Vol. 41 No. 6 IPSJ Journal June 2000

Regular Paper

Understanding Illustrations of Origami Drill Books

Jien Kato,[†] Toyohide Watanabe,[†] Hiroyuki Hase[††]
and Takeshi Nakayama[†††]

This paper introduces a new approach for understanding the folding process of origami based on interpreting a series of illustrations of origami drill books. Although many graphics recognition schemes have succeeded in dealing with individual graphical images, they cannot be adapted for recognition of a sequence of contextually related graphics such as origami illustrations. Our approach is based on a dynamically updated internal model of origami. The key characteristics of this approach are as follows: (1) It interprets not only what folding operations are specified in each single image, but how origami will change and what origami will seem to be in other images, according to the recognized operations. (2) It provides an efficient way to associate intermediate interpretation with a related image. (3) As a result of the understanding, the folding process can be completely described using the internal model.

1. Introduction

Over the last three decades, many researchers have investigated methods for graphics recognition and interpretation with the objective to enable graphics components to be accessed as semantically complete entities, for applications such as indexing image components from a pictorial database, determining locations in a geographical information system or modifying graphics in a CAD system[1)~8)]. From the viewpoint of the style of information processing, these methods can be divided into two types: top-down and bottom-up[9)]. Usually, the former type is knowledge based. For top-down methods, Niyogi and Srihari[2)] described a production system for document understanding. Antoine, et al.[3)] proposed an approach for interpretation of different classes of technical documents such as city maps and mechanical drawings using a priori knowledge. The latter type is generally based on structure analysis, that is, structural analysis of lines and their interconnections is required to generate meaningful and succinct descriptions of the objects. For example, Kasturi, et al.[6)] and Ejiri, et al.[7)] proposed typical methods of this type for recognition of flow-charts, engineering drawings and maps. Although these methods are different with respect to style, they all deal with individual graphics images. In other words, these methods limit the focus to the same recognition issue: How can various graphical shapes and their spatial relationships be recognized?

On the other hand, we need to observe graphics recognition and interpretation from another point of view, namely, that an object can be perceived not only as a simple static entity which is contained in individual image but as a changing object associated with a series of contextually related graphics images. We can easily find many examples in real-world applications for such objects: weather maps, comic strips and illustrations of origami drill books for instance. In this paper, we distinguish interpreting several related graphics from interpreting single graphics. The difference is that in the former case, the contents of a following image always depend on the contents of a previous image. So, in addition to the question "*how can various graphical shapes and their spatial relationships be recognized in each individual image?*", a satisfactory solution has to be found to the issues such as

- *How can intermediate interpretation be described?*
- *How can the interpretation results for a previous image be associated with a following image so that the interpretation process can be continued?*
- *How can intermediate interpretation be used to help recognition of the following graphics images?*

Unfortunately, although the methods proposed so far have led to much success in coping with individual graphics, they have not discussed these issues. Of course these methods are inadequate to the purpose of recognition and inter-

[†] Department of Information Engineering, Graduate School of Engineering, Nagoya University
[††] Faculty of Engineering, Toyama University
[†††] International Academy of Media Arts and Sciences

pretation of a series of related graphic images.

In this paper, we propose an approach for understanding the folding process of origami based on interpreting a series of illustrations. We select origami as the recognition object for discussing the issues stated above, because of the advantage that the domain of origami is smaller and closed. In a general origami drill book, the folding process of origami is explained by a sequence of illustrations [10]. Each frame of the illustrations depicts the present state of the origami, which results from the folding operations indicated in previous frames, and specifies a few new folding operations. So, the state of the origami always varies from frame to frame. The key characteristics of our approach are that it interprets not only what operations happened in a frame but how the origami changed from one frame to another during the folding operations. Moreover, folding process can be accurately represented in a dynamically updated internal model of origami as the recognition results.

In the following sections, we first discuss some basic problems when interpreting several related illustrations of origami drill books, and introduce an internal model to overcome these problems as the basis of our methodology. We then propose a method to associate the states of origami, described in the internal model, with the illustrations for interpreting what folding operation is specified and how origami is changed, based on DTW (dynamic time warping) and simulated annealing. Lastly, we evaluate our approach through examples. This paper does not cover the topic of document image analysis. The issue of image analysis with the purpose of recognition of various meaningful graphical entities has been discussed in Kato, et al. [11]

2. Approach

This section describes the basic concepts in our approach, introduces the internal model of origami, and presents a summary of our recognition system.

2.1 Some Definitions about Origami

We know that there are more than 10 kinds of folding operations used in traditional paper folding art. But most of them are essentially composed of a *basic* operation "folding back". This operation divides a face into two, and rotates one partition by 180° around the division line. Moreover, as shown in **Figs. 1** (a) and (b), it can further be divided into two different types: "mountain-folding" and "valley-folding" according to the rotation direction. Hence the factors which can be specified in a basic operation are the *folding line*, *folding direction* and *folded face.* In contrast to basic operations, more complicated operations are classified as *compound* operations since they can be a combination of two or more basic operations under some physical restrictions. For example, in Fig. 1 (c) "tucking in" and "covering" can be regarded as a combination of a "mountain-folding" and a "valley-folding" where two neighboring faces have to be simultaneously rotated in opposite directions.

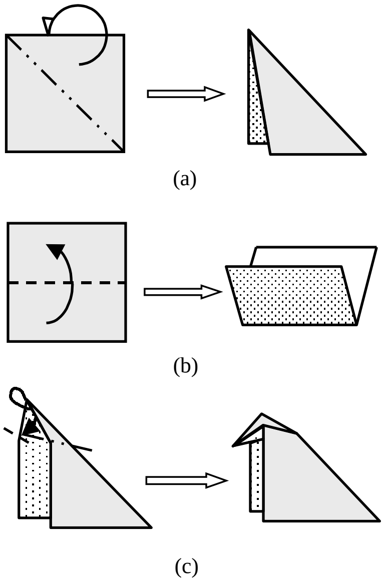


Fig. 1 Folding operations are generally described in origami drill books in the way shown in (a)–(c). (a)(b) Two types of the basic operation. (c) Compound operations "tucking in" and "covering". The shadowed faces denote the inside faces of the origami.

On the other hand, we find that the folding operations can be also categorized into the different classes such as "single-face folding" and "multi-face folding" in accordance with the folded faces. Namely, the operations where only one face is folded belong to the former class, and conversely, those where two or more faces are folded fall into the later class. Obviously, the compound operations are certainly multi-face folding. The basic operation "folding back" seems to be single-face folding, but it is not always true. We would like to stress that when "folding back" is applied to a face, not only is this face folded but any face that has *connectivity* or *overlap* relationships directly or indirectly with the folded face may folded or moved at the same time. The same thing can be said for the compound operations. As a result, we classify the folded faces as follows:

- Active face, the face an operation is immediately applied to.
- Passive face, the face is folded or moved due to connectivity or overlay relationships with an active face.
- Divided passive face, the passive face that is divided.
- Undivided passive face, the passive face that is not divided but moved.

We give three examples in **Fig. 2** to show the distinctions between an active face and a passive face, between a divided passive face and an undivided passive face and between connectivity relationships and overlap relationships among faces. In particular, Fig. 2 (a) shows a case of undivided passive faces where a passive face F_p is moved because of the connectivity relationship with an active face F_a. Figures 2 (b) and (c) show the examples of divided passive faces F_ps due to the connectivity, overlap relationships with active faces F_as, respectively.

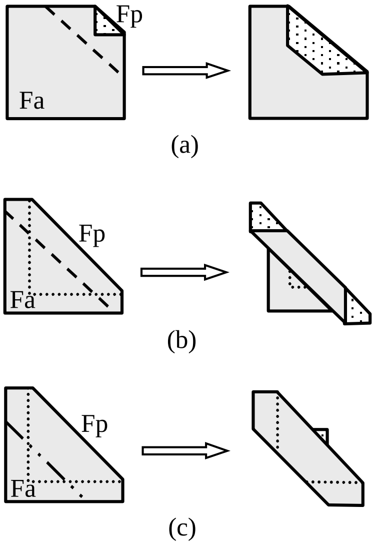


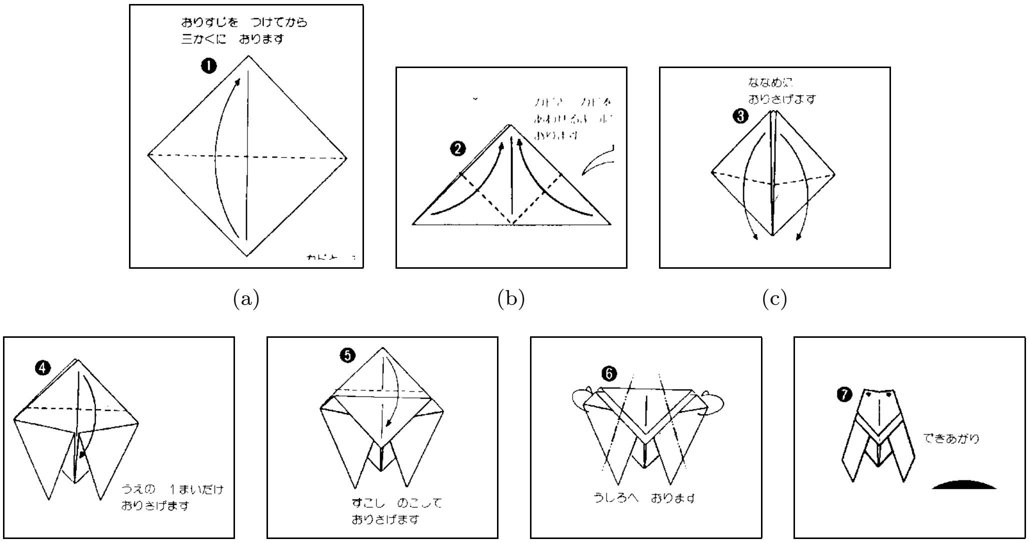
Fig. 2 The distinctions between an active face and a passive face, between a divided passive face and an undivided passive face, and between connectivity relationships and overlap relationships are illustrated here. (a) An active face F_a and an undivided passive face F_p. F_p moves because of the connectivity relationship with F_a. (b) An active face F_a and a divided passive face F_p. F_p (hidden behind F_a) is folded because of the connectivity relationship with F_a. (c) An active face F_a and a divided passive face F_p. F_p (hidden behind F_a) is folded due to the overlap relationship with F_a.

The discussion until now has made it clear that when interpreting a folding operation, the folding lines, directions and folded faces which define the operation must be recognized. Here the folded faces include not only active faces but passive faces, and moreover the passive faces may exist due to either connectivity or overlap relationships among faces.

2.2 Problems in Folding Process Recognition

Besides explanatory text, we think that the most useful information in the illustrations of origami drill books are indicative information and contextual information. In this section, we distinguish the two kinds of information, and argue the problems in interpreting several contextually related illustrations that represent a folding process.

Indicative information helps us perceive the operations at each step (frame). **Figure 3** shows seven images that illustrate a complete folding process called "cicada". We find that indicative information in each image is conveyed to readers by certain of meaningful graphical entities: polygons to express the states of origami and some symbols to indicate the contents of folding operations. The second type of the entities includes broken lines and arrows. The broken lines are used to represent folding lines, while two types of arrow-heads combined with different patterns of broken lines, that is, a filled arrow-head with a dashed line and an unfilled arrow-head with a dash-dot-dot line are customarily to express two different folding directions. Furthermore, the arrows indicate moving partitions of folded faces by the positions of their end points. So, it is convenient to use indicative information for efficient recognition of folding operations from each individual image. However, indicative information is not enough to find all the folded faces because the connectivity and overlap relationships among faces are not visually evident in a 2D drawing. What is more, the further a folding process goes, the more complicated the relationships become. How to maintain the connectivity and overlap relationships during the recognition process is one of problems related to folding process recognition.



(a) (b) (c)

(d) (e) (f) (g)

Fig. 3 The folding process called “cicada”.

In addition to indicative information, contextual information is very important and available to interpret individual operations and successive operations. Contextual information is provided by neighboring frames in a series of illustrations. It tells us both of the sequence of folding operations and the states of origami before and after performing a certain operation. The states of origami in neighboring frames of the illustrations are usually expressed from the same angle, but sometimes the view in the following frame is rotated (mostly by 180°) from the view in the previous frame. The variation between views is always clearly indicated by a symbol, a special arrow. To recognize successive folding operations, if the operation specified in the previous frame has been identified, not only do the new connectivity and overlap relationships resulting from this operation have to be registered, but the changs in the shape of the origami, taking the view into account, must be associated with the contents of the following frame of illustrations so as to prepare for the next recognition step. This is another difficult problem for folding process recognition.

2.3 Internal Model of Origami

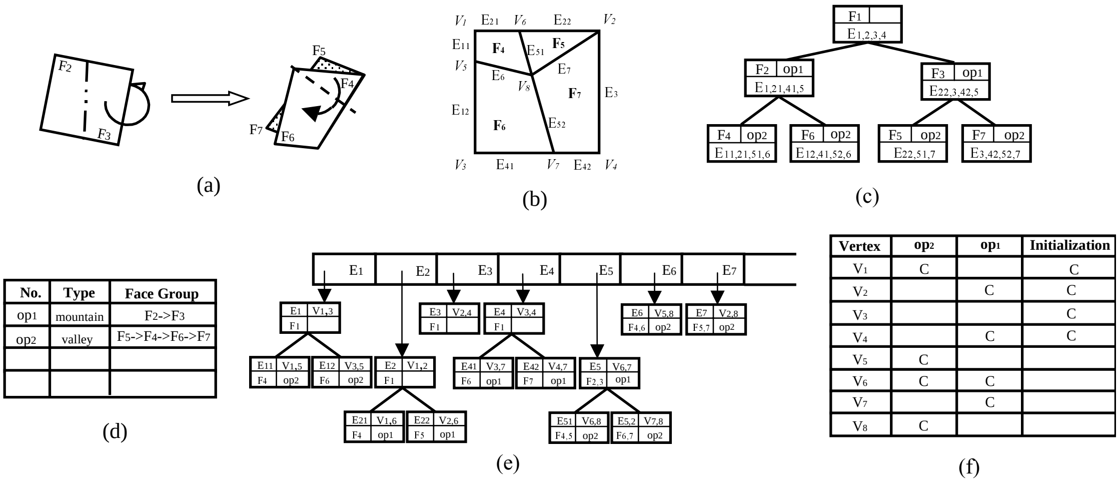
We introduce an internal model of origami to solve the problems stated above. As we have pointed out, this model should be able to describe the changes in the states of origami and maintain the connectivity and overlap relationships among faces during the recognition process. Furthermore, on the basis of this model not only the rotation, expansion/reduction of origami, but its projection on any plane has to be easily acquired.

We define the internal model as consisting of three layers: face layer for description of the division of faces, edge layer for description of the division/generation of edges and vertex layer for description of the movement of vertices. Objects of three classes, namely tree, stack and list are used to construct this model.

2.3.1 Face Layer

The division course of origami is described as a binary tree, called face tree. The nodes of the tree indicate faces. The root especially stands for the initial state of the origami, a sheet of paper without any folding lines. When a face is folded, the child nodes, corresponding to the two face segments, are generated and connected to the node, corresponding to the folded face. Thus, the branches of the tree preserve the connectivity relationships between the faces that are separated in the same operation. In this manner, a face tree can describe successive folding operations. The terminal nodes always represent the current state of the origami.

During the folding process, the faces corresponding to the terminal nodes generally form one or more planes in space. It should be clear that overlap relationships exist only among faces which are on the same plane and touch with each other. Therefore, when a folding op-



No.	Type	Face Group
op1	mountain	F2->F3
op2	valley	F5->F4->F6->F7

(d)

Vertex	op2	op1	Initialization
V1	C		C
V2		C	C
V3			C
V4		C	C
V5	C		
V6	C	C	
V7		C	
V8	C		

(f)

Fig. 4 Examples of descriptions of the internal model. (a) Illustrations. (b) The crease pattern of origami. (c) The face tree. (d) The operation list. (e) The edge trees. (f) The vertex stacks.

eration has taken place, the faces related to the terminal nodes can be divided into groups so that the faces in the same group have the overlap relationships and others do not. We describe the overlap relationships within a face group by using a stack. Naturally, the overlap relationships depend on both of the current situation of origami and the folding operation being applied under this situation.

As a consequence, the face layer is composed of a face tree and an operation list as shown in **Fig. 4**. An operation list is used to describe interpretation results such as operation types and the stacks that keep the overlap relationships resulting from the operations. A face tree and an operation list for the example shown in Fig. 4 (a) are illustrated in Figs. 4 (c) and (d). Each node in Fig. 4 (c) contains the face number, edge numbers and operation number. Figure 4 (d) shows that there is only one face group. The faces are overlapped in the order of $F_2 \rightarrow F_3$ after "mountain-folding", and in the order of $F_5 \rightarrow F_4 \rightarrow F_6 \rightarrow F_7$ after "valley-folding". Connectivity relationships among faces can be easily obtained by edge information attached to each node. That is, the faces with connectivity relationships should have a common edge. For example, F_4 is connected to F_5 and F_6 because the common edge E_{51} exists between F_4 and F_5, and E_6 between F_4 and F_6, respectively.

2.3.2 Edge Layer

As a face is separated into smaller pieces, some of the edges may be divided and some new edges may be yielded. A similar binary tree, called edge tree, is used to described the division course of edges as illustrated in Fig. 4 (e). When an edge is separated, the nodes corresponding with two edge segments are generated and connected to the node corresponding to the divided edge. Each node contains related face, vertex and operation numbers so as to conveniently access the data in other layers.

The number of edge trees grows as new edges appear. For example, in Fig. 4 (e) the tree with the root E_5 is generated as the face F_1 is folded into two (F_2 and F_3), and the trees with the root E_6 and E_7 are generated as the same face is folded into four (F_4, F_5, F_6 and F_7). We use a list to preserve the roots of edge trees.

2.3.3 Vertex Layer

The vertex layer is composed of a group of stacks, called vertex stacks, as shown in Fig. 4 (f). Each stack is assigned to a vertex to maintain its coordinates in space. Whenever a new vertex is created or an existing vertex is moved, the new coordinates are stored into the corresponding stack. We use "C" instead of the values and put it at the column corresponding to the related operation. The top of each tack is at the left side. For example, in the first operation op_1 the vertices V_2 and V_4 are moved and V_6 and V_7 are newly generated, thus the new coordinates are added into the stacks these vertices belong to. In the second operation op_2 since the vertices V_1 and V_6 are moved

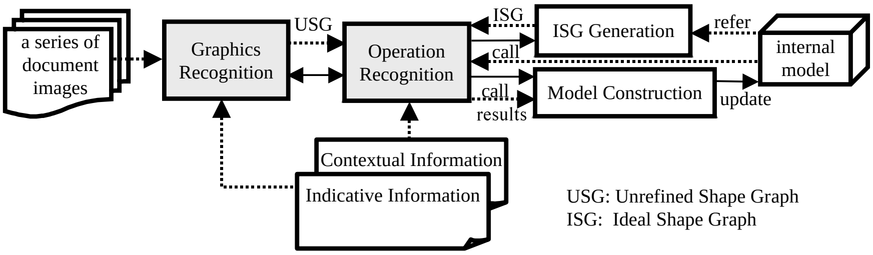


Fig. 5 Framework of folding process recognition.

and V_5 and V_8 are newly generated, the new coordinates are also put into the corresponding stacks. Accordingly, stacks for V_3, V_5, V_7 and V_8 preserve one set of coordinates since these vertices never moved, whereas stacks for V_1, V_2, V_4 and V_6 keep two sets of coordinates because they moved once.

We have described the internal model of origami using three data layers. Advantages of this model can be summarized as follows:

- It can easily describe the connectivity and overlap relationships among faces. When origami turns over, the internal model can adjust the overlap relationships only by reversing the order of faces in the face groups (it can be easily done when using stacks).
- The shape of the origami that will seem to be in the following frames can be obtained according to this model.
- Backtracking can be easily implemented just by removing related nodes from face tree and edge trees, and related items from operation list and vertex stacks, which are accompanied with target operation numbers.

2.4 Outline of Recognition

This section presents the overall view of our recognition approach. Our approach consists of two main stages[12]. In the *first* stage, various meaningful graphical entitiesare detected for recognizing the folding operations specified in each individual image. In the *second* stage of our approach, the recognition results from the first stage is associated with the current state of the origami, obtained from the internal model.

Therefore, our recognition system is composed of graphics recognition module and operation recognition module as shown in **Fig. 5**. In Fig. 5, the ISG generation process and model construction process are the sub-modules and invoked by the operation recognition process. At first, the model construction process is invoked once to create the initial state of the internal model of origami. The graphics recognition process then interprets the folding operations based on various meaningful graphical entities that were detected, and describes the recognition results in a graph structure called USG (Unrefined Shape Graph). Once a USG is obtained, the ISG generation process is invoked to make an ISG (Ideal Shape Graph), a projection of the internal model on x-y plane (perpendicular to the view of the image). An ISG has a similar data structure to a USG, however it is not constructed from the current image but the previous image. It shows how the origami is expected to be in the current frame of the illustrations. The operation recognition process then matches the ISG to the USG. If the corresponding relationships between the faces in the ISG and USG can be precisely obtained, we can identify the folding lines, directions and folded faces (active faces), and moreover discover all the passive faces according to the connectivity and overlap relationships among faces described in the internal model. Lastly, the model construction process is invoked to update the internal model in accordance with the effects of the folding operations. Such recognition steps will be repeated until a folding process is completed.

3. Recognition of Folding Process

The main steps for recognition of folding process are as follows:

(1) Generation of ISGs
(2) Matching a USG to an ISG
(3) Updating the internal model

Now we describe these steps in the following sections with an emphasis on Step (2).

3.1 USG and ISG

The issue of graphics recognition with the aim of extracting various meaningful graphical entities has been presented in Kato, et al.[11] We summarise the details here for introducing

USGs.

Graphics recognition consists of 5 steps:

- Removing character strings
- Separation of solid regions
- Segmentation into line segments
- Arrow detection
- Broken line detection

Separation of text regions from an original image of origami illustrations is the first task. After the text regions have been separated, the image almost contains thin entities except illustration numbers and filled arrow-heads as depicted in Fig. 3. The solid regions are then differentiated. When solid regions have been also distinguished and removed, a thinning algorithm is applied to the image so that the remaining entities can be described by thin lines. Then we find out all the feature points such as junctions, intersections and sharp corners from the obtained thinned image, and segment the image into line segments. Lastly, we detect the two types of arrows and two types of broken lines.

It can be seen from the above discussion that in the graphics recognition stage, we have not only recognized the meaningful graphical entities such as arrows and broken lines, but also decomposed remaining thin entities into line segments at some places which include the significant feature points. Note that some of them may not be feature points because the segmentation is based on a polygonal approximation algorithm. Hence a part of these line segments and their ending points will be identified as the edges and vertices of origami in the subsequent operation recognition stage.

A USG is mainly used to describe the results of graphics recognition where arrows and broken lines have been removed for convenience. It is characterized by edges and vertices:

- Any edge in a USG stands for a line segment (a straight line).
- Any vertex in a USG stands for a terminal of a line segment.

Thus the edges and vertices of a USG are interconnected in such a way that the shape of origami are expressed as it is described in the illustration. **Figure 6** gives us a pretty good idea of what a USG is and where it comes from.

On the other hand, an ISG has the same data structure (graph) as that of a USG but contains different types of information. An ISG is basically a orthographic projection of the internal model. It also consists of edges and vertices, but

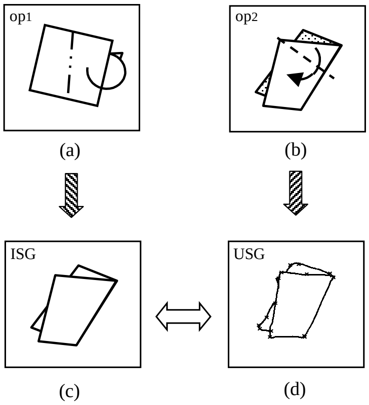


Fig. 6 Examples of a USG and an ISG. (a)(b) The illustrations as shown in Fig. 4 (a). (c) The ISG generated based on the internal model when the first operation has been recognized. (d) The USG generated based on the results of graphics recognition. The vertices of the USG are denoted by cross marks.

- Any edge in an ISG indicates a folded line.
- Any vertex in an ISG indicates a terminal of a folded line.

Here a folded line is a line formed when origami is folded in a perfect way in the sense that the corresping vertices or edges completely meet. We regard four edges of origami as the special cases of folded lines.

An ISG is generated as follows:

(1) Choose a face group perpendicular to the view from the current situation of the face tree.

(2) Project orthographically all the faces in selected group. Note that the projection should obey the constraints of overlap relationships among the faces.

(3) Generate the graph. The edge number and the vertex number used in the internal model are attached to each edge and vertex in the graph.

The ISG generated for the example in Fig. 4 (a) after applying the first operation can be obtained using the above procedure. The face group consisting of F_2 and F_3 is chosen and projected, taking the overlap relationship into account (i.e., put F_3 behind F_2). The generated ISG should have the structure shown in Fig. 6 (c).

As we have mentioned, origami is usually drawn from the same angle in neighboring frames of illustrations. Although sometimes the view is rotated (by 180°), such a variation must be distinctly indicated in the previous frame using a special arrow. Therefore, the ISG being matched to the current image can be gained by projecting the model on x-y plane or doing that after rotating the model by a specified angle.

3.2 Matching an ISG to a USG

A USG is not a simple revolution or expansion/reduction in size of an ISG. Here we stress that:

- A USG includes some extra parts that are not found in an ISG, for example, noise and the graphical entities which slightly express origami in 3D effect (see Fig. 3).
- A USG almost never has the completely same topological structure as its corresponding ISG, since illustrations are not strictly accurate copies of real origami.

Because of these problems, simple matching techniques based on comparing the coordinates of all vertices or examining the connections of all edges are inappropriate and useless[13].

A USG actually resembles its corresponding ISG. It is proper to define the meaning of resemblance as similarity in shape and topology. From this viewpoint, we propose a flexible and feasible matching method with higher tolerance for the local differences in shape and topology between a USG and an ISG. Our method consists of two stages. In the first stage we match vertices of an ISG to those of a USG based on dynamic programming, so as to estimate the expansion/reduction ratio of the two graphs. Then in the second stage we associate the ISG to the USG using simulated annealing technique. The solution obtained will be optimal in some meaningful sense, namely, it will best "explain" the corresponding relation between the ISG and the USG.

3.2.1 The First Stage of Matching

Suppose that a vertex A and a vertex B are characterized as a sequence of vectors as shown in **Fig. 7**. Each of vectors is parallel to an edge and connected to the vertex in the direction apart from it:

$$A = a_0 a_1 \cdots a_I,$$
$$B = b_0 b_1 \cdots b_J.$$

To calculate the distance between A and B, how the pattern A should be compared with B must be determined. We introduce a concept called "route" for this purpose. A route from $(0, 0)$ to (I, J) is defined as

$$\tilde{p} = (p_0, p_1, \cdots, p_K),$$
$$p_k = (i(k), j(k))$$

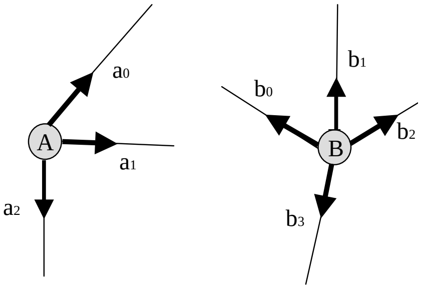


Fig. 7 The definition of vertices.

$$\in \{0, 1, \cdots, I\} \times \{0, 1, \cdots, J\},$$

for $k = 1, \cdots, K$, and it must meet either one of these conditions:

$$i(k) = i(k-1) + 1 \text{ and } j(k) = j(k-1),$$
$$i(k) = i(k-1) + 1 \text{ and } j(k) = j(k-1) + 1,$$
$$i(k) = i(k-1) \text{ and } j(k) = j(k-1) + 1.$$

That is, a route gives a map from a set of element positions of A ($\{0, 1, \cdots, I\}$) into a set of element positions of B ($\{0, 1, \cdots, J\}$). So, A can be compared with B on the basis of a route.

However, if $i(k-1) = i(k)$, the point $a_{i(k-1)}$ ($= a_{i(k)}$) is related to two points $b_{j(k-1)}$ and $b_{j(k)}$ since $j(k-1) \neq j(k)$. In this case, we think $a_{i(k)}$ which should be related to $b_{j(k)}$ is omitted, or $b_{j(k)}$ which did not exist originally has been inserted. Such kind of insertion/omission often happens in USGs and ISGs because a USG is never completely the same as its corresponding ISG. We define the functions:

$$\alpha(p_k) = \begin{cases} a_{i(k)} & \text{if } i(k) = i(k-1) + 1; \\ * & \text{if } i(k) = i(k-1), \end{cases}$$
$$\beta(p_k) = \begin{cases} b_{j(k)} & \text{if } j(k) = j(k-1) + 1; \\ * & \text{if } j(k) = j(k-1), \end{cases}$$

to cope with the insertion/omission case, where an omission is denoted by "$*$".

Now, the distance between A and B connected with route $\tilde{p}$ can be defined as the sum of the distance between all the element pairs in $\tilde{p}$:

$$D(A, B; \tilde{p}) \equiv \sum_{k=1}^{K} d(\alpha(p_k), \beta(p_k)). \qquad (1)$$

The distance between A and B is naturally viewed as the minimum of the distances related to all the routes as

$$D(A,B) \equiv \min\{D(A,B;\tilde{p}) \mid \tilde{p} \in P((0,0),(I,J))\}, \tag{2}$$

where $P((0,0),(I,J))$ denotes all the routes from $(0,0)$ to (I,J). When A stands for a vertex of an ISG and B for a USG, the distance of each element pair, d, is defined as the angle between vectors:

$$d(\alpha(p_k),\beta(p_k)) = \begin{cases} \operatorname{acos}\left(\frac{a_{i(k)}\cdot b_{j(k)}}{|a_{i(k)}||b_{j(k)}|}\right) & \text{if } \alpha(p_k) \neq *, \beta(p_k) \neq *, \\ 4\pi & \text{if } \alpha(p_k) \neq *, \beta(p_k) = *, \\ 0 & \text{if } \alpha(p_k) = *, \beta(p_k) \neq *. \end{cases} \tag{3}$$

In Eq. (3), a large distance 4π is given if an insertion arises in an ISG, while a zero-distance is given if an omission occurs in an ISG. This is because the former case usually signals a bad matching and the latter can be accepted.

We efficiently compute Eqs. (1)–(3) using dynamic programming[14]. Some vertices which are very close to each other are selected to calculate the expansion/reduction ratio between an ISG and a USG.

3.2.2 The Second Stage of Matching

For convenience, we use the notation

$ISG: v_1, v_2, \cdots, v_M,$

$USG: \nu_1, \nu_2, \cdots, \nu_M, \cdots, \nu_N.$

to express an ISG and a USG. Suppose that the first M vertices of the USG correspond to the first M vertices of the ISG. We specially call such an arrangement "state" for the USG. Let S be a finite set of the states, if each state S_j in S is given a cost $c(S_j)$, the objective of the second stage of matching is to search for the optimal state S_{opt} so that

$$c(S_{opt}) \leq c(S_j), \quad \forall S_j \in S. \tag{4}$$

Here the cost is a function which represents a quantitative measure of the "badness" of matching.

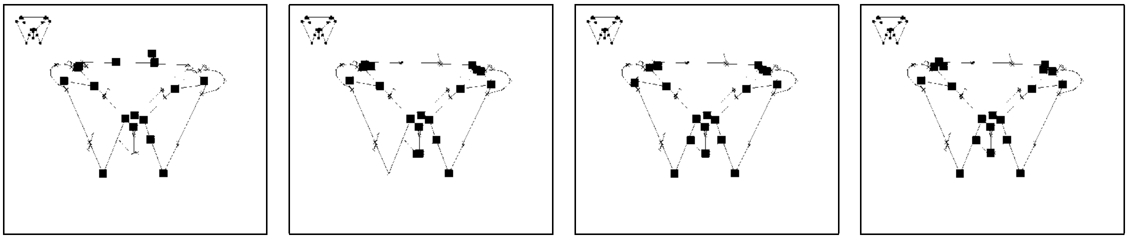
All exact methods for solving such a problem require a computing effort that increases exponentially with the size of S. When S is very large, simulated annealing will become a much more powerful means than any exact method[15]. We use a simulated annealing algorithm to search for global optimal solutions for our problem because at the steps near to the end in a complicated folding process, the number of vertices in a USG may increase into tens of hundreds.

Our algorithm is described as follows. The initial temperature T_0 and the initial state S_0 are first given. Then, a basic annealing step is repeated unless a stopping criterion is satisfied. In each iteration of this basic annealing step, a small random alteration of states is produced by exchanging the positions of two vertices. The caused variation in the cost, Δc, is then calculated based on a cost function. If the cost is less than the old state, the alteration is accepted and the new state is used as the starting point of the next repetition.

On the other hand, cases where the cost in the new state is more than that in the old state are treated probabilistically. The new state is accepted with the probability $p(\Delta c, T) = \exp(-\Delta c/T)$. A random number uniformly distributed in the interval $(0,1)$ is generated and compared with $p(\Delta c, T)$. If it is less than $p(\Delta c, T)$, the new state is accepted, otherwise the old state is used to start the next iteration again. In any case the temperature is lowered slightly. We repeat the basic annealing step until a stopping criterion is satisfied, that is, there is no any new state to be accepted during repeating the step n times (n is a large integer). When the temperature becomes sufficiently low, the final state can be expected to be the optimal solution.

The above algorithm is described in C style as follows.

```
Simulated Annealing(S0, T0) {
  T = T0;
  S = S0;
  for (i = 0; "stopping criterion" is not satisfied; i++) {
     for (j = 0; j < n; j++) {
          S' = generate(S);
          if (accept(c(S'), c(S), T)) {
                 S = S';
          }
          t = i * n + j;
          T = update(T, t);
     }
  }
}
generate(S) {
    m = random(1, N);
    n = random(1, N);
    S' = exchange(S, m, n);
    return(S');
}
accept(c(S'), c(S), T)) {
```



(a) $T_0 = 0.72135$, $t = 0$ (b) $T = 0.5870$, $t = 5000$ (c) $T = 0.5429$, $t = 10000$ (d) $T = 0.4779$, $t = 35000$

Fig. 8 An example of simulated annealing process. The USG and the ISG are shown in the center and the left-upper corner in each image. The vertices of the USG are indicated by a cross mark, but the selected M vertices which are supposed to match to the M vertices of the ISG are emphasised by thick rectangles. (a)–(d) show how the M vertices of the USG gradually approach the correct positions.

```
        $\Delta c = c(S') - c(S)$;
        $y = p(\Delta c, T)$;
        $r = random(0, 1)$;
        if ($r < y$)
            return(1);
        else
            return(0);
}
update(T, t) {
        return($\Delta/\log(t + 2)$);    /* $\Delta$ is a constant.*/
}
```

Where the probability p is given by

$$p(\Delta c, T) = \begin{cases} \exp(-\Delta c/T) & \text{if } \Delta c > 0 \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

and the cost function is defined as the total dissimilarities of two graphs:

$$c(S) = \sum_{i=1}^{M} \sum_{j=1}^{M} (\alpha \cdot E_a(i,j) + \beta \cdot E_s(i,j)), \tag{6}$$

$$E_a(i,j) = \overrightarrow{v_i v_j} \cdot \overrightarrow{\nu_i \nu_j} / |\overrightarrow{v_i v_j}||\overrightarrow{\nu_i \nu_j}|, \tag{7}$$

$$E_s(i,j) = ||\overrightarrow{v_i v_j}| - R \cdot |\overrightarrow{\nu_i \nu_j}||, \tag{8}$$

where R denotes the expansion/reduction ratio between an ISG and a USG which has already been obtained in the first stage of matching.

It should be clear that the initial temperature must be high enough so that the most of states, especially the states leading to the cost increase, can be accepted. Hence the formula

$$T_0 = \frac{\overline{\Delta c}^{(+)}}{\ln \chi_0^{-1}} \tag{9}$$

is often used, where $\overline{\Delta c}^{(+)}$ is the average of increased costs and χ_0 is a large positive number[16]. We adopt Eq. (9) to calculate T_0 because we want to avoid local minimum solutions. If T_0 is not high enough, the final solution will be trapped in a local minimum. The coefficients of α and β in Eq. (6) can be set to $\alpha = \beta = 1$ since it is reasonable to think that the discrepancy of one degree, the measure for the direction, is roughly the same as that of one pixel, the measure for the distance.

An example of the simulated annealing process is shown in **Figs. 8** (a)–(d). In the experiment $T_0 = 0.72135$ and $n = 5000$. States under different temperatures are expressed by thick rectangles at the first M vertices in the USGs. T and t indicate the temperature and the repeating times of the basic annealing step.

3.3 Updating the Internal Model

In this section, we show the principal rules for modifying folding lines and determining passive folded faces. We also briefly describe how to renew the internal model.

3.3.1 Modifying Folding Lines

If an ISG is correctly associated with a USG using the method described in the previous sections, the positions of the folding lines related to the specific folded faces can be obtained through the ISG and the USG. Since small errors may exist, we apply some rules to revise the positions. The basic rules are as follows:

- If a terminal of a folding line is very close to a vertex of a face, move the folding line to pass through the vertex.
- If a folding line is very close to an edge, move the folding line so that it overlaps this edge.

3.3.2 Searching for Passive Folded Faces

Passive folded faces contain divided and undivided ones. A divided face has a moving por-

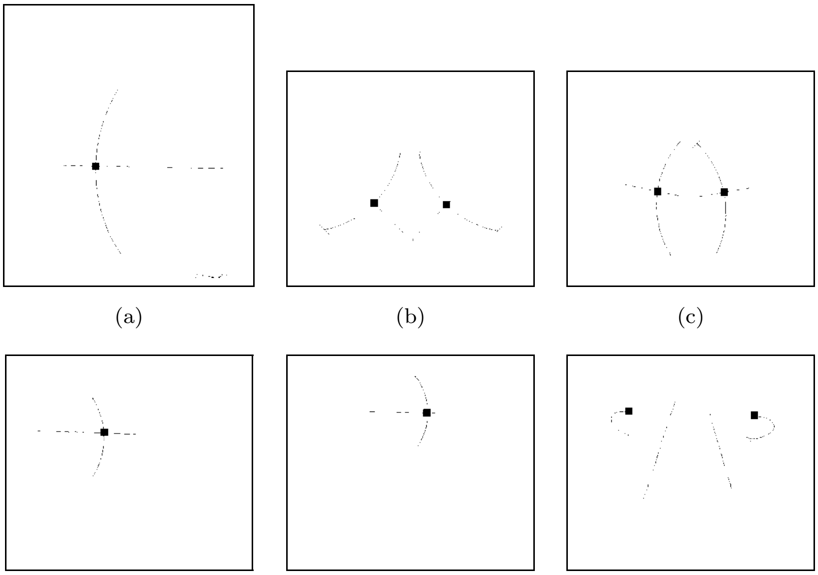


Fig. 9 The extracted different types of arrows and different types of broken lines in graphics recognition stage. They are used to interpret the individual folding operations.

tion and a fixed portion, while an undivided face does not have any fixed portion because it is moved in its entirety. We search for the passive folded faces according to the rules as follows:

- If a face F_p has a common edge with a moving portion of a folded face F_a, this face is determined to be a passive face. If the folding line passes through it, F_p is a divided face, otherwise it is an undivided face (see Figs. 2 (a) and (b)).
- If a face F_p overlaps a moving portion of a folded face F_a, and the moving portion is rotated toward F_p, the face F_p is determined to be a passive face (see Fig. 2 (c)).

3.3.3 Renewing the Internal Model

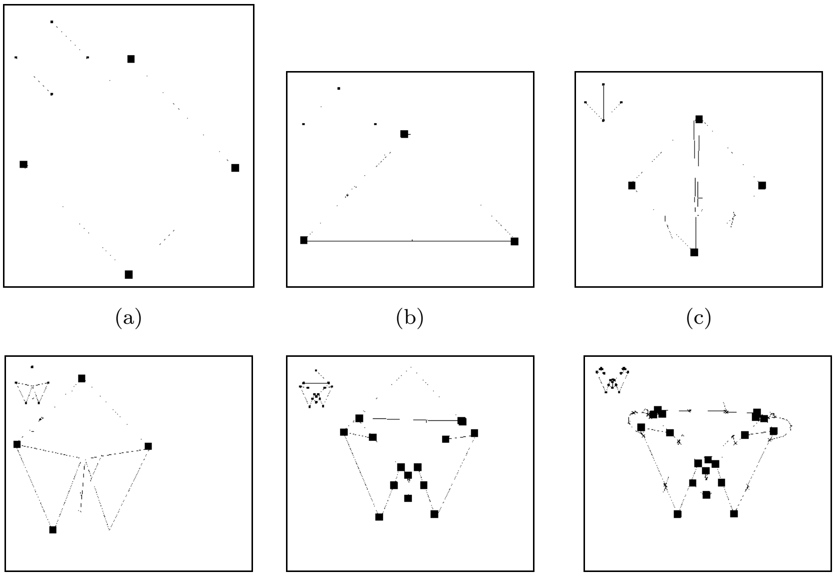
The internal model is updated in accordance with identified folding lines, directions and all folded faces. For an active face or a divided passive face, some terminal nodes are generated and connected to the face tree and edge trees. Coordinates of moved vertices are pushed into corresponding vertex stacks. The new edges, if they exist, are registered as a new edge tree and their terminals are newly registered in a vertex stack. For an undivided passive face, only the new coordinates of vertices should be added to the stacks. No node is connected to the face tree or the edges trees. A new face stack for the overlap relationships among faces must be generated according to the current state of the origami.

4. Experimental Results

Our approach for recognition and interpretation of the folding process has been tested with a number of origami illustrations. In this section, we evaluate our approach through an example: the folding process "cicada" consisting of seven images (see Figs. 3 (a)–(g)).

We show some of the outputs of our graphics recognition in **Fig. 9**. All the arrow-heads, including filled and unfilled types, and arcs were correctly detected and distinguished. In Fig. 9, filled arrow-heads and arrow-tails are indicated by small and big cross marks, and unfilled arrow-heads, crossing points between an arc and a broken line by a thick rectangle. As expected, different types of broken lines are also correctly extracted. For filled arrow-heads, the arrows crossing a dashed line are interpreted as an indication of "valley-folding", while for unfilled arrow-heads, the arrows along with a dash-dot-dot line are interpreted as "mountain-folding".

Figure 10 shows the situation of USGs after the first stage of matching. In addition to a USG, each image includes an ISG, generated according to the internal model, in the left-upper corner. Results of DP matching for estimat-



(a) (b) (c)

(d) (e) (f)

Fig. 10 The results of DP matching. Each image not only shows a USG but includes the corresponding ISG in its left-upper corner. The vertices of USGs which have been associated with a short-distance vertex of ISGs are emphasised by a thick rectangle.

ing the expansion/reduction ratio can be seen from the vertices. The vertices of USGs which have been associated with a short-distance vertex of ISGs are indicated by a thick rectangle. It can be observed that all the vertices in ISGs have been almost matched to the correct ones of USGs in the first three frames. But because of the differences between an ISG and a USG or the similarities among vertex patterns within a graph, remaining images do not completely reach the correct solution. For example, in Fig. 10 (d) the vertex in the center of the ISG cannot find a short-distance vertex, since the topology of the ISG is too different from the USG in this part. On the other hand, in the same image two vertices in the bottom of the ISG are mis-matched to the same vertex in the USG, since they closely resemble each other. Although the results of DP matching contain some errors, they are satisfactory for estimating the expansion/reduction ratio. We show the ratios of each edge in ISGs to the corresponding edge in USGs in **Fig. 11**. They are calculated according to the results of DP matching of Figs. 10 (d)–(f). Clearly, the expansion/reduction ratio can be decided as $R = 0.50$. The data greatly apart from R come from the mis-matched edges.

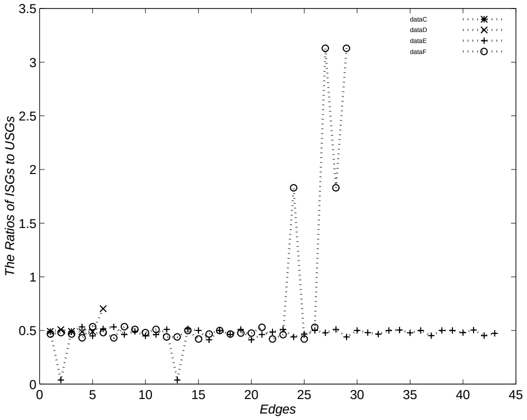
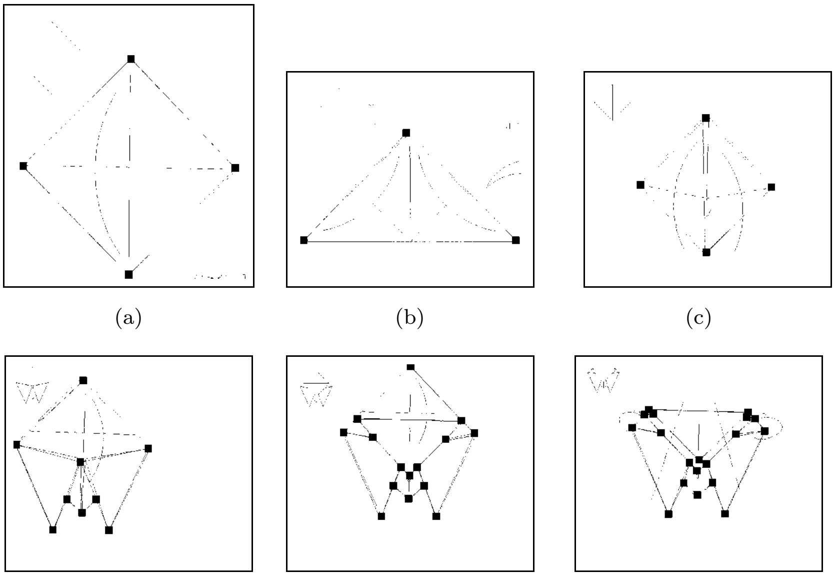


Fig. 11 The expansion/reduction ratio between the USGs and ISGs at $R = 0.50$.

The results of DP matching constitute the initial states of the simulated annealing algorithm. The final states of simulated annealing are shown in **Figs. 12** (a)–(f). Note that unlike Fig. 10, Fig. 12 does not show USGs but thinned images where the M selected vertices in the final state and the edges between them are drawn. It is obvious that the USGs are almost a perfect match to the ISGs in all images. In Fig. 12 (d), the ISG has fewer vertices than that of the USG (see Fig. 10 (d)) because of the different topological structures. It forces a face in the right-lower part of the USG to have a com-



(a) (b) (c)

(d) (e) (f)

Fig. 12 The final states of simulated annealing process. On the thinned images of illustrations, the M selected vertices are indicated by thick rectangles. To show the recognized faces, we link the vertices if there exists an edge between them.

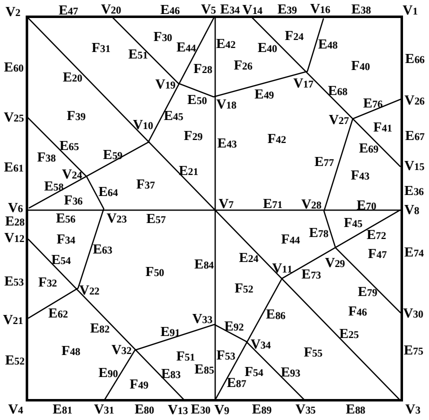
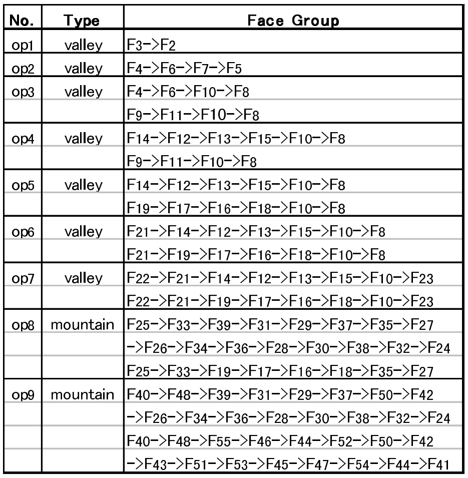

V2 E47 V20 E46 V5 E34 V14 E39 V16 E38 V1
F30 F31 E51 E44 E42 E40 F24 E48
E60 E20 F28 F26 F40 E66
V19 E50 V18 E49 V17 E68
V25 F39 V10 E45 F29 E43 F42 V27 E76 V26 F41 E67
E65 F38 E59 E21 E77 E69 F43 V15
E61 V24 E58 F37 E64 F36 E36
V6 V7 E71 V28 E70 V8
E28 E56 V23 E57 F45
V12 F34 E63 F44 E78 E72 F47 E74
E54 F50 E84 E24 V11 E73 V29
E53 F32 V22 F52 E79
V21 E62 E86 F46 V30
E82 V33 E91 E92 E25
E52 F48 V32 F51 F53 V34 F55 E75
E90 E83 E85 F54 E93 E87 F49
V4 E81 V31 E80 V13 E30 V9 E89 V35 E88 V3

Fig. 13 The crease pattern for "cicada".

mon vertex with its neighboring face, but such a common vertex does not exist in the USG. This result can be improved by examining the connections among vertices. Namely, if a vertex of a USG has inconsistent connections with those of a corresponding vertex in an ISG, it should be replaced by another appropriate one. Of course, the selected vertex and the replaced vertex have to be close with each other in the distance. To show the recognized faces, we link the vertices in Fig. 12 if an edge exists between them.



No.	Type	Face Group
op1	valley	F3->F2
op2	valley	F4->F6->F7->F5
op3	valley	F4->F6->F10->F8
		F9->F11->F10->F8
op4	valley	F14->F12->F13->F15->F10->F8
		F9->F11->F10->F8
op5	valley	F14->F12->F13->F15->F10->F8
		F19->F17->F16->F18->F10->F8
op6	valley	F21->F14->F12->F13->F15->F10->F8
		F21->F19->F17->F16->F18->F10->F8
op7	valley	F22->F21->F14->F12->F13->F15->F10->F23
		F22->F21->F19->F17->F16->F18->F10->F23
op8	mountain	F25->F33->F39->F31->F29->F37->F35->F27
		->F26->F34->F36->F28->F30->F38->F32->F24
		F25->F33->F19->F17->F16->F18->F35->F27
op9	mountain	F40->F48->F39->F31->F29->F37->F50->F42
		->F26->F34->F36->F28->F30->F38->F32->F24
		F40->F48->F55->F46->F44->F52->F50->F42
		->F43->F51->F53->F45->F47->F54->F44->F41

Fig. 14 The operation list for "cicada".

The internal model generated for "cicada" is shown in **Figs. 13–17** for completeness. As the result of understanding the folding process, it can be used for the input of a graphics simulation system.

The experiments were performed on a Sun SPARC station-10 with 40 MHz clock, running Solaris version 1.0.2. We used C++ to implement our approach. Typical CPU times for each execution of the principal stages, namely,

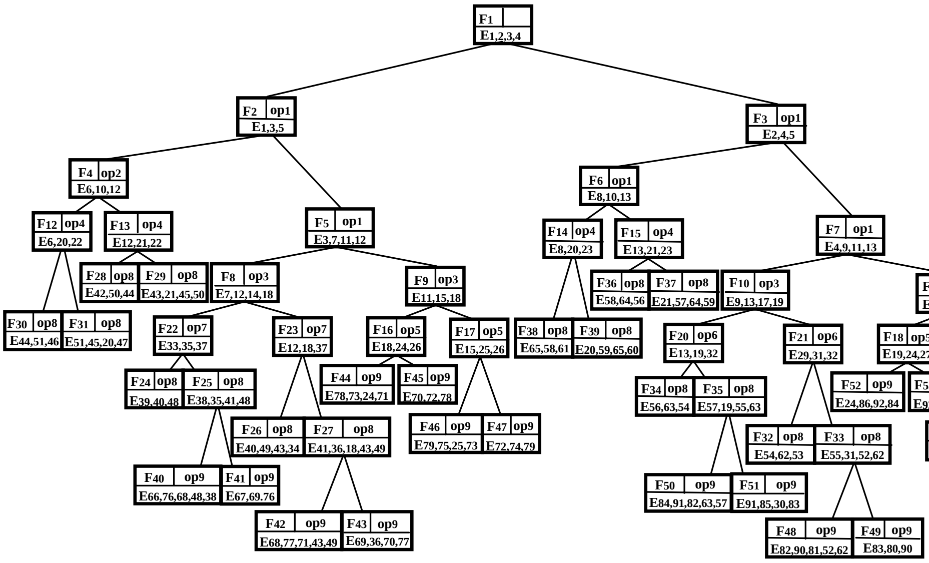


Fig. 15 The face tree for "cicada".

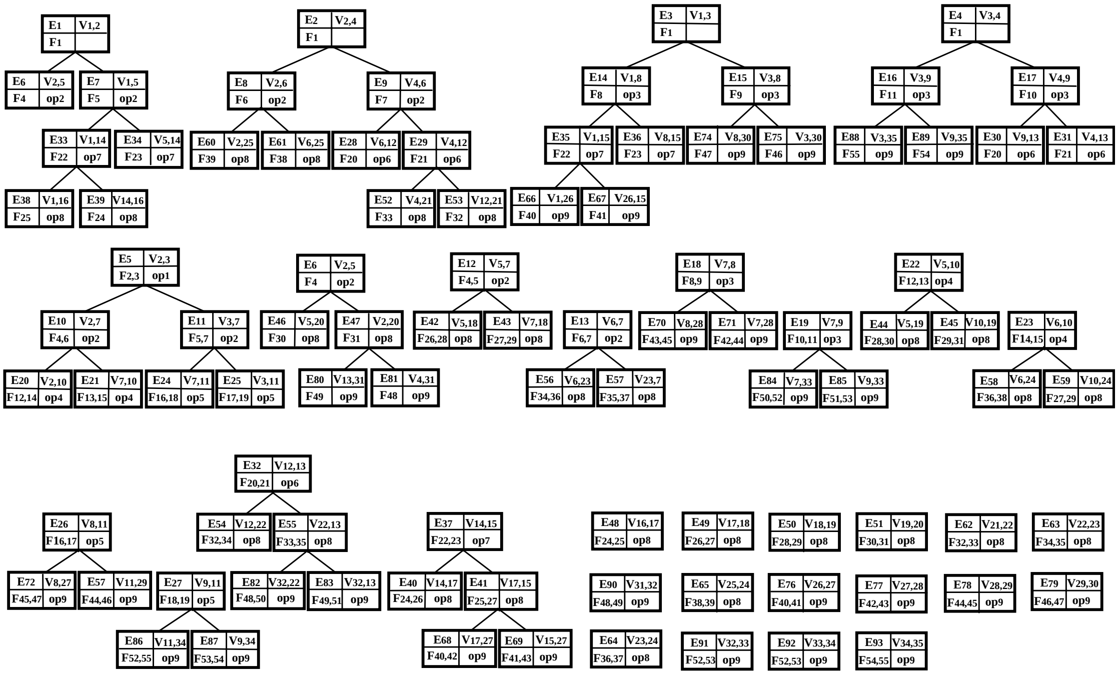


Fig. 16 The edge trees for "cicada".

graphics recognition stage and operation recognition stage are in the order of minutes.

5. Conclusions

In this paper, we presented a new approach for folding process recognition of origami. Although much of the work on origami has discussed issues such as knowledge representation of folding process[17], animating folding process by means of key-frame images[18], and simulating interactive folding operations in 3D virtual space[19], the topic of understanding fold-

Vertex	op9	op8	op7	op6	op5	op4	op3	op2	op1	Initilize
V_1			0.0,14.0							0.0,50.0
V_2						-11.5,-7.7		0.0,50.0		-50.0,0.0
V_3					11.5,-7.7		0.0,20.0			50.0,0.0
V_4				0.0,9.3					0.0,50.0	0.0,-50.0
V_5		1.5,14.0						25.0,25.0		
V_6		1.5,14.0						-25.0,25.0		
V_7								0.0,0.0		
V_8	-1.5,14.0						25.0,25.0			
V_9	-1.5,14.0						25.0,25.0			
V_{10}						0.0,20.0				
V_{11}		1.5,20.6			0.0,20.0					
V_{12}										
V_{13}	-1.5,20.6			-20.4,29.7						
V_{14}		1.5,23.9	-18.0,32.0	20.4,29.7						
V_{15}	-1.5,23.9		18.0,32.0							
V_{16}		-20.4,29.7								
V_{17}		-6.6,32.0								
V_{18}		-14.0,14.0								
V_{19}		-10.7,22.1								
V_{20}		-17.3, 6.2								
V_{21}		-14.0,23.3								
V_{22}		-7.5,29.7								
V_{23}		-14.0,14.0								
V_{24}		-10.7,22.1								
V_{25}		-17.3, 6.2								
V_{26}	20.4,29.7									
V_{27}	6.6,32.0									
V_{28}	14.0,14.0									
V_{29}	10.7,22.0									
V_{30}	17.3, 6.2									
V_{31}	14.0,23.3									
V_{32}	7.5,29.7									
V_{33}	14.0,14.0									
V_{34}	10.7,22.1									
V_{35}	17.2, 6.2									

Fig. 17 The vertex stacks for "cicada".

ing process based on recognition and interpretation of a series of illustrations has remained almost untouched. The main characteristics of our methodology can be summarized as follows:

- It is effective to interpret not only what operations are specified in a single image, but under the operations how origami will change and how origami will seem to be in a related image. Furthermore, we proposed an efficient way to associate the interpretation results with the images. Namely, our approach succeeded in interpreting several contextually related graphics images.
- Recognition and interpretation are based on a dynamically updated internal model of origami. This internal model has the sufficient ability to describe folding process, including the connectivity and overlap relationships among faces in each recognition step. As a result, this model can be used for applications such as input to a graphics simulation system, for demonstration or training.

The effectiveness of our approach have been verified through a number of samples such as "cicada", "kappa" and "airplane". We should point out that these instances are essentially composed of the basic folding operations. For more complicated samples which include compound operations, it is necessary to extend the graphics recognition algorithm to detect various patterns of arrows, for example, partly missing arrows or the line patterns of arcs that partly change. However, the recognition and interpretation methods described in this paper are applicable to other types of folding operations.

As further extension of our present work, the approach for integrating pattern information and natural language information has to be investigated. In fact, additional text in each frame of illustrations are often of interest. Although they provide supplementary explanation, in some cases they become more important than graphics. For example, for 3D origami (we mean the origami including some 3D operations at the final stage, such as "extend the wings" in "crane"), the 3D operations usually cannot be well recognized without the text. Incorporating natural language information and constructing a mechanism for integrating the two different kinds of information is the ultimate aim of this research project. The work presented in this paper is an essential step toward such an intelligent graphics recognition system.

References

1) Nagy, S., Seth, S.C. and Stoddard, S.D.: Document Analysis with an Expert System, *Proc. Intl. Workshop on Pattern Recognition*, Gelsema, E.S. and Kanal, L.N. (Eds.), Practice 2, pp.149–159, Elsevier Science Publishers (North-Holland) (1985).

2) Niyogi, D. and Srihari, S.N.: A Rule-Based System for Document Understanding, *Proc. AAAI*, Philadelphia, pp.789–793 (1986).

3) Antoine, D., Collin, S. and Tombre, K.: Analysis of Technical Documents: The REDRAW System, *Proc. IAPR Workshop on Structural and Syntactic Pattern Recognition*, New Jersey, pp.192–230 (1990).

4) Baird, H.S. and Thompson, K.: Reading Chess, *IEEE Trans. PAMI*, Vol.12, No.6, pp.552–559 (1990).

5) Fahn, C.S., Wang, J.F. and Lee, J.Y.: A Topology-Based Component Extractor for Understanding Electronic Circuit Diagrams, *Computer Vision, Graphics and Images Processing*, Vol.44, pp.119–138 (1988).

6) Kasturi, R., Bow, S.T., El-Masri, W., Shah, J., Gattiker, J. and Mokate, U.: A System for Interpretation of Line Drawings, *IEEE Trans. PAMI*, Vol.12, No.10, pp.978–992 (1990).

7) Ejiri, M., Kakumoto, S., Miyatake, T., Shimada, S. and Iwamura, K.: *Automatic Recognition of Engineering Drawings and Maps in Image Analysis Applications*, Kasturi, R. and Trivedi, M.M. (Eds.), Marcel Dekker (1990).

8) Lysak, D.B. and Kasturi, R.: Interpretation of Line Drawings with Multiple Views, *Proc. 10th ICPR* (*Pattern Recognition Systems and Applications Subconference*), pp.220–222 (1990).

9) Kasturi, R. and O'Gorman, L.: Techniques for Line Drawing Interpretation: An Overview, *Proc. IAPR Workshop on Machine Vision Applications*, Tokyo, pp.151–160 (1990).

10) Yamaguchi, M.: *A Cyclopaedia for Paper Folding Art* (in Japanese), p.324, Seito Company (1990).

11) Kato, J., Watanabe, T. and Nakayama, T.: Recognition of Essential Folding Operations: A Step for Interpreting Illustrated Books of Origami, *Proc. 4th Intl. Conference on Document Analysis and Recognition*, Ulm, Germany, pp.81–85 (1997).

12) Kato, J., Watanabe, T., Nakayama, T., Guo, L. and Kato, H.: A Model-Based Approach for Recognizing Folding Process of Origami, *Proc. 14th Intl. Conference on Pattern Recognition*, Brisbane, Australia, pp.1808–1811 (1988).

13) Maeda, N. and Hata, S.: A Global Identification Method of Object Patterns in Picture Book Using Positional Relation of Region, Technical Report of IEICE (in Japanese), PRU93-72, HC93-46 (1993).

14) Sakoe, H. and Chiba, S.: Dynamic Programming Algorithm Optimization for Spoken Word Recognition, *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-26, Vol.1, pp.43–49 (1978).

15) Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: Optimization by Simulated Annealing, *Science*, Vol.220, No.4598, pp.671–680 (1983).

16) Aarts, E. and Korst, J.: *Simulated Annealing and Boltzmann Machines*, Wiley (1990).

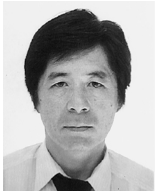
17) Uchida, T. and Itoh, H.: Knowledge Representation of Origami and Its Implementation, *J. IPS Japan* (in Japanese), Vol.32, No.12, pp.1566–1573 (1991).

18) Agui, T., Takeda M. and Nakajima, M.: Animating Planar Folds by Computer, *Computer Vision, Graphics, and Image Processing*, Vol.24, pp.244–254 (1983).

19) Miyazaki, S., Yasuda, T., Yoko, S. and Toriwaki, J.: An Origami Playing Simulator in the Virtual Space, *J. Visualization and Computer Animation*, Vol.7, pp.25–42 (1996).

(Received February 4, 1999)
(Accepted March 2, 2000)

Jien Kato received the ME and PhD degrees from Nagoya University in 1990 and 1993, respectively. After graduation, she became a research associate in the Faculty of Engineering at Toyama University. In 1999, she joined the Robotics Research Group at University of Oxford. Presently she is an associate professor in the Department of Information Engineering at Nagoya University. Her current research interests include document understanding, pattern recognition and computer vision. She is a member of the Information Processing Society of Japan, Institute of Electronics, Information, and Communication Engineers of Japan and IEEE Computer Society.



Toyohide Watanabe received the BS, ME, and PhD degrees from Kyoto University in 1972, 1974, and 1983, respectively. In 1987, he became an associate professor in the Information Engineering Department at Nagoya University. He is currently a full professor. Dr. Watanabe's research interests include the semantic data model, multi-media database, object-oriented paradigm, parallel and distributed processing, and document image understanding. He is a member of the Information Processing Society of Japan, Institute of Electronics, Information, and Communication Engineers of Japan, Japan Society for Software Science, Japan Society for Artificial Intelligence, ACM, AAAI, and IEEE Computer Society.

Hiroyuki Hase received his BE degree in electrical engineering from Toyama University in 1971, and PhD degree from Tohoku University in 1989. He is associate professor of the Department of Intellectual Information Systems Engineering, Toyama University. His current research interests are document image analysis, character recognition, tracing of moving objects and face detection. He is a member of the Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers, the Institute of Image Information and Television Engineers, the Institute of Image Electronics Engineers of Japan and IEEE Computer Society.

Takeshi Nakayama graduated with a degree in Psychology in 1957 from Waseda University, where he obtained a Master's degree in 1960, and a Dr. of Engineering in Electric Communication in 1970. He joined the Central Research Laboratory of Hitachi, Ltd. in 1963, where he engaged in research on various human interface systems including sound quality evaluation, image quality evaluation, the evaluation of Japanese text entry systems, and the evaluation of programming environments. Since 1987 he had been on the Faculty of Engineering, Toyama University, Toyama, Japan, where, he was a Professor in the Department of Intellectual Information Systems. He retired Toyama University at the end of March, 1999. He started his work again in April, 2000, as a Professor of International Academy of Media Arts and Sciences in Ogaki City, Gifu Prefecture. His current research interests include human computer interaction and intellectual agents. Dr. Nakayama is a member of the Information Processing Society of Japan, the Institute of Electronics, Information and Communication Engineers of Japan, the Japanese Psychological Association and Human Factors and Ergonomics Society.