métodos e Sistemas Gráficos, Departamento de Matemática,
Universidade de Coimbra, Apartado 3008, P–3000 Coimbra, Portugal.
The stay at the Computer Graphics Center in Darmstadt is supported by the JNICT/CIENCIA scholarship BD/2205/92–IA.

A set of tools to assist the painting and inbetweening stages of the traditional cartoon production is currently being developed, having as input scanned hand–drawn layers. It allows the (semi–)automatic painting of a sequence of layers, having as a start–up the previously painted first layer in the sequence and using shape matching methods to identify corresponding regions and assign colors. A further, more ambitious step is the (semi–)automatic generation of inbetweens, based on two previously painted layers showing extreme or characteristic positions of a cartoon character. The need for appropriate data storage and management facilities has also to be stressed.

Following this introduction, the reasoning behind the toolbox structure and the fundamental aspects of the developed assisted painting approach are presented. Special emphasis is given to the preprocessing of the scanned images and to the methods and algorithms pertaining to assisted painting.

2 – Overall Structure of the Toolbox

The design of the toolbox structure was constrained by the requirement to accept hand–drawn images as input, which implies the need for a preprocessing stage to treat the digitized images and extract meaningful information to be used later. According to the design aims – (semi–)automatic image painting and inbetweening –, it must be possible to manually paint and structure in a simple way at least one reference image, which will then be used as start–up image(s) for the assisted painting process, and to represent image contours in an appropriate way to allow later the generation of intermediate images between two key ones.

Given the previous requirements the overall structure of the toolbox was divided into five modules, whose main features are:

- **Preprocessing** – the cleaning and closing of a scanned image, followed by the identification of the image regions and the extraction of region features (e.g. region contours) are done automatically.
- **Painting** – the coloring of a preprocessed image and a simple, hierarchic structuring of its contents are accomplished by the user.
- **Assisted Painting** – having a previously painted image as reference, the next image in the sequence is colored and structured (semi–)automatically using shape matching techniques based on the previously extracted region features.
- **Vectorizing** – as a preceding step to inbetweening, a piecewise approximation to the image contours is computed.
- **Inbetweening** – using the vector description of two key–images, additional intermediate images are (semi–)automatically generated.

The modules were designed to work independently, allowing high flexibility and parallel development and use. Nonetheless, they can (and are intended to) be grouped as a processing pipeline, which is depicted in Fig. 1. The modules' hierarchy and the information paths are easily recognizable.

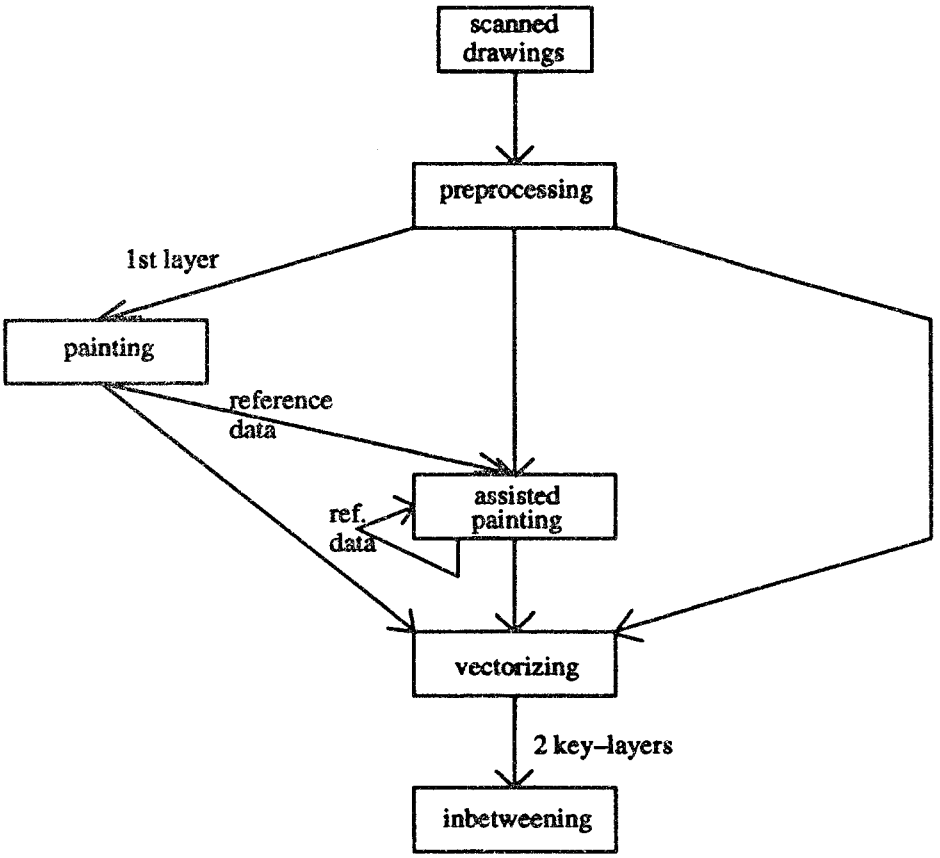

Figure 1 – Simplified representation of the processing pipeline

Based on the previous pipeline, prototype implementations of the preprocessing, painting and assisted painting modules were already accomplished and the vectorizing prototype is being implemented. The first aim of the prototypes is to allow, for each stage of the processing pipeline, an evaluation of various methods and algorithms, regarding their applicability, robustness and computational efficiency.

The user–interface for each prototype is built up under X–Window/OSF–Motif [24], using the user–interface management system TeleUSE [21]; the application code is structured in an object–oriented way and written in C++ [19].

3 – Preprocessing

In comparison to the traditional cartoon production, the preprocessing step is an extra stage that is introduced in the production. Therefore, and in order to reduce to a minimum

the amount of time associated to it, only methods which do not require any user intervention during their execution were considered for evaluation.

Usually the layers to be scanned contain black lines – corresponding to lines that will be represented as black ones in the final painted layer – and blue lines; region contours can be piecewise defined by black and blue line segments, the latter being not visible in the final painted layer. Since lines of different colors have to be identified, the layer has to be scanned in color mode to allow the recognition and separation of the different lines. If the scanning is performed with a too small resolution, lines melt together, new unwanted regions are created and also thin regions or parts of them might disappear. Experiments have shown that the minimum resolution needed is approximately 150 dpi; sometimes 300 dpi or more are needed to resolve thin structures.

A simple thresholding and weighted color component averaging algorithm is applied to the digitized color image and as many binary images as the number of line colors known to be present in the original image are created. The thresholds used are highly dependent on the color characteristics of the hand–drawn layers and should be set according to them prior to the scanning of each layer sequence. Otherwise, higher level color separation techniques – e.g. histogram–based methods or cluster analysis – have to be applied [7,8,15].

After generating the binary images the next task is to clean them of noise resulting eventually from the scanning process or already existing in the original image. That is, removing isolated foreground pixels or blobs from the background or setting isolated background pixels or blobs. This can be accomplished in several ways: one powerful approach seemed to be the mathematical morphology operations [8,9] – however, their limitations and side–effects (e.g. melting of small neighboring regions) prevent them from being successfully used in the cartoon context. The currently implemented alternative is based in the identification of connected components [8]; all regions whose seize is below a given threshold are removed, however they are shaped.

Since it was intended from the beginning to base the matching of regions in the assisted painting stage on image topology, it is necessary to extract the region neighborhood information from binary images. This can only be efficiently accomplished if the image lines were previously skeletonized (i.e. thinned). The thinned lines should maintain the connectedness of their original counterparts (to avoid creating line gaps during the thinning process), be placed near their medial axis (to provide a good basis for an accurate contour extraction and vectorization) and be one pixel thick (to ease the recognition of the neighborhood information). Skeletonizing can be accomplished in various ways; three different approaches were selected, implemented and evaluated.

The first approach, based on mathematical morphology operations [11], consists in successively applying a set of masks to the image to be skeletonized, until the thinned image

is obtained. It satisfies the stated goals, but it is quite sensitive to noise and its run–time inefficiency is a big disadvantage.

The idea for the second method is iterative peeling [12]: based on neighborhood criteria pixels which can be removed without a loss of connectedness are successively unset. However, depending on the sequence of removal operations, different results are obtained. Since the approximation to the medial axis is also poor, the method cannot be used to fully skeletonize the original binary image.

The algorithm which turned out to be the best computes the image skeletons from its distance transform [14]. In a few words: first the set of pixels associated to local maxima and saddle points of the distance map is selected; then, pixels corresponding to up–hill climbing paths in the distance map connecting elements of the first set are also selected. The pixels of the original binary image selected in the previous steps make up the image skeleton. Although redundancy checks are performed during the execution of the algorithm, the skeleton lines are not perfectly thin. A post–thinning step was implemented using just one peeling iteration, since the skeleton lines are at most two–pixels thick. A good approximation to the medial axis of the image lines is obtained; the execution time is independent of image line thickness.

The skeletonized image is then segmented (i.e. its regions are identified and their pixels appropriately labeled), line gaps are closed and region features (e.g. region contours) are extracted.

During the region segmentation step an identifier (i.e. a region label) is assigned to all pixels belonging to each region, which is later used to determine the region neighborhood relationships. Three different algorithms were implemented and evaluated: the traditional recursive flood fill algorithm [5], which is too slow due to the thousands of recursive calls; the sequential segmentation algorithm [8], with its two image passes, the first to assign labels, the second to resolve label equivalences; and Heckbert's flood fill algorithm [10], which again fills regions recursively, but avoids the recursive calls by the use of a local stack. The latter was judged to be the fastest algorithm, although its execution time is not independent of the image contents.

It has turned out to be efficient to close small gaps in the image lines in the skeletonized image and after the extraction of the endpoints. The gaps are closed using line oriented methods and various proximity criteria; for instance, the closing probability is higher the shorter the distance and the closer the angle of the lines to which the endpoints belong is to 180 degrees. An approach using mathematical morphology operations and masks of various seizes and patterns was evaluated at first; but its side–effects and the need for user intervention make it impossible to use in an automatic way.

The feature extraction step allows the extraction of information to be used in the succeeding processing stages. Region contour extraction is performed and region holes are also de-

tected; the region neighborhood information is determined by scanning around each line pixel.

Fig. 2 depicts the pipeline of preprocessing steps, the methods judged to be the best from the set of tests carried out are also shown.

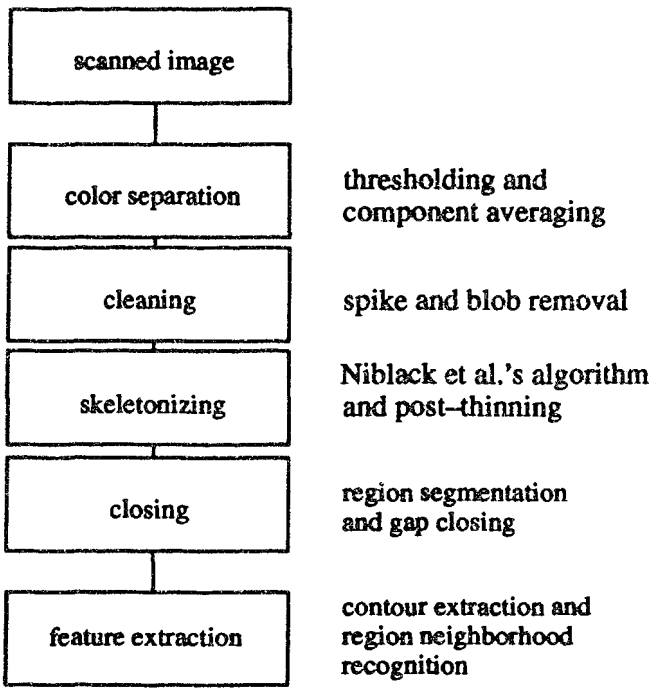

Figure 2 – The preprocessing pipeline

As an example of the performance of the preprocessing stage, an image containing 200.000 pixels is preprocessed in about 25 seconds. Clearly, there is still room for improvements and algorithm optimization, and this is currently being carried out.

4 – Painting

The need for a manual painting stage results from the fact that before the color information can be automatically propagated from image to image (i.e. cartoon layer to cartoon layer) in the assisted painting module, it has first to be established in (at least) the first image of any sequence.

The painting environment is a highly interactive one, where the user disposes of some of the usual coloring and editing facilities. Given the segmentation step accomplished in the preprocessing stage, the coloring of each image region is reduced (internally) to a manipulation of the color table, which considerably speeds up the interactive painting process.

Simultaneously, and with almost no extra effort on the part of the user, the image is structured in objects. This is easily accomplished by letting the user indicate that he/she wants

to start the definition of an object (i.e. a cartoon character or parts of it); until it is explicitly stated that the current object is fully defined, attributing a color to a region also signifies that it becomes part of the object being defined. At the end of painting, and in order to facilitate later the matching of image regions, an object region adjacency matrix is computed for each image object, using the global neighborhood information.

The painting environment also presents to the user a comfortable interactive tool for closing large line gaps, which could not have been closed with the automatic gap–closing facilities of the preprocessing stage; again the closing is based on the endpoint information associated to each image region.

Note that any of the subsequent layers in the cartoon sequence can either be painted with the painting or with the assisted painting environments. The intention is, however, to use the former to manually color only the first layer in a sequence, and then use the latter to color the remaining ones.

5 – Assisted Painting

The previously presented preprocessing and painting modules are the basis for (semi–) automatic image coloring, whose aim is to propagate the color information of a previously painted image – the reference – to its successor image in a sequence. The latter is then used as reference to paint the its successor image and the process repeats itself as long as no too strong deformations or changes in image topology (e.g. in the number of image regions) occur, or until the end of the image sequence is reached.

From the beginning it was intended to base the region matching process on appropriate descriptions of region shape, which should be (relatively) invariant regarding affine transformations and flexible enough to allow the identification of corresponding regions even when some contour deformations occur. The questions were then how to best define the shape of a region, how to describe it efficiently, and how to compare two such descriptions to obtain a measure of the similarity between their corresponding regions. The decision was taken to use pixel–based, contour–oriented shape descriptions; both global shape descriptions – which comprise all contour pixels – and local ones – which comprise a small sub–set of contour pixels (i.e. the contour landmarks) – were considered.

Global shape descriptions based on the direction of the tangent and curvature vectors at each contour pixel were implemented. In both cases a string description is used to code the shape information, string comparisons being then performed to evaluate the dissimilarity of their respective regions [13,22].

A local approach was also implemented: for each region contour, the set of its landmarks – i.e. points of high curvature determining the contour shape – is determined [20], then a piecewise approach to contour comparison using sphericity [1] – i.e. a measure of similarity between triangles – is taken, to compare successive groups of three contour landmarks; a

measure of similarity for the whole contours is afterwards computed using an appropriate dynamic programming algorithm [1].

From the set of tests carried out, one of the global shape description methods was judge to be the best: for each region, determine the direction of the curvature vector at each contour pixel using the Sobel operator [7,13,15] and code this shape information as a string; a dissimilarity measure between any two regions is computed by comparing their string descriptions [13,22].

The overall matching process can be described as follows:

First the user selects a region in the reference image belonging to the object he/she wants to paint, and its corresponding region in the image to be painted; this is done to avoid the costly search – in an automatic way – of two corresponding regions to start the matching process. Then the matching process is started.

Using the neighborhood information of the two corresponding regions, two sets of matching candidates – one for each image – are determined; a dissimilarity measure is then computed for every possible pair of matching candidates using one of the previously mentioned algorithms. The use of heuristics reduces the number of dissimilarity values that have actually to be computed; for instance, for a pair of candidate regions whose shape description s are judged to be too different a maximum dissimilarity value is automatically assigned.

After obtaining a dissimilarity value for each pair of matching candidates the final assignment of regions between the two images can be performed. This is accomplished by formulating the region assignment a linear programming problem, where the sum of the dissimilarity values between each pair of assigned regions has to be minimized. Such a problem is in reality a particular transportation problem , which is solved using the corresponding version of the simplex method [6].

Using each pair of regions assigned in the previous step the process proceeds until all assignable regions are matched, or no more assignments can be performed – i.e. the remaining regions cannot be matched since their shape is too different and/or they do not belong to corresponding region neighborhoods. Clearly, when there is an increase in the number of regions in the image to be painted, no color information can be assigned to the new regions that have *just* appeared; their colors have to be interactively assigned by the user. As in the painting stage, the user disposes of a set of coloring and editing facilities, to color such regions and to correct errors that might occur.

In comparison to the coloring methods of some commercial systems, the proposed approach needs less user actions and is relatively robust regarding image transformations and deformations (although limited by the use of heuristics). The execution time is also small enough to be competitive with purely manual approaches; a balance was stricken between automatization, correctness of results, need for user intervention and execution time.

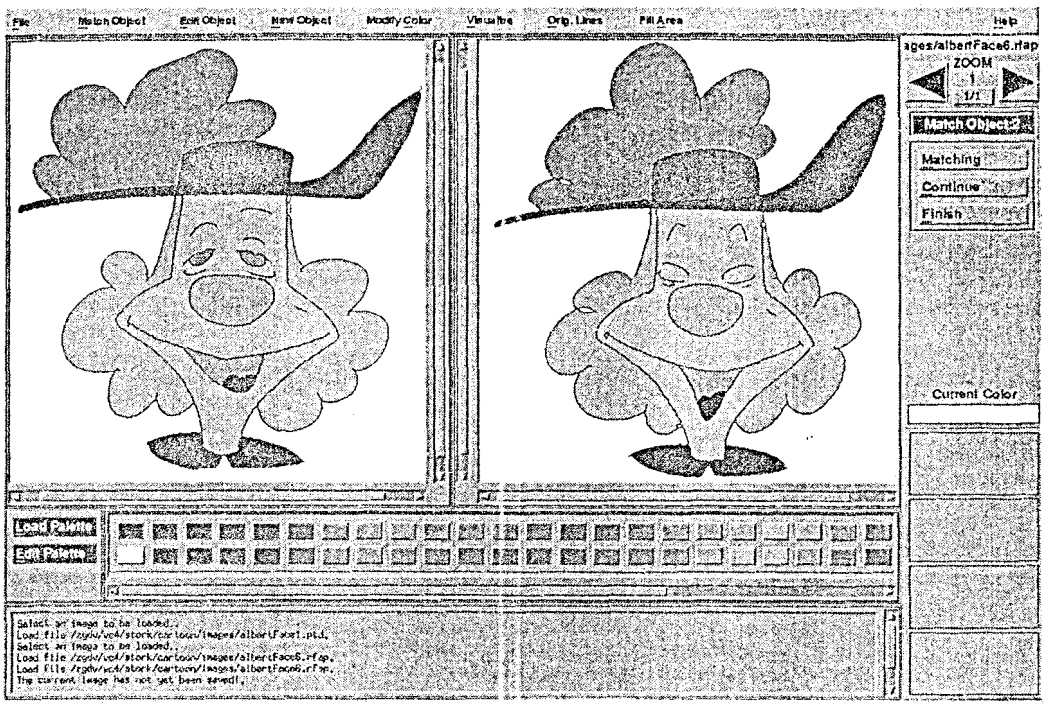

Figure 3 – The assisted painting environment: the reference image (left) and the automatically painted image (right).

For the example depicted in Fig. 3, the assisted painted step takes about 12 seconds, on a SUN SPARCstation 10, after the assignment by the user of the two regions that initiate the process; the image sizes are 516x532 and 504x544. One should note the deformations of individual regions and the appearance/disappearance of regions (e.g. the eyes are closed on the right image). In spite of these effects the result is fully correct.

Clearly, the method judged to be the best is not always confusion–free, and some improvements are being investigated. Nevertheless, the framework of the assisted painting approach is valid and any methods conducing to improved dissimilarity measures between regions can easily be integrated in the assisted painting environment.

6 – Further Work

The vectorizing module of the toolbox is currently being implemented. In a similar way to preprocessing, digitized images are vectorized in a fully automatic way. Several approaches are being evaluated: namely, approximation of the image skeletons using piecewise Bézier curves [16], extraction of line thickness using the distance transformed original image, and representation of the topology of the vectorized image using a data structure based on Weiler's Face–Edge structure [23].

After this module is completed work will start on the implementation of the inbetweening prototype. In spite of the limitations and problems inherent to inbetweening [2,4], the approach taken in the last years by Sederberg [17,18] seems to be promising.

7 – Conclusion

The reasoning behind the design of the toolbox to assist the cartoon production currently being developed was presented. Special emphasis was given to the methods and algorithms associated to the preprocessing of the digitized images and to the developed approach to assisted painting.

Other application areas where the developed ideas could also be applied are: coloring of black and white films, object recognition, and graphical editing – an approach to pattern editing using a shape matching technique is presented in [3].

8 – Acknowledgements

The authors thank Professor Dr. José L. Encarnação and Dr. Markus Groß for their guidance and help, as well as for all the opportunities given.

The authors are also indebted to former and current students, especially Renato de Amorim, Paulo Bernardes, Gino Brunetti, Steffen Kremser, Jürgen Preuss and Michael Schmidt, who have helped implement parts of the toolbox.

9 – References

[1] – Nirwan Ansari and Edward J. Delp, Partial Shape Recognition: A Landmark–Based Approach, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 12, N. 5, p. 470–483, May 1990.

[2] – Edwin Catmull, The Problems of Computer–Assisted Animation, *ACM Computer Graphics (Proc. SIGGRAPH'78)*, Vol. 12, N. 3, p. 348–353, August 1978.

[3] – A. Della Ventura, P. Ongaro and R. Schettini, Pictorial Editing by Shape Matching Techniques, *Computer Graphics Forum*, Vol. 12, N. 2, p. 111–122, 1993.

[4] – Charles X. Durand, The "TOON" Project: Requirements for a Computerized 2D Animation System, *Computers & Graphics*, Vol. 15, N. 2, p. 285–293, 1991.

[5] – James Foley, Andries van Dam, Steven Feiner and John Hughes, *Computer Graphics – Principles and Practice*, 2nd Edition, Addison–Wesley, 1990.

[6] – Saul I. Gass, *Linear Programming – Methods and Applications*, 4th Edition, McGraw–Hill, 1975.

[7] – Rafael C. Gonzalez and Paul Wintz, *Digital Image Processing*, 2nd Edition, Addison–Wesley, 1987.

[8] – Robert M. Haralick and Linda G. Shapiro, *Computer and Robot Vision*, Vol. 1, Addison–Wesley, 1992.

[9] – Robert M. Haralick, Stanley R. Sternberg and Xinhua Zhuang, Image Analysis Using Mathematical Morphology, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 9, N. 4, p. 532–550, July 1987.

[10] – Paul S. Heckbert, A Seed Fill Algorithm, in *Graphics Gems*, Andrew S. Glassner (Ed.), p. 275–277, Academic Press, 1990.

[11] – Ben–Kwei Jang and Roland T. Chin, Analysis of Thinning Algorithms Using Mathematical Morphology, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 12, N. 6, p. 541–551, June 1990.

[12] – Louisa Lam, Seong–Whan Lee and Ching Y. Suen, Thinning Methodologies – A Comprehensive Survey, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 14, N. 9, p. 869–885, September 1992.

[13] – Hong–Chih Lin and M.D. Srinath, A String Descriptor for Matching Partial Shapes, in *Computer Vision and Image Processing*, L. Shapiro and A. Rosenfeld (Eds.), p. 575–592, Academic Press, 1992.

[14] – C. Wayne Niblack, Phillip B. Gibbons and David W. Capson, Generating Skeletons and Centerlines from the Distance Transform, *CVGIP: Graphical Models and Image Processing*, Vol. 54, N. 5, p. 420–437, September 1992.

[15] – William K. Pratt, *Digital Image Processing*, John Wiley, 1991.

[16] – Philip J. Schneider, An Algorithm for Automatically Fitting Digitized Curves, in *Graphics Gems*, Andrew S. Glassner (Ed.), p. 612–625, Academic Press, 1990.

[17] – Thomas W. Sederberg and Eugene Greenwood, A Physically Based Approach to 2–D Shape Blending, *ACM Computer Graphics (Proc. SIGGRAPH'92)*, Vol. 26, N. 2, p.25–34, July 1992.

[18] – Thomas W. Sederberg, Peisheng Gao, Guojin Wang and Hong Mu, 2–D Shape Blending: An Intrinsic Solution to the Vertex Path Problem, *ACM Computer Graphics (Proc. SIGGRAPH'93)*, Annual Conference Series, p.15–18, 1992.

[19] – Bjarne Stroustrup, *The C++ Programming Language*, Addison–Wesley, 1986.

[20] – Cho–Huak Teh and Roland T. Chin, On the Detection of Dominant Points on Digital Curves, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 11, N. 8, p. 859–872, March 1989.

[21] – Telesoft Inc., *TeleUSE System Manual*, 1992.

[22] – Robert A. Wagner and Michael J. Fischer, The String–to–String Correction Problem, *Journal of the ACM*, Vol. 21, N. 1, p. 168–173, January 1974.

[23] – Kevin Weiler, Edge–Based Data Structures for Solid Modeling in Curved–Surface Environments, *IEEE Computer Graphics and Applications*, Vol. 5, N. 1, p. 21–40, January 1985.

[24] – Douglas A. Young, *The X Window System – Programming and Applications with Xt – OSF/Motif Edition*, Prentice–Hall, 1990.