PAPER

Calligraphy Generation Using Deformable Contours

Lisong WANG[†*], **Lifeng HE**[††], **Tsuyoshi NAKAMURA**[†], **Atsuko MUTOH**[†], ***Nonmembers,***
and **Hidenori ITOH**[†], ***Member***

SUMMARY This paper considers the problem of generating various calligraphy from some sample fonts. Our method is based on the deformable contour model *g-snake*. By representing the outline of each stroke of a character with a g-snake, we cast the generation problem into global and local deformation of g-snake under different control parameters, where the local deformation obeys the energy minimization principle of *regularization* technique. The base values of the control parameters are learned from given sample fonts. The experimental results on alphabet and Japanese characters *Hiragana* show such processing as a reasonable method for generating calligraphy.

key words: *calligraphy generation, deformable contour, regularization, font, g-snake*

1. Introduction

Nowadays, the desktop publishing system (DTP) is rapidly growing, inherently, imitating and generating artistic calligraphy of a character in diversity styles automatically with a computer is always demanded. However, writing style varies from person to person, the shape variety of the strokes of a character is quite complicated. Usually, it is not only difficult to capture the inner relationship between different writing style by simple rules, but also evaluate the quality of presented calligraphy. For these reasons, currently there are only two methods for calligraphy generation. One is preserving some different fonts in advance, and giving out a type of writing among them opportunely, as used in almost all editor or printing tool, and the other is using some manually defined deformation regulations based on some expert's experiences, as applied in many calligraphy processing systems [5], [6], [9].

Obviously, the first method lacks the flexibility, the number of type of calligraphy generated is font-limited, and enormous computer resources are required with the increase of amounts of the font, also to prepare fonts in advance is a work of time consuming. The second one needs human intervention too much, it depends on large numbers of trial-and-error trying, generally the satisfied results are hard to be obtained without some special knowledge about calligraphy. Moreover, both methods do not explain the relationship between different types of the same character.

Addressing to the weaknesses of the current methods, we propose a method for generating versatile calligraphy from limited fonts based on the model of deformation contour. Our problem can be formulated as follows. Given two sample calligraphy C_0, C_1 and a controlling parameter $t \in [0, 1]$, to construct versatile intermediate object C_t, together with the increase of t, C_t is a more violent deformation in resembling tendency from C_0 to C_1. In fact, our proposed method for generating calligraphy from limited fonts is an application of *regularization* technique [7]. We pay attention to the contours of strokes of a character, and cast the generation problem into global and local deformation of a deformable contour model.

Some deformable contour models based on regularization technique have been developed and successfully applied to many aspects of machine vision. Kass first established a well-known active contour model, the *snake* model to process vision tasks such as edge detection, boundary formulation and stereo and motion matching [2]. The deformation of a *snake* completely relies on the sum energy minimization.

To achieve the aim of contour extraction of arbitrary object under noisy image [4], Lai gives a more innovative one, so-called generalized active contour, the *g-snake* model, based on the stochastic relaxation theory [1]. The common feature of both models is that, for each model, there is a defined internal energy to keep the contour continuity, i.e., to keep the initial shape of a contour, and an imaginary external force acted on the contour to make it deform. The contour is attracted by such actions to deform to a desirable final state, where the sum of internal and external energy of the contour is minimal. Due to the different usage, the external force could be image intensity, edge, termination etc. An advantage of g-snake is that it is capable of representing any arbitrary shape. Moreover, it not only accounts for global changes due to rigid motions, but also retains the ability for local control.

When dealing with the calligraphy generation problem, we must consider both global and local deformation of a stroke. Therefore, the g-snake model

Manuscript received September 3, 1998.

[†]The authors are with the Department of Intelligence and Computer Science, Nagoya Institute of Technology, Nagoya-shi, 466–8555 Japan.

[††]The author is with the Faculty of Information Science and Technology, Aichi Prefectural University, Aichi-ken, 480–1198 Japan.

[*]Presently, with Imaging System Business Group, Ricoh Co., LTD.

Table 1 Some solutions using regularization.

Target object	**Regularization treatment**
Edge detection	$\min_{f} \int [(Qf - i)^2 + \lambda(f_{xx})^2] dx$
Intensity Restoration for sequential images	$\min_{f} \int\int\int [(Qf - i)^2 + \lambda(\delta f \cdot V + f_t)^2] dxdydt$
Optical flow detection from territory	$\min_{u,v} \int\int [(i_x u + i_y v + i_t)^2 + \lambda({u_x}^2 + {u_y}^2 + {v_x}^2 + {v_y}^2)] dxdy$
Surface reconstruction detection	$\min_{f} \int\int [(i_x u + i_y v + i_t)^2 + \lambda({u_x}^2 + {u_y}^2 + {v_x}^2 + {v_y}^2)] dxdy$
Color	$\min_{z} \|I^v - Az\|^2 + \lambda\|P_z\|^2$
Optical flow detection from contour	$\min_{v} \int \left[(V \cdot N - V^N)^2 + \lambda \left(\frac{\partial V}{\partial s}\right)^2 ds\right]$

can be used as the deformation model for our task. In our method, for sample data C_0 and C_1 mentioned above, we represent each stroke of a sample by a closed g-snake, investigate global deformation between corresponding strokes, imagine that there are pulling forces acted on C_0 to make it deform to C_1, and learn the contour deformable control parameter by using C_1 as the final convergence of enery minimization from initial position C_0. Then, we can control the generation of a series of interpolation C_t by varying the learned parameters.

The rest of this paper is constructed as follows. In next section, we briefly review the basic concepts used in regularization technique. Section 3 introduces g-snake model. We describe the application of g-snake for our problem in detail in Sect. 4, and show experimental results on alphabet and Japanese character "Hiragana" in Sect. 5. Finally, conclusion and future works are summarized in Sect. 6.

2. Regularization

For a problem P, if its solution relative to the initial conditions can be determined directly, it is called a well-posed problem. Inversely, if there are too much solutions, means the solution can not be determined directly, it is an ill-posed problem. For example, one equation with two variables is a typical ill-posed problem.

Regularization technique is a basic method to transform an ill-posed problem to a well-posed one. Here we use a simple model, the solution of linear equations, as our example to explain its basis concepts. This problem can be described as: for given vector $\mathbf{x}$, to find the vector $\mathbf{y}$ to meet Eq. (1).

$$A\mathbf{y} = \mathbf{x} \tag{1}$$

where, A is a linear operator that contains coefficients in each equation.

When the inverse matrix A^{-1} of A does not exist, $\mathbf{y}$ cannot be directly determined for a given $\mathbf{x}$, the problem is ill-posed. In such situation, a reasonable way is introducing some constraints on $\mathbf{y}$ to make it to be directly determined. For example, finding $\mathbf{y}$ such that for any linear transformation B, $\mathbf{y}$ makes the normal second power B, i.e., $\|B\mathbf{y}\|^2$, minimum. The problem is formulated as find $\mathbf{x}$ in expression 2.

$$\min_{y} \|A\mathbf{y} - \mathbf{x}\|^2 + \lambda\|B\mathbf{y}\|^2 \tag{2}$$

where, λ is Lagrangian multiplier. By such assumption, the solution can be directly determined, and then the problem is changed to be well-posed. Here, λ is also called regularization parameter.

Regularization has found a lot of applications in recognition aspect, such as edge detection, optical flow detection, surface reconstruction, etc. Table 1 lists some solutions using regularization to treat ill-posed problems in recognition aspect [7]. The body in integral symbol are the defined energy mentioned above, the Table 1 illustrates the regularization technique is a problem to find a minimum sum energy indeed. Detailed symbol definitions and their meanings should be referred to [7].

3. Deformable Contours Model: g-Snake

In [4], the major role of the g-snake model is to deal with an ill-posed problem, modeling and extracting arbitrary deformable contours of any object from an observational noisy image.

3.1 Representing Contour of an Object

In g-snake model, a contour is represented as a link of point vector constructing it, $\mathbf{U} = [\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u_n}]$, where, each $\mathbf{u} \in E = \{(x, y) : x, y = 1, 2, \cdots, M\}$, thus $\mathbf{U} \in \mathbf{E^n}$. Each element in $\mathbf{U}$ is also called *snaxel* (snake pixel). According to the vector combination principle, a snaxel $\mathbf{u_i}$ can be expressed as a linear combination of its two adjacent snaxel vectors:

$$\mathbf{u}_i = \alpha_i \mathbf{u}_{i_\alpha} + \beta_i \mathbf{u}_{i_\beta} \tag{3}$$

where the basis indices are given by:

$$i_\alpha = \begin{cases} i-1; & i>1 \\ 3 & i=1 \end{cases} \quad i_\beta = \begin{cases} i+1; & i<n \\ n-2 & i=n \end{cases} \tag{4}$$

Accordingly, collecting and putting all of snaxel i together, the shape equation of the contour of an object can be written down as:

$$\mathbf{AU^T} = 0 \tag{5}$$

where $\mathbf{A}$ is called shape matrix that contains the necessary information to describe the shape:

$$\mathbf{A} = \begin{bmatrix} 1 & -\beta_1 & -\alpha_1 & 0 & \cdots & 0 \\ -\alpha_2 & 1 & -\beta_2 & 0 & 0 & \cdots \\ 0 & -\alpha_3 & 1 & -\beta_3 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -\alpha_{n-1} & 1 & -\beta_{n-1} \\ 0 & 0 & \cdots & -\beta_n & -\alpha_n & 1 \end{bmatrix} \tag{6}$$

A contour $\mathbf{U}$ is called *nontrivial* if $|x_{i_\alpha} y_{i_\beta} - x_{i_\beta} y_{i_\alpha}| > 0$ for $1 \le i \le n$. The non-trivial condition ensures that $\mathbf{u_{i\alpha}}$ and $\mathbf{u_{i\beta}}$ are linearly independent for all i. Furthermore, $\mathbf{A}$ is stable for nontrivial contours because α, β and U are related by linear equations. This kind of shape representation is account for both global and local deformations:

- **Global Deformations**
  Global deformations correspond to rigid motion of contour such as scaling, rotation, stretching and dilation. It has been proven in [4] that the contour representation mentioned above keeps the following invariance property for a contour:

  > Two nontrivial contours satisfy the same shape equation if and only if they are related by a linear transformation.

  Therefore, according to this invariance property, the global deformation of a contour can be computed simplely by affine transformations without any influence on the shape matrix. This is a very desirable property, because the local deformations are troublesome, one hopes not having to consider their influence on global deformation.
- **Local Deformations**
  The g-snake models the local deformations as random fluctuation of snaxels in an interested family of contours $\mathbf{U} \in \Omega \supseteq E^n$. An internal energy E_{int} induced by shape matrix $\mathbf{A}$ is defined to represent such fluctuation:

$$E_{int}(\mathbf{U}) = \frac{(\mathbf{AU}^T)^T \mathbf{R}^{-1} (\mathbf{AU}^T)}{l(\mathbf{U})} \tag{7}$$

where $\mathbf{R} = diag\{\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2\}$ are the deformation variances σ_i^2 of snaxel i, $l(\mathbf{U})$ is a normalizing constant:

$$l(\mathbf{U}) = \frac{1}{n} \sum_{i=1}^{n} \| \mathbf{u}_{i+1} - \mathbf{u}_i \|^2$$

Now the probabilities of contour $\mathbf{U}$, $p(\mathbf{U})$, can be assigned as expression 8 which is so-called *Gibbs measure*:

$$p(\mathbf{U}) = \frac{1}{Z} \exp(-E_{int}(\mathbf{U})) \tag{8}$$

where Z is also a normalizing constant:

$$Z = \sum_{U \in \Omega} \exp(-E_{int}(\mathbf{U}))$$

Equivalently, expression 8 also defines the conditional probability of *Markov random field* to yield prior distributions for any arbitrary contour.

$$p(\mathbf{u}_i | \mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n) = p(\mathbf{u}_i | \mathbf{u}_{i_\alpha}, \mathbf{u}_{i_\beta}) \tag{9}$$

The expression 9 means that a contour $\mathbf{U}$ is entirely specified by all probability of $\mathbf{u}_i$, while a probability of $\mathbf{u}_i$ is specified by its two adjacent basis snaxels.

3.2 Maximum Posterior Estimation

The g-snake model treats the contour extraction task as a process to estimate unknown deformation of contour from an image. By using Bayesian framework, it can be realized as maximum posterior(MAP) estimation. Rewriting and denoting the internal and external energy as follows:

$$E_{int}(\mathbf{u}_i) = \frac{\| \mathbf{u}_i - \alpha_i \mathbf{u}_{i_\alpha} - \beta_i \mathbf{u}_{i_\beta} \|^2}{l(\mathbf{U})} \tag{10}$$

$$E_{ext}(\mathbf{u_i}, \mathbf{g}) = 1 - \mathbf{h_i}^T \mathbf{f}(\mathbf{u_i} + \mathbf{g}) \tag{11}$$

where $\mathbf{g}$ is an arbitrary reference point, $\mathbf{h}_i$ is an unit vector which indicates the interested deformation direction of a snaxel on the contour, and function $\mathbf{f}(x)$ corresponds the external force which pulls the contour to desirable position.

Then contour extraction task turns into solving problem of MAP estimation, formulated as an energy minimization problem of finding $\mathbf{U}_{map}$ and $\mathbf{g}_{map}$ in expression 12.

$$\{\mathbf{U}_{map}, \mathbf{g}_{map}\} = \arg\min_{\mathbf{U}, \mathbf{g}} \sum_{i=1}^{n} \left\{ \frac{\lambda_i}{1-\lambda_i} E_{int}(\mathbf{u}_i) + E_{ext}(\mathbf{u}_i, \mathbf{g}) \right\} \tag{12}$$

where

$$\lambda_i = \frac{\sigma_\eta^2}{\sigma_\eta^2 + \sigma_i^2} \in [0,1]$$

are the *local regularization parameters* which control the amount of local template deformation. σ_η^2 is the Gaussian distribution of the noise contained in force measurement. λ_i are derived from a method named local minimax criterion [3].

4. Calligraphy Generation System

In this section, we introduce our calligraphy generation system. The problem domain we address is to find agreement regulations to bind different writing types of a given calligraphy character, and to use such regulations to generation new calligraphy. We think that a different calligraphy of a character is a deformation of its stroke contours satisfying energy minimum principle under some external constraints.

Because of the deformation complexity, it is difficult to arrange simple and plain rules to comprehend deformations of every points for a stroke. Our basic thinking is to investigate the deformation of several salient feature points on sample calligraphy C_0 and C_1, assuming that the deformation is owing to some forces coming from C_1 and acting on the outline of strokes of C_0, and it will convergence to somewhere. To keep the sum of energy of points minimum, we can charge the position determination of remaining points with the energy minimum processing, and hence, to generate versatile calligraphy. The deformation of strokes can be simply described by the energy minimum principle. Now we describe our system in detail.

4.1 Processing Flow

Given two samples C_0, C_1, in order to generate varying calligraphys C_t, our system proceeds in following stages:

- Pre-process.
- Extract strokes from input characters.
- Complete the correspondence completion between each stroke of C_0 and C_1.
- Initialize a g-snake on each stroke.
- Investigate global deformation between corresponding strokes.
- Learn g-snake parameter to control local deformation.
- Vary the learned parameters to generate new calligraphy.

4.2 Pre-Processing

Because our method addresses the deformation of contour of stroke, we must find where the contour of stroke is at first. The aim of pre-processing is to detect position information of contour for every strokes of both C_0 and C_1.

The input of pre-processing is a sample calligraphy $C_l, l \in \{0,1\}$, such input can be treated as a binary image. The output is binary array matrix T_l ($l \in \{0,1\}$) that contains position information of contour. The dimension of matrix T_l is $i \times j$ where i (j) equals the width (height) of the input calligraphy binary image respectively. Then the binary map T_l is simplely established by following process: scan each pixel contained in the image from left up to right down, investigating whether the 8 neighbors of the pixel being processed are all with the same value, black or white. If the 8 neighbors a pixel contains both black and white pixels, it is a pixel on contour of stroke. The value of its correspondence elements in T_0 or T_1 must set to be 1, otherwise, for a non-boundary pixel, its correspondence elements in T_0 or T_1 is set to 0.

4.3 Stroke Extraction

After getting sets T_0 and T_1, the next task is to class its elements into strokes, since the stroke is the basic processing unit in our system. With such classification, we can decompose sample calligraphy to separate stroke set. The stroke extraction can be realized with a boundary tracing algorithm naturally, but one thing, the intersection of two different stroke must be considered in case suspicious extraction results.

In our system, the stroke extraction processing is semi-automatic. The salient feature points are decided manually. They are clicked and informed to the system with a man-machine interface. For example, to extract the first stroke of *Hiragana* "a," 6 points dotted in Fig. 1 have to be clicked.

For a character whose strokes are separated, no extra work needs to do except a boundary tracing is necessary. Supposing n feature points are clicked near the related stroke of each sample C_0 and C_1, each point clicked is aligned to its nearest contour, by matching it with T_0 and T_1 obtained in pre-processing stage, and saving their coordinates in two lists $K_0 = [k_{00}, k_{01}, \cdots, k_{0n}]$ and $K_1 = [k_{10}, k_{11}, \cdots, k_{1n}]$ at first, where, the elements in K_0 and K_1 with the same subscript are thought as correspondence. We call it a partial correspondence. For a stroke with no intersection,

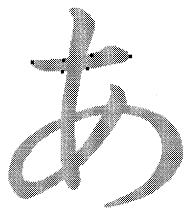


Fig. 1 Salient feature clicked.

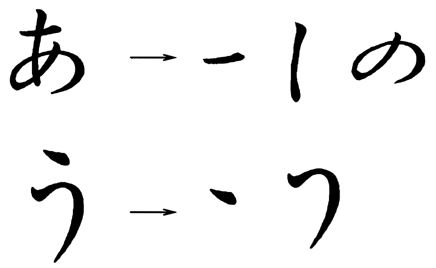


Fig. 2 Decomposing calligraphy into strokes.

its contour is traced by getting a route for each feature point to its next feature point: $k_{bi}, k_{bi+1}, b \in \{0,1\}, i \in \{0,1,\cdots,n\}$. For the nth feature point, its next point is the first one clicked.

If a stroke is acrossed by other stroke, the process is slight tiresome. With n clicked feature points, a stroke is divided into n segments. For each pair $k_{bi}, k_{b(i+1)}, 0 \le i < n, b \in \{0,1\}$, using $T_l (l \in \{0,1\})$ to trace and record a route along the boundary of the stroke (for the point last clicked searching a route to the first clicked one). If an intersection is encountered, the program will switch boundary tracing operation to spline interpolation, to fill the interval of boundary due to the intersection. Figure 2 is an example of decomposing *Hiragana* "a" and "u" into strokes.

4.4 Correspondence Completion

In this stage, the partial correspondence of feature points will be turned into a full correspondence of snaxels. Candidate snaxels to initialize a g-snake are selected from each route between two feature points produced at last stage for each related stroke, and a full correspondence for related stroke of two samples is established. Given a selected space value between two snaxel on C_0, for each element in K_0, the necessary number and coordinates of from itself to its next one can be calculated except the last one who's coming element is the first element. Then, for its feature point on C_1, assuming the number of snaxel to next element should be the same as its correspondence element in K_0, the proper adjusted space can be calculated and snaxel can be selected from the route produced for C_1 in last stage. This operation ensures that the number of snaxels between two correspondence feature points is the same. We think that there is a binding from the first to the last one, in all of segments between any two feature points. This is called a full correspondence map of two sample calligraphy. The full correspondence map will be used to produce a force map in order to do local deformation laterly.

4.5 g-Snake Initialization

To prepare for the deformation, using the snaxels found in the last stage, a closed g-snake is initialized and located on the contour of each stroke, which contains the necessary shape information. Figure 3 is the example of locating two g-snakes on the first stroke of *Hiragana* course, the space between two snaxels can be adjusted according to calligraphy being processed. Short space should be selected for calligraphy with steep slope, but it takes more time to find it minimum energy position afterwards. We determine it as 5 empirically for test.

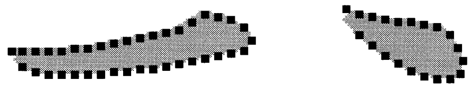


Fig. 3 Locating g-snake on a stroke.

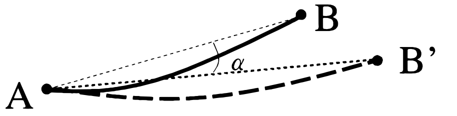


Fig. 4 Global deformation of stroke.

4.6 Investigating Global Deformation

In this processing stage, the global deformation of rigid motions such as rotation scale and stretching and dilation arose between sample stroke is investigated. After knowing the global deformation, we can make the stroke deform globally to some extent using affine-transformation. Because the rotation and the stretching have the strongest visual appeal, we investigate them between each related stroke of sample C_0 and C_1. The investigation occurs along the g-snakes in the unit of a pair of partial correspondence segments. Supposing that there are m segments on one g-snake, a segment on C_0 starts at x_0, y_0, ends at x_1, y_1, while their correspondence points on C_1 is x'_0, y'_0 and x'_1, y'_1, we first put the start snaxel of C_0 together with that on C_1 by translation $(x'_0 - x_0, y'_0 - y_0)$, then by comparing their coordinates of start and end snaxel, we can know the necessary stretching $\vec{s}_i$ and rotation angle θ_i for C_0 to deform to C_1 is:

$$\vec{s}_i = \frac{\sqrt{(y'_2 - y'_1)^2 + (x'_2 - x'_1)^2}}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \tag{13}$$

$$\theta_i = \arctan\left(\frac{y'_2 - y'_1}{x'_2 - x'_1}\right) - \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{14}$$

Then such deformation coefficients are put into list $S_U = (\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_m)$ and $\Theta_U = (\theta_1, \theta_2, \cdots, \theta_m)$. The elements in the two lists mark the global deformation of each segment. For example shown in Fig. 4, assuming arc AB and AB' are contour correspondence segments, the global stretching is calculated by comparing the length of line segment AB and AB', and the rotation angle α is the angle between them.

4.7 g-Snake Control Parameters Learning

In our system, the local deformation is simulated as the energy minimization process of closed g-snake on the outline of stroke. The function $\mathbf{f}(\mathbf{u}_i)$ describing the external energy in expression 11 is designated as a pulling force acted on snaxel on C_0, with direction points to its correspondence snaxel on C_1 and the strength in proportion to the normalizing constant $l(\mathbf{U})$ is described in the last section. Therefore, using the correspondence table created in previous stage and putting the heads of g-snake on C_1 together with its corresponding global deformed one on C_0, such directions and strengthes to form a force map is easily generated. Then we use this force map to minimize the total energy for the g-snake, and the local regularization parameters λ_i in expression 12 can be estimated by *local minimax criterion* [3].

4.8 Generating New Calligraphy

Having known the global deformation parameter S_U, Θ_U and the local regularization parameter λ_i, then we can generate a new calligraphy by a control variable $t(0 < t < 1)$. Giving a different value of t, first deform each stroke segment globally using affine transformation under parameters tS_U, $t\Theta_U$, and next deform it locally by g-snake energy minimization with regulation parameter $t\lambda_i$. In this way, we can control the deformation from weakly to violently.

5. Experiment Results

This section shows some examples of generated calligraphy using our novel generation method. We performed the simulation on a SUN-SPARC-II workstation. The calligraphy used as samples C_0 and C_1 are prepared as 256×256 bitmap image in advance, they are either fonts residented in our computer system, or read in with a scanner. Up to now, we tested our system on Japanese alphabet, *Hiragana*, and English alphabet for calligraphy generation.

In order to clarify the robust of our algorithm, we deliberately selected four kinds of calligraphy fonts of *Hiragana* existing in our computer system which are called *kaisho, gyosho, mini* and *gothic*. Calligraphy of such fonts for *Hiragana* "a" are shown in Fig. 5. It is obvious that these fonts have clearly different appearance. Distinctly, sample free is desirable for an algorithm. That is, an algorithm should be able to deal with all kind of samples in the problem domain.

Using a pair of calligraphy fonts of *Hiragana* "a" as samples, and executing the algorithm described in last section, due to the difference of control parameters λ, some generated examples are shown in Fig. 6.

In Fig. 6, calligraphy of the first row is used as sample C_0 and last row for C_1. The first column in Fig. 6 is a progressively proceeding from *kaisho* to *gyosho*, the second column is *kaisho* to *mini*, and the third is *gothic* to *mini*. After learning global and local deformation parameters, the calligraphy in second and third row are generated by setting t to 0.3 and 0.6. We can observe from this picture that the increase of t makes the lose the peculiarity of C_0, simultaneously, get that of C_1. This example shows that our algorithm is available for any selected sample.

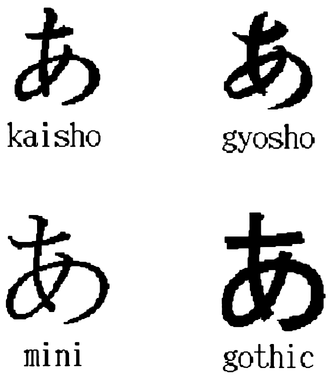


Fig. 5 Different type of calligraphy.

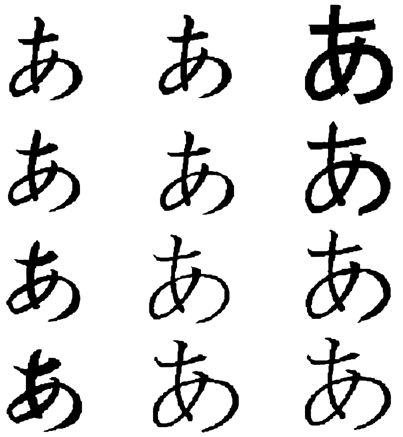


Fig. 6 Generated calligraphy.

Figure 7 gives generated calligraphys for some other *Hiragana* "i," "u," "e," and "o." Same as above, calligraphy of the first row is used as sample C_0 and last row for C_1. The control parameter is set to 0.3 and 0.6 for the second row and the third row respectively.

We also test our method on processing real calligraphy. The right down of Fig. 8 is a real calligraphy of *Hiragana* "a" read from a scanner. By using it and existing *Hiragana* "a" of font *kaisho* (left up of Fig. 8) as samples, we can generate intermediate calligraphy, the right up and the left down shown in Fig. 8 with t selected as 0.3 and 0.8.

The tests on English alphabet are also made. As it is simpler than the case of *Hiragana*, we only show one example in Fig. 9, where the left up is font calligraphy in our computer system, the right down is real calligraphy

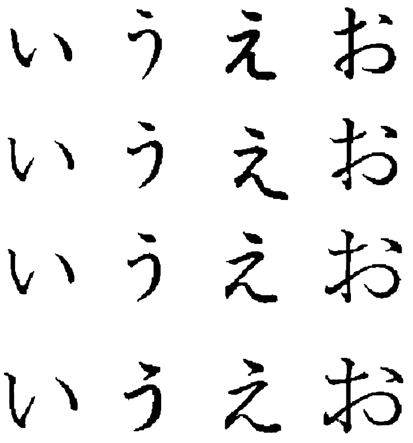


Fig. 7 Other generated calligraphy.

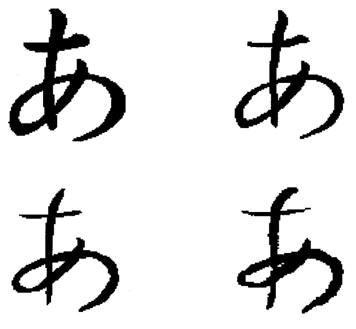


Fig. 8 Generated calligraphy from real sample.

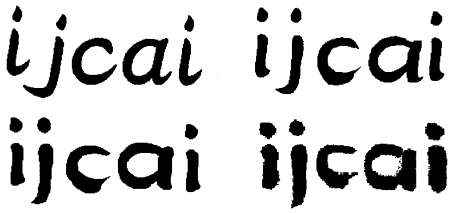


Fig. 9 Generated calligraphy.

read in with a scanner, and they are used as sample C_0 and C_1 respectively. The right up and left down are generated interpolations with the control parameter as 0.3 and 0.6.

From these experimental results, we know that that our algorithm do generate the medium states of samples, and the sample calligraphy can be any type of font including the real calligraphy. The deformation can be controlled by varying the parameter t. The results are similar to calligraphy written by real people.

6. Conclusions

We demonstrated that generating various calligraphy from different fonts can be modeled as global and local deformation of its stroke contour obeying the energy minimization principle of *regularization* technique. Therefore, we applied the deformable contour model g-snake solving the calligraphy generation problem. Such a method makes the calligraphy generation more flexible and the generated results are more natural. Some useful left work can be indicated as: extracting strokes of a character automatically, interpolating writing scratch between strokes, i.e., the semicursive style of writing, and the extension of our system to calligraph processing of Chinese characters.

References

[1] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," IEEE Trans. Pattern Anal. & Mach. Intell., vol.PAMI-6, pp.721–741, 1984.

[2] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," International Journal of Computer Vision, pp.321–331, 1988.

[3] K.F. Lai and R.T. Chin, "On regularization, formulation, and initialization of the active contour models (snakes)," Asian Conference on Computer Vision, pp.542–545, 1993.

[4] K.F. Lai and R.T. Chin, "Deformable contours: Modeling and extraction," IEEE Trans. Pattern Anal. & Mach. Intell., vol.17, no.11, pp.1084–1090, Nov. 1995.

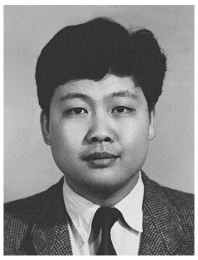
[5] T. Nakamura, H. Seki, and H. Itoh, "A calligraphy system based on analyzing user writing speed," Proc. 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, pp.189–190, 1994.

[6] T. Nakamura, H. Seki, and H. Itoh, "A fuzzy-based calligraphy system using fractals," Proc. 3rd European Congress on Intelligent Techniques and Soft Computing, pp.1440–1444, 1995.

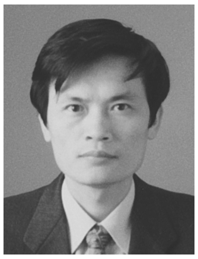
[7] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," Nature, vol.317, pp.314–319, Sept. 1985.

[8] T.W. Sederberg and E. Greenwood, "A physically based approach to 2-D shape blending," SIGGRAPH '92, ACM Computer Graphics, vol.26, no.2, pp.25–34, 1992.

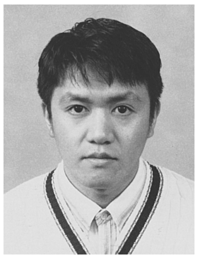
[9] X. Zhang, H. Ji, H. Sanada, and Y. Tezuka, "Forming brush-written HIRAGANA with the capability of providing versatile stroke-connecting flows," IEICE Trans., vol.J76-D-II, no.9, pp.1868–1877, 1993.



Lisong Wang received his M.S. and Ph.D. degrees from Nagoya Institute of Technology in 1995 and 1998, respectively. He currently works at Printer Business Division, Imaging System Business Group, Ricoh co., LTD. His research interests include fuzzy reasoning and image processing.



Lifeng He received his B.E. degree from Northwest Institute of Light Industry, China, in 1982, the M. S. and Ph.D. degrees from Nagoya Institute of Technology in 1994 and 1997, respectively. In 1998, he joined the Faculty of Information Science and Technology, Aichi Prefectural University, where he became an associate professor in 1999. His research interests include automated reasoning, theorem proving, knowledge bases, multi-agent system and image processing. He is a member of Information Processing Society of Japan.



Tsuyoshi Nakamura received his Ph.D. degree from Nagoya Institute of Technology (NIT) in March 1998. He is currently a research associate at NIT. His interest and work have been in developing softcomputing applications. He is a member of Information Processing Society of Japan and Japan Society for Fuzzy Theory and Systems.

Atsuko Mutoh received her B.Eng. from Nagoya Institute of Technology (NIT) in March 1998. She is currently a technical staff at NIT. Her interests include evolutionary learning and softcomputing.

Hidenori Itoh received his Ph.D. degree from Nagoya University, Japan, 1974. From 1974 to 1985, he worked at NTT. From 1985 to 1989, he was with ICOT. Since 1989, he has become a professor at the Nagoya Institute of Technology. His current research interests include multimedia, artificial life and AI. He is a member of Information Processing Society of Japan and Japan Society for Fuzzy Theory and Systems.