A METHOD TO CONVERT FROM HAND-DRAWINGS INTO HIERARCHICAL DATA STRUCTURES

Kazushi Mukaiyama

Kyoto City University of Arts, Graduate School of Art, Kyoto, Japan
13-6 Kutsukakecho, Ohe, Nishikyo-ku, Kyoto, 610-1197 Japan

This paper describes the process of converting hand-drawn images into combinatorial data structures. For this paper, I created a virtual sketch board, using a computer screen as a canvas, and recorded the users' strokes as input data. First, the program categorises strokes, learning using a kind of pregnanz Gestalt algorithm from that data. The program then creates a hierarchical bone structure from these strokes. After that, the method adds thickness to them to define ellipse segments as the least basic elements. Finally, we can get a bold structure from it, which has ellipse segments as basic elements of the hierarchy tree.

1 REASON TO CONVERT

When visual artists draw or paint, they use the properties of space, shape and perspective to create an image. However, the ability to perceive an object mentally is not a result of visual perception alone. According to Kusahara [7], a blind painter will develop his own method of spatial recognition, not based on visual but tactile perception. For example, when drawing a tin can, an individual employing visual perception might draw a wide ellipse, which represents the top of the can–due to recognition of spatial perspective. The blind painter, however, draws the can akin to an expansion plan because he does not recognize visual perspective. (Fig.1) Although the construction of the blind painter might be incorrect for the visual artist, it is NOT important for the artist to represent the world as an exact one for one representation; the representation can be more abstract. Many painters utilize the capacity to formulate a mental vision of an object to create their works; to some extent, they draw blind. This method of perception does not make use of one-to-one copying, but the actuality that we as humans skew what we see in direction of what we know [9]. It means that the viewer who perceives the work does not trace the shape of objects but reconstructs it in his/her mind; nevertheless, we need to understand–at some level–an object before it can be reproduced. This raises the question; **what is the minimum amount of graphical information that is required in order to understand drawn objects?** Relative size and location are the most important elements of both visual and tactual modalities. There are however, other attributes to consider such as colour, texture, etc... Can the world be understood by reducing it to its constitutive elements? To represent constitutive elements effectively in a computer, we may represent combinations of LEGO-like elements as data structures. In order to use a set of elements, I define an ellipse as most basic element. A curved shape such as an ellipse does not lend itself to represent an object with sharp edges. Ellipses however,

are better suited at representing natural shapes such as those drawn by hand. Of course, there are several other ways to represent shapes by combining primitives, for example Biederman's geon theory [1], or by Marr's cones [8].

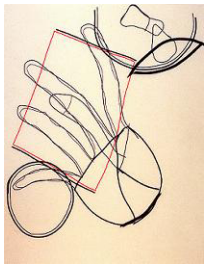

Figure 1: Drinking a can of coffee, 1996. material, paper and Letraline [10]

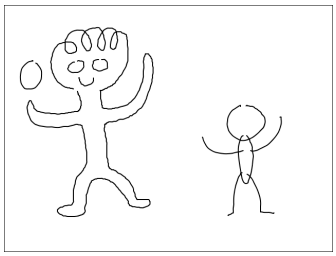

Figure 2: input sample

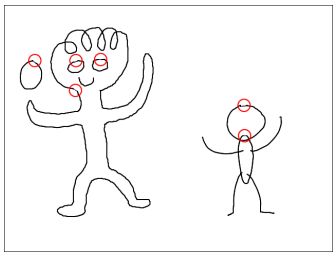

Figure 3: Evaluation open/close of stroke

2 INPUT DATA

In the case of an image, there are many ways to input it into a computer. One way is to capture a discrete image with scanner or video camera. It is possible for scanned pictures to be processed so as to pick up the outline of an image. This method has the advantage that it does not include added stress for a user. It has, however, the disadvantage that it is problematic to separate intersecting lines or to pick out objects from the background– depending on the ambient light. In addition, the computer must spend a relatively long time computing if an image is of high resolution. Therefore, I choose a method that tracks and records strokes made by a user upon the virtual sketch board. (Fig.2) This decision was made because I am more interested in the cognitive aspects of the process than the process of capturing the image. This is a model of visual processing, not that of the eye but of the brain. In this manner, we can deal with each line from start to end, analyzing each stroke independently of the other.

3 FIRST STEP: GROUPING STROKES

The first step is to pre-process the input data. The program executes the following operations.

3.1 Classification of strokes as "open" or "close"

Cohen [2] referred to children's drawings when he began to implement his artificial painter– named AARON. A young child who has a pen first time in his/her life usually draws only scribbles. Before three years of age, however, he/she learns the feedback between pen and his/her body. The first characteristic mark, which appears at this time, is a "circle". Therefore, it is important that the start and end points are connected. I define all input

tracking strokes that a user draws on screen as G. I also define the individual strokes as S. S has points $P(x; y)$ on XY orthogonal axis. Then, S is represented as one line by connecting P with order, $S = \{Pi; i = 1, 2, 3 \ldots n\}$. The program evaluates the open/close attribute of element S as follows;

$$|Pn - P0| < \frac{\sqrt{(x(max)-x(min))^2+(y(max)-y(min))^2}}{4.0} \quad (1)$$

$x(max); x(min); y(max); y(min)$ are the maximum/minimum location number of $P \in S$, $||$ means distance. If the above condition is true then program closes S, otherwise it keeps the program keeps the stroke open. (Fig.3)

3.2 Separating at an intersecting part

In the next step, the program evaluates whether a stroke contains any intersecting parts. If a stroke has an intersecting part, the program separates out that part as in Figure 4. This results in obtaining the correct bone structure from a twist-closed-stroke during skeletonisation, and has the positive aspect that the basic structure does not change even if it is separated. After separating, we get a new G, which has no intersecting strokes and is organized by size, with the largest first. (Fig.5)

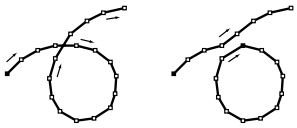

Figure 4: Separating intersection

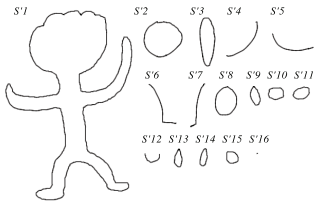

Figure 5: Separated sample

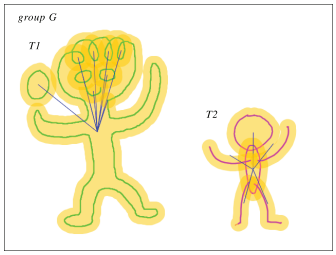

Figure 6: Result of grouping

3.3 Grouping in touch and neighborhood

In the third step, the program defines $A(S)$, an occupied area of S, dealing with all of S as a closed-stroke. If the size of $A(Sb) \cap A(Sa)(a < b)$ is not 0.0 then Sb becomes a child of Sa. This means that it groups elements S in a hierarchical tree, whether the elements touch each other or not. When this process is complete, the program obtains T as the resulting hierarchical structure. Additionally, in one of the pregnanz Gestalt algorithms, shapes that are close enough each other have a tendency to be identified as belonging to the same group. Therefore, the program evaluates again each T; using the expanded optional value w, a new area is derived from $A(S)$ added $E(S)$. This means that if the size of $A(S \in Tb) + E(S \in Tb) \cap A(S \in Ta) + E(S \in Ta)(a < b)$ is not 0.0 then $S \in Tb$ becomes a child of $S \in Ta$. Although the value w depends on the size of stroke, I set $w = 30.0$ because of the canvas size used–width 640 pixel and height 480 pixel. In this

second evaluation, Ta contains Tb. At this time, the tree of Tb is reconstructed to set $S \in Tb$, which uses the evaluation as this new root. Then finally, the group that has these trees is

$$G = \{Tj; S \in T\} \tag{2}$$

For example, group G is Fig.6. Each hierarchical element T consists of only one segment.

4 SECOND STEP: CONVERTING EACH STROKE INTO ELLIPSE SEGMENTS

In the second step, the program converts elements T into the sorted group G. $S \in T$ continue to track the strokes and ensure they belong to the correct node. The program picks skeletons from S. Skeletonisation within the context of this project denotes a method of obtaining a set of bones from a contour shape. There are several basic algorithms for skeletonisation; Hilditch's [4] method that is well known, obtains thin lines by deleting neighbouring pixels on a discrete image. These algorithms, however, can only obtain geometric data about a given bone, and are unable to understand the relationships between bones. Therefore, I would like to propose a new method; by combining both Igarashi's [6] method of skeletonisation with Delaunay process of triangulation and Hontani's [5] hierarchical process to obtain a hierarchical structure we can obtain faster and more relative results. By utilising this combination of both processes, we can produce not only a hierarchical structure but it is possible to obtain faster skeletonisation. Element S has two types; open-stroke or close-stroke. An open-stroke can be dealt with as bone. In this case, the program jumps to section 4.6 in order to remove the distortion of an open-stroke, and converts the open-stroke to ellipse segments directly. Therefore, the following sections up to 4.6 concern themselves exclusively with closed-strokes. From now on, closed-stroke images will be referred to as contours.

4.1 Delaunay Triangulation

Delaunay triangulation is a method that allows a contour to be divided into triangles that will have duality with a Voronoi polygon [3]. The total sum of smallest angles of all triangles is the maxim. This means that this is an ideal triangulation, which does not tend to contain any squashed triangles. For this reason, it is useful to obtain these triangles by using techniques similar to those used in 3d graphics to produce polygons. We can get triangles like Fig.7 from a contour, by segmenting its stroke into many same sized pieces. According to Igarashi, these triangles can be classified into three groups by analysing the triangles visual appearance. The first is referred to as terminal triangle; having two edges as part of the contour. The second is referred to as sleeve triangle; having one edge as part of the contour. The third is referred to as junction triangle; having no edge as part of the contour. Using this classification, it becomes apparent that a junction triangle will always be positioned at the junction of a contour. Therefore, this type of triangle can be used to correctly identify a bone junction.

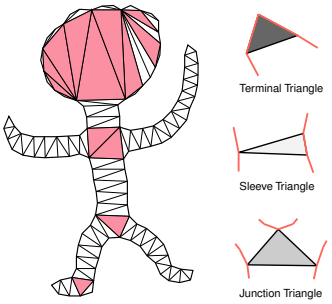

Figure 7: Split a contour shape with Delaunay triangulation

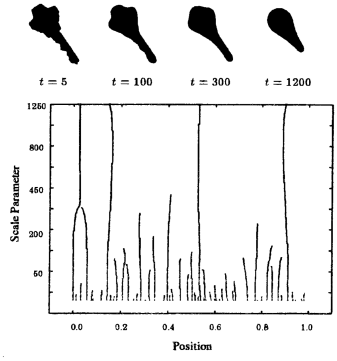

Figure 8: Scale space of curvature max points[[12], p.4, fig.3]

4.2 Scale space

Scale space is metric space i.e. it is two-dimensional. Located on the horizontal x-axis are the positions on a contour, and located on the vertical y-axis is smoothing power. To attain a smooth contour [11], the algorithm deletes unnecessary max curvature points, in much, the same way a painter will identify characteristics of a shape by only looking at the more pronounced features i.e. the painter ignores local details. Fig.8 plots curvature max points in scale space. Some curvature max points merge while others dissolve with increased smoothing power. Hontani's concept attempts to derive a hierarchical structure of the skeleton, by checking whether a curvature max point has dissolved or merged. As curvature max points are at limits of a bump in a contour, they correspond with a bone in the skeleton on each smooth grade. Therefore, bones in the skeleton will disappear when the corresponding point disappears. Meaning that if a max point dissolves the corresponding bone is a child. See Fig.9. If two max points merge then the corresponding bone of the merged point is a parent. See Fig.10.

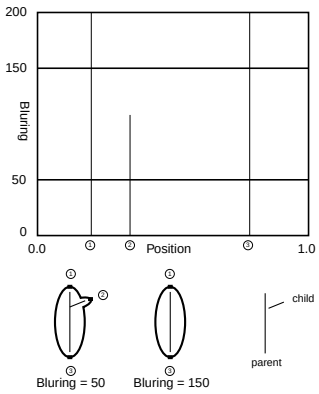

Figure 9: Dissolve a curvature max point

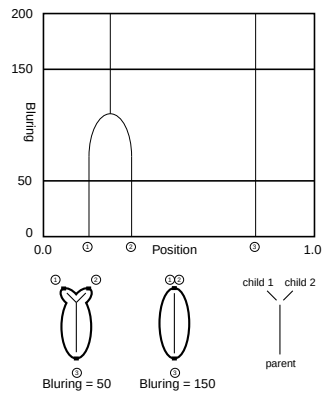

Figure 10: Merge two curvature max points

4.3 Relationship of splitting lines and scale space

It becomes apparent quickly that contour bumps, which are able to merge with one another, have almost the same size. (Fig.11) Using this principle, it is possible to order the three edges of a junction triangle by size. This is achieved by finding the sum of the combined areas of the triangles contained within the bump along each edge. Next, the program deletes the edge that has the biggest bump area. In considering the last two edges, the program determines if bumps of two edges are almost of the same size. If this is true then the program will keep both of the edges because the edges in question have the same status as a merging curvature max point. Else, it deletes the edge that has the second biggest bump area (Fig.12). A contour may have many extra splitting lines, due to the presence of small bumps. These splitting lines are to be deleted as they are considered noise. To do so, the program deletes all of the edges with small bumps. As a result, we obtain the correct splitting lines for a given contour.

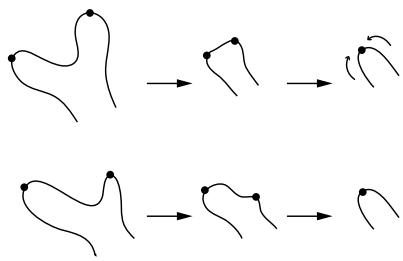

Figure 11: Difference between dissolving and merging in shape

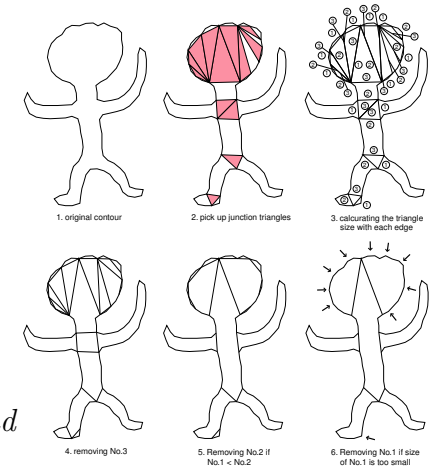

Figure 12: Pick up splitting lines

4.4 Splitting a contour and getting its root

Itemisation of the vertices of a triangle is processed in a counter-clockwise direction; this gives direction to splitting lines. (Fig.13) The contour is divided into segments by the splitting lines. One splitting line always produces two parts, and both parts are added to an array recursively. Yet, to produce a hierarchical tree, we must decide which contour is a parent. If a segment has the same direction of the splitting line it will always be the parent. When complete we produce a hierarchical tree of all the contours parts. (Fig.14)

4.5 Picking up skeleton bones

As described in section 4.1, triangles are classified into three types. Additionally, junction triangles are sub categorised into three types. With the remaining edges as splitting lines. Therefore, triangles are classified into five types in totality. The program makes four types

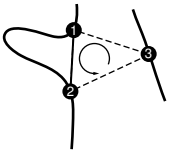

Figure 13: Order of vertexes in a triangle

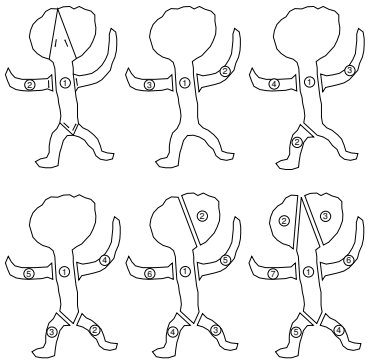

Figure 14: Splitting steps

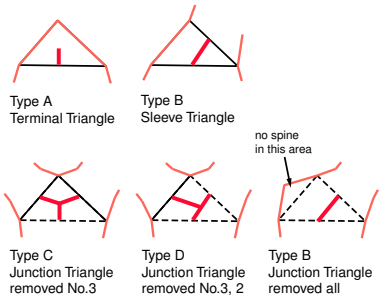

Figure 15: Types of spine

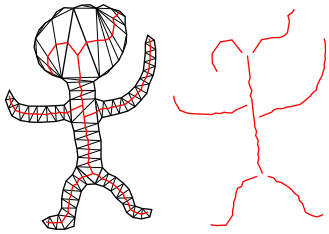

Figure 16: Generated skeletons with hierarchy

of spine depending on each triangle type like Fig.15. These spines become bones in each separated segment of a contour. However, to remove noise form the spine the program does not include small bumps when creating the spine. The program then joins a parent and child spine that may touch each other. This results in each segment containing a single bone; the bone is created from all the spines previously found in each segment. Fig.16

4.6 Removing distortion of bones

Type D, which is one of the four types of spines in Fig.15, is formed in order to remove large distortions, which are caused by type C junction triangles. However, by just obtaining bones directly from the spines of these triangles we cannot remove all the distortion, even using type D. This becomes problematic as to obtain the final ellipse segment we require a simple smooth line. Therefore, the program obtains a common area; this is achieved by examining the area at which a bone expands in some value while limiting this expansion to the area of the contour. A new bone is created by deleting points of an existing bone. The new bone should be created within the area of the contour and less than a given angle as in Fig.17. As a result, most of the segments contain only one bone. However, as a segment that has a heavy curve will contain only a single bone, that bone will be made up of several spines.

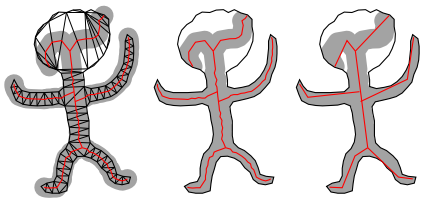

Figure 17: Remove distortion of skeletons

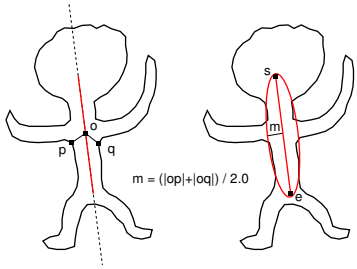

Figure 18: How to get thickness of an ellipse segment

4.7 Ellipse segmentation

4.7 Ellipse segmentation A single bone will create one-ellipse segment. The program obtains the two minimum distances between the centre of a bone and points on each side. Then the muscle is the half-length of the total of the two distances. (Fig.18) Furthermore, a bone can be represented by the following parameters;

$$\beta = \begin{pmatrix} StartPoint : s(x1, y1), \\ EndPoint : e(x2, y2), \\ Muscle : m(m \geq 0) \end{pmatrix} \tag{3}$$

$m = 0$ in an open stroke.

4.8 Reconstructing hierarchical tree

The separate parts of a contour have several ellipse segments within the same hierarchy. In this case, it needs be reconstructed as in Fig.19. In essence, the program reconstructs the several ellipse segments parent-child relationship by analyzing the nearest adjoining parent and making the current ellipse a child for that ellipse. This process is recursive and is repeated for all unsorted bones. So that the original parent becomes the root parent. Fig.20

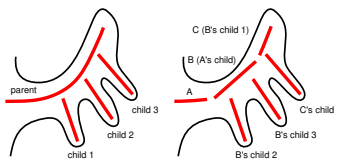

Figure 19: Reconstruct the hierarchical information

Figure 20: Ellipse segmentation

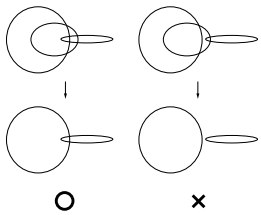

Figure 21: Optimize unnecessary segments

4.9 Optimizing unnecessary segments

Although at this stage the result is satisfactory, there are still unnecessary segments that need to be removed. In Fig.21, the target segment does not affect the overall shape of the contour. These unnecessary segments should be deleted to achieve the simplest result. Concretely, when the area size of a segment is expressed as $A(\beta)$, if

$$\begin{array}{l} A(\beta) \neq 0.0 \\ A(\beta(parent)) \neq 0.0 \\ \frac{A(\beta)}{A(\beta(parent))} - 1.0 < 2.0 \end{array} \tag{4}$$

then β is the target to remove. But, if $A(\beta(parent)) \cup A(\beta(child)) = 0.0$, then it is not removed, because $\beta(parent)$ and $\beta(child)$ have been separated. (Fig.21)

5 FINAL STEP: MERGING ALL SEGMENTS

Now we define the converted elements $S \in T$ as the hierarchical structure t. We need to flatten them all in T level. Because, for example, the figure $T1$ which is drawn separately: head($= \beta 1$), body($= \beta 2$), right arm($= \beta 3$), left arm($= \beta 4$), is strutted in

$$\begin{array}{l} T1 = \{t1, \{t2, \{t3, t4\}\}\} \\ t1 = \{\beta 1\}, t2 = \{\beta 2\}, t3 = \{\beta 3\}, t4 = \{\beta 4\} \end{array} \tag{5}$$

However, figure $T2$ which draws both head and body at once and each arm separately, is strutted in

$$\begin{array}{l} T2 = \{t1, \{t2, t3\}\} \\ t1 = \{\beta 1, \{\beta 2\}\}, t2 = \{\beta 3\}, t3 = \{\beta 4\} \end{array} \tag{6}$$

Both hierarchical structures are different here. But they should be identified as $T1 \equiv T2$.

To do so, with t and $t(parent)$, the program get 2 points of the nearest distance between all points s, e of $\beta \in t$ and all center points $\beta \in t(parent)$. The nearest point in t is $p(nearest)$, the segment which has the nearest center point is $\beta p(nearest)$. From them, t is reconstructed with the new root($= \beta(root)$) decided by the new starting point, $p(nearest)$. And t is linked at $\beta p(nearest)$. Therefore, in $T1$,

$$\begin{array}{l} t2 \Rightarrow (\beta(root) = \beta 2, \beta p(nearest) = \beta 1) \\ t3 \Rightarrow (\beta(root) = \beta 3, \beta p(nearest) = \beta 2) \\ t4 \Rightarrow (\beta(root) = \beta 4, \beta p(nearest) = \beta 2) \\ t1 = \{\beta 1\} \Rightarrow t1 = \{\beta 1, \{\beta 2, \{\beta 3, \beta 4\}\}\} \end{array} \tag{7}$$

and in $T2$,

$$\begin{array}{l} t2 \Rightarrow (\beta(root) = \beta 3, \beta p(nearest) = \beta 2) \\ t3 \Rightarrow (\beta(root) = \beta 4, \beta p(nearest) = \beta 2) \\ t1 = \{\beta 1, \{\beta 2\}\} \Rightarrow t1 = \{\beta 1, \{\beta 2, \{\beta 3, \beta 4\}\}\} \end{array} \tag{8}$$

then, $T1 \equiv T2$ consists after casting as $T1 = t1 \in T1, T2 = t1 \in T2$. Furthermore, we can get T, final hierarchical segments. The program applies to every T in group G.

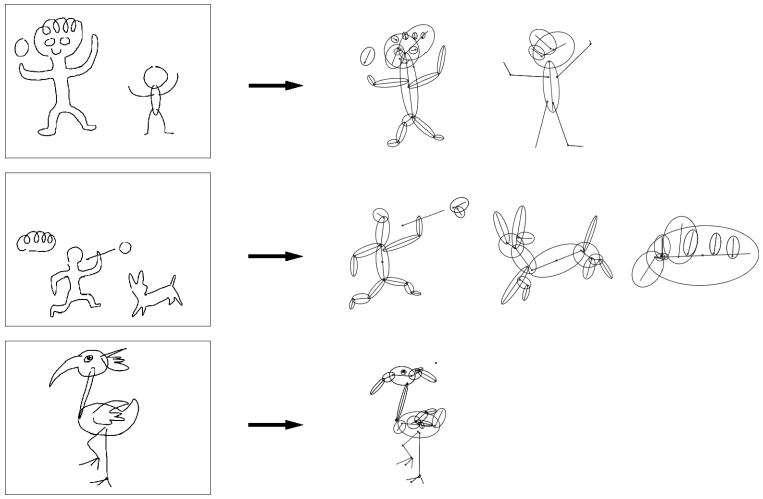

Figure 22: Result 1

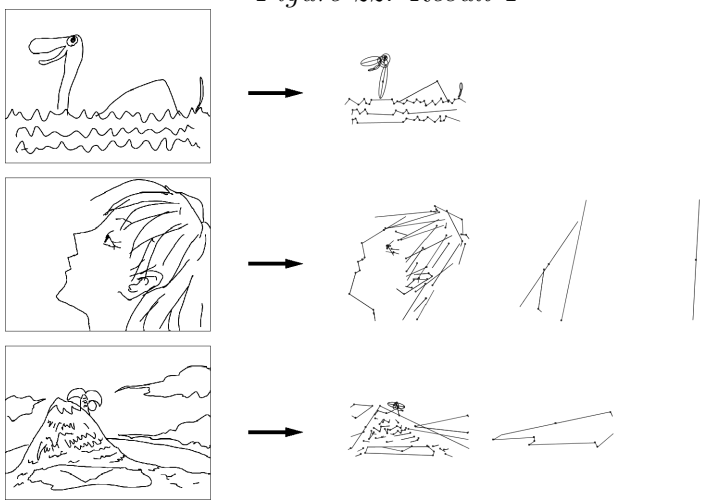

Figure 23: Result 2

Fig.22 and Fig.23 show the results of various experiments. Most single stroke drawings appear to become what is expected. However, apparently more complex drawings, created with many small strokes do not appear as clear. Because each stroke was dealt with not a single close-stroke but as open-strokes. To avoid this problem, it is possible to use the outline of a discrete image. However, I hesitate to do because it is quite different to understand an image drawn beforehand than a drawing created through a known sequence of events. The latter case means that people can anticipate an image even if the drawing is not finished. It is not unnatural if I recognize that this program is stubborn. I believe the next logical evolution of this project is to consider understanding the whole image at once in the first step of the program.

7 CONCLUSION AND NEXT STAGE

In this paper, I have demonstrated how hand-drawn images can be converted into hierarchical ellipse segments. The program can not only group simple contours, but also figures that consist of multiple separate parts, like human-faces, referring to a kind of psychology cognitions. It is natural for the computers to process hierarchical patterns. In the next stage, I would like to add the talent of imagination to the association of memory, using a self-organized map, with which the program can autonomously create a new figure, combining, for example, a giraffe's head and an elephant's body.

Acknowledgements

I would like to thank the following people so much for giving me many opinions and suggestions. Professor Tsuneo Nakai and Professor Sin Watanebe, Kyoto City University of Arts, and Assistant Professor Yoshiaki Mima, Hakodate Future University, and Professor Mauri Kaipainen, University of Art and Design Helsinki UIAH. Associate Professor H. Hontani answered me about scale space. Associate Professor H. Fujiwara advised me about numeric computing in many situations. John Evans BA (Wales) who was dedicated in correcting my papers English. And Professor Emeritus H. Cohen who discussed with me the artificial intelligence painter. I would like to thank them all very much.

References

[1] I. Biederman. *Psychological Review*, 94:115–147, 1987.

[2] H. Cohen. http://crca.ucsd.edu/ hcohen/. Author of the artificial intelligence painter, AARON.

[3] M. de Berg, M. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin Heidelberg, Germany, 1997.

[4] C. Hilditch. Linear skeleton from square cupboards. *Machine Intelligence*, 6:403–420, 1969.

[5] H. Hontani, M. Nakano, and K. Deguchi. Contour figure segmentation using multiresolution skeletons. *IPSJ JOURNAL*, 40(7), 1999.

[6] Matsuoka S. Igarashi, T. and H. Tanaka. Teddy: A sketching interface for 3d freeform design. *Siggraph99 Proceedings*, pages 409–416, 1999.

[7] M. Kusahara. Inspiring and sharing images between visual and tactual sense. *IPSJ SIG Notes, Computer Vision and Image Media*, (124):81–84, 2000.

[8] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and Company, San Francisco, 1982.

[9] P. McCorduck. *AARON's CODE*. W. H. Freeman and Company, New York, 1991.

[10] T. Mitsushima. http://www.nextftp.com/museum-access-view/atelier.html. The blind artist.

[11] F. Mokhtarian and A.K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.

[12] M. Nakano, H. Hontani, and K. Deguchi. A multi-resolution skeleton analysis for hierarchical description of planar contour shape. *IPSJ SIG Notes, Computer Vision and Image Media*, (104):1–8, 1996.