Representation for Knot-Tying Tasks

Jun Takamatsu, Takuma Morita, Koichi Ogawara, Hiroshi Kimura, and Katsushi Ikeuchi, *Fellow, IEEE*

***Abstract*—The learning from observation (LFO) paradigm has been widely applied in various types of robot systems. It helps reduce the work of the programmer. However, the applications of available systems are limited to manipulation of rigid objects. Manipulation of deformable objects is rarely considered, because it is difficult to design a method for representing states of deformable objects and operations against them. Furthermore, too many operations are possible on them. In this paper, we choose knot tying as a case study for manipulating deformable objects, because the *knot theory* is available and the types of operations possible in knot tying are limited. We propose a knot planning from observation (KPO) paradigm, a KPO theory, and a KPO system.**

***Index Terms*—Knot-tying task, learning from observation (LFO), movement primitives, Reidemeister moves, state representation (P-data).**

I. Introduction

AROUND 1990, the learning from observation (LFO) paradigm was proposed, and since then, robotics researchers have been focusing on it [1]. In this paradigm, a robot system makes an observation of human tasks, recognizes them, and generates a program to reproduce the tasks. Thus, the LFO paradigm reduces human efforts in programming by effectively employing information obtained from observation. There are many research methods based on the LFO paradigm [2]–[5].

To recognize tasks, information obtained from various observational means, for example, vision system, sensor grove, and virtual reality, is automatically transformed into abstract information. By using such information, a robot can reproduce a task under a different environment from one which a robot encounters when observing. Therefore, the abstraction is the key to composing the system.

Up to now, methods following the LFO paradigm have considered only an assembly task of rigid objects, or a task where there is little or no physical interaction with the environment (for example, reach motion and gestures). The tasks to manipulate deformable objects have not been considered, because it is difficult to transform information on such tasks into abstract information.

In this paper, tasks of tying knots in one rope are examined to study manipulation of deformable objects. In manipulation of deformable objects, their abstract shape representation is essential. The reason we selected knot-tying tasks for our case study is that we can design this representation for the required tasks using the *knot theory* [6]. This is a novel trial, and we still need to determine how to apply our conclusions to other kinds of manipulation of deformable objects.

To reproduce a knot-tying task, we have developed a *knot planning from observation* (KPO) system. It observes a human tying a knot through a stereovision system [7]. The observation is recognized as a sequence of so-called movement primitives in [1]. By duplicating the sequence, the system can tie a similar knot.

In designing a KPO system, two problems with respect to the transformation into abstract information need to be solved initially.

- How should the state of a knot be represented?
- What kinds of movement primitives should be defined?

This paper solves these two problems by using knowledge of the knot theory [6], as mentioned above. The theory is a mathematical study that investigates the characteristics of tangled loops.

Considering state representation, abstraction of a knot shape, that is, topological information of a knot,[1] is preferred to a knot shape itself. In robot execution, it is difficult to precisely realize a desired knot shape. Fortunately, it is sufficient to realize only the topological information of a knot, not its shape, in order to tie a knot.

Although there are various kinds of representations of the topological information, it is important to be able to reverse such a representation to one example of a knot shape. This reversibility is very useful in solving a so-called path-planning problem. We propose to employ *P-data* representation [8] and prove the reversibility using the knot theory [6] and the graph theory.

To define movement primitives in advance eases the implementation of a KPO system for various robots, which have different configurations (for example, degrees of freedom (DOFs) of manipulators, end-effectors, and so on), as mentioned in [9]. Considering efforts of the implementation, fewer movement primitives are preferred. However, a sufficient number of them should be prepared for reproducing any knot-tying tasks. We define four kinds of movement primitives; three of the four are introduced from the knot theory [6]. And

[1]In this paper, a topological information of a knot means an adjacency relation among intersections, edges, and faces on a knot projection mentioned later.

Manuscript received September 30, 2004; revised March 21, 2005. This paper was recommended for publication by Associate Editor G. Oriolo and Editor F. Park upon evaluation of the reviewers' comments. This work is supported in part by CREST, the Japan Science and Technology Corporation (JST), and in part by the Ministry of Education, Culture, Sports, Science and Technology under a Grant-in-Aid for Scientific Research on Priority Areas (C) 16016218. This paper was presented in part at the IEEE International Conference on Robotics and Automation, Taiwan, R.O.C., September 2003.

J. Takamatsu and K. Ogawara are with the Institute of Industrial Science, University of Tokyo, Tokyo 153-8505, Japan (e-mail: j-taka@cvl.iis.u-tokyo.ac.jp; ogawara@cvl.iis.u-tokyo.ac.jp).

T. Morita is with Sony Corporation, Tokyo 108-6201, Japan (e-mail: takuma.morita@jp.sony.com).

H. Kimura is with Graduate School of Information Systems, University of Electro-Communications, Tokyo 182-8585, Japan (e-mail: hiroshi@kimura.is.uec.ac.jp).

K. Ikeuchi is with the Graduate School of Interdisciplinary Information Studies, University of Tokyo, Tokyo 153-8505, Japan (e-mail: ki@cvl.iis.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TRO.2005.855988

we show the sufficiency of the movement primitives for reproducing any knot-tying tasks. Note that using our proposed state representation and movement primitives [7], Wakamatsu *et al.* have proposed a method to automatically plan the knot-tying process from only the beginning and the end of knot states [10].

In this paper, we concentrate on the recognition phase, but we briefly describe our current implementation of observation and execution phases, which are still under implementation. The paper is organized as follows. Section II describes related works on the LFO paradigm and on manipulation of deformable objects. Section III illustrates an outline of a whole KPO system and outlines of current observation and execution phases. Section IV describes how to abstractly represent a knot state. To obtain such representation, first, a shape of a knot is represented as a projected image on a 2-D plane. Next, its topological information is represented in P-data representation. Section V defines three types of *Reidemeister moves* and the *Cross move* as movement primitives for knot-tying tasks. Three types of Reidemeister moves are employed to investigate an equivalence of two knots in the knot theory [6]. Section VI describes a method to specify an appropriate movement primitive from a P-data transition, that is, change of P-data caused by the movement primitive, and shows the results of applying two types of knots (a bowline knot and a bow knot) to our proposed method. Section VII summarizes this paper. Finally, we add an Appendix that proves the reversibility of P-data.

II. Related Works

First, we survey past research methods on the LFO paradigm. Dillmann *et al.* roughly classified them based on the difference of task representation, which is strongly dependent on the transformation into abstract information, as follows: 1) trajectory; 2) object position; and 3) environment effects (abstract task representation) [5]. Most research methods employ a trajectory [11], [12] or object position [13] as task representation. However, they are ineffective for representing a knot-tying task, which is a kind of manipulation of deformable objects. There is no guarantee that repetition of the same trajectory realizes the same result, because of differences in the initial state of such an object. Furthermore, to reproduce the same object position (shape) is extremely difficult, and may not even be at all practical. That is why representation of knot-tying tasks in a KPO system is based on environment effects.

As far as we know, only research on assembly tasks employs environment effects [2], [4]. In assembly tasks, it is well known that topological contact relations (for example, a vertex of an object A is in contact with a face of an object B) are very useful for expressing the process of the tasks. That is, topological contact relations correspond to the states of the tasks. However, we do not yet know how to represent other kinds of tasks based on environment effects. We therefore need to propose appropriate state representations for knot-tying tasks.

As for research on deformable objects, many methods to deal with objects like a blade spring, of which physical behavior can be expressed by Hooke's law about spring force, have been proposed [14]. Recently, several methods have been proposed to model deformable objects like a rope using the *finite-element method* [15], and to plan manipulation of deformation of such an object using the model [16].

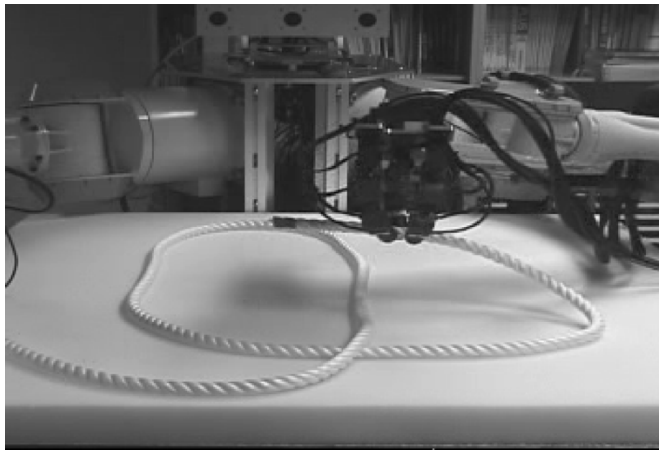


Fig. 1. Goal of our KPO system. A robot duplicates the observed knot-tying task while putting a rope on a table.

Regarding research on manipulation of a rope, Inaba and Inoue proposed a hand–eye manipulation system [17]. Although they concentrate on visual feedback for success of this manipulation, they do not consider how to generate the motion to execute the manipulation.

Hopcroft *et al.* proposed a higher language to express a knot-tying task [18]. The compiler converts such a language into a robot command using visual information. It permits only the motion that one of two terminals of a rope crosses over or under a part of a rope. The limitation forces a robot to unnaturally tie a bow knot, which is a complicated knot.

As for research on state representation of a knot, Wolter and Kroll proposed a method to express a knotted rope using a graph representation [19]. This research method is well suited to express the location of a knot, but not suitable to express the process of tying a knot.

As mentioned above, there is the knot theory [6] to investigate characteristics of tangled loops. As an application of the theory, Yamada *et al.* proposed a method to define a state of a loop on a Cat's Cradle [20], which is a traditional Japanese play to design various figures from a simple loop. Each state on the Cat's Cradle is equivalent to a trivial knot, that is, it cannot be distinguished by the conventional knot theory. To distinguish it, they improved a so-called polynomial invariant.

III. Outline of KPO System

A. Outline of the Whole System

The current goal of a KPO system is to tie a knot in a rope that is placed on a flat surface, such as a table, as shown in Fig. 1. We decided to lay the rope on a surface to ease implementation of the execution phase. Tying a knot in the air is our final goal, but this raises various difficult problems, such as firm grasp of a rope and grasp-position planning for maintaining knot topology in the air. A task demonstrator performs the target task in the same situation, that is, to tie a rope on a table so that a robot can easily obtain knot projections described later. The surface is the projection plane.

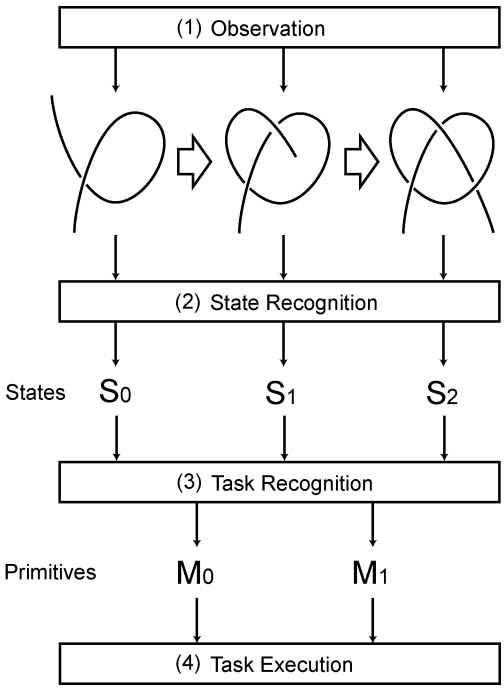


Fig. 2. Outline of a KPO system.

In a KPO system, a robot observes, recognizes, and reproduces a knot-tying task through the following steps.

1) *Observation:* Obtain sequential images of demonstrator's knot-tying motion from a vision system and extract a rope's shape from each image.
2) *State Recognition:* Recognize a knot state at each moment from the shape.
3) *Task Recognition:* Specify an appropriate movement primitive from a knot-state transition.
4) *Execution:* Reproduce the same knot tying by sequentially invoking the movement primitives.

Fig. 2 illustrates the process.

In *Observation*, a KPO system obtains intensity and disparity images using a stereovision system, and extracts a rope's shape from the images using various techniques of image processing [7].

In *State Recognition*, it converts extracted rope shapes into knot states. In *Task Recognition*, it compares two temporally continuous knot states and specifies an appropriate movement primitive. Here, a movement primitive is an often-appeared and essential movement for reproducing any knot-tying tasks. A KPO system has a sufficient number of movement primitives for the reproduction in advance; that means the system can convert any knot-tying task by a demonstrator into a sequence of movement primitives.

In *Execution*, a KPO system reproduces the same knot tying by duplicating the sequence. To do this, it should acquire parameters necessary for the duplication, including the current rope's properties, using a vision system, a force/torque sensor, and other tools, if needed.

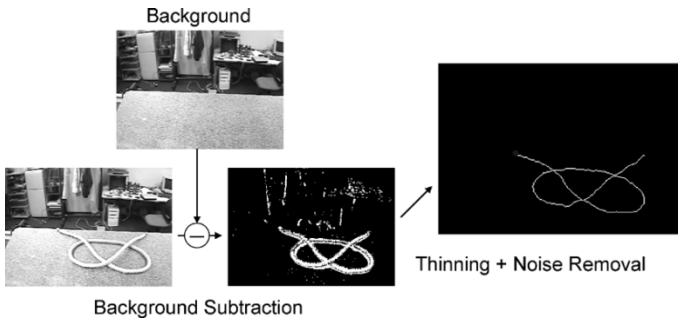


Fig. 3. Steps of extraction of a rope's shape.

B. Outline of Observation Phase

In our current implementation, we assume that a demonstrator shows each state of a knot without occlusion by his/her hands to a robot. He/she is forced to manually segment a whole knot tying into several states. If tracking of a rope could be completely realized, one could also realize automatic segmentation. The idea of the Snakes method [21] may be useful in implementing the tracking.

Our current system extracts a rope's shape as follows (see Fig. 3).

1) *Image Acquisition:* Capture red–green–blue (RGB) and disparity images from the upper surface through a vision system. From the disparity data, we calculate a 3-D position at each point of the captured image in the robot coordinates.
2) *Background Subtraction:* Subtract background data from the image to extract only a knot. The field of view is assumed to be fixed for the subtraction.
3) *Thinning:* Change the knot into a thin line by using a Hilditch filter and remove some errors.
4) *Graph-Representation Extraction:* Find intersections and terminals in the knot by counting the neighboring points for all pixels of the thin line image. And find all segments by following neighboring pixels from the intersection or the terminal until reaching another.
5) *Reordering:* Reorder the segments and points (intersections and terminals) so that they are in an order that can be traced from one terminal of the knot to the other.
6) *Vertical Position Extraction:* By using disparity data obtained from the vision system, determine vertical positions of the intersections.

It is easy to convert information about this knot shape into a knot state described later.

C. Outline of Execution Phase

The execution phase has a robot tie knots by translating each movement primitive into robot commands. The conversion requires two types of information: task parameters and skill parameters. Task parameters indicate how to execute each step of knot tying, that is, the type and parameters (for example, operated segment IDs) of movement primitives. In contrast, skill parameters indicate how to execute each step **well**.

For example, consider the case of tying a simple knot, as shown in Fig. 4. If you only consider making a loop, you may make it in the middle of a rope [see Fig. 4(a)]. However, if you further consider the next operation, that is, to move a terminal

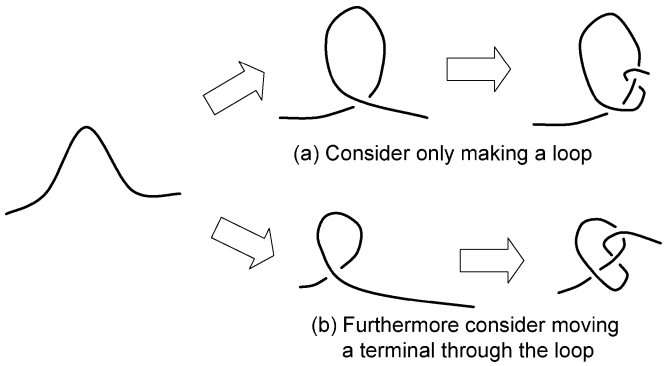


Fig. 4. Skill parameters.

through the loop, you may make it far away from the terminal [see Fig. 4(b)]. In this case, the kind of operation is a task parameter, and the position of the loop is a skill parameter. Note that the type of movement primitive uniquely determines what kinds of skill parameters are necessary.

Task parameters have been already obtained in the recognition phase. Our current system retrieves skill parameters from a human demonstration that is regarded as the most suitable method for knot tying. We might also consider a method to obtain skill parameters through reconstruction of a knot from P-data. While this gives us a more flexible choice and can be theoretically implemented (see reversibility of our state representation in the Appendix), we must solve a new problem of choosing the most suitable skill parameters for knot tying. This problem is still unsolved, but it is quite interesting to consider its solution.

When representing skill parameters, we use two levels of parameters: object-level parameters and robot-level parameters. Object-level parameters give reference to knot-relative positions and directions according to the rope's geometric structure. We defined such parameters, because parameters depending on metric properties of a knot are apt to change and cannot be determined before manipulation. Dependence on metric properties would force a knot to conform to the shape. This is too restrictive. On the other hand, robot-level parameters give reference to knot positions and directions using the robot coordinates, in other words, the metric properties of a knot. For the actual robot commands, metric information is essential.

The execution phase works for each movement primitive in the following steps.

1) *Object-level Parameter Acquisition:* Determine object-level parameters by using the movement primitive and the human demonstration.
2) *Present Knot-State Acquisition:* Obtain the present shape of the rope that the robot will manipulate.
3) *Robot-level Parameter Acquisition:* Acquire robot-level parameters by using object-level parameters and the present knot shape.
4) *Robot Command Conversion:* Generate and execute a sequence of robot commands using robot-level parameters.

D. Concept of Designing a KPO System

In the LFO paradigm, the method to automatically generate robot motion through the aforementioned steps is general and well established. Employing abstract task information rather than directly employing motion that is seen while doing a task enables a robot to flexibly adapt to a different situation. Especially in knot-tying tasks, because there is no guarantee that repetitions of the same motion realize the same result, the flow of the steps that a KPO system employs is quite natural and reasonable. Duplication using the abstraction requires us to obtain various concrete information (the rope's properties), just as on execution.

In actuality, when one recognizes and reproduces some knot-tying task from observation, one naturally follows the flow. Upon recognizing it, one obtains abstract representation of a task (for example, a terminal of a rope moves through a loop), without considering parametric information like the rope's properties. And when reproducing it, one obtains new parametric information, which may be different from that obtained in the recognition state. Following the flow, one can reproduce the same knot-tying task using any rope with different properties.

In short, the key for implementing a KPO system consists of the following three items.

- How should a state of a knot be represented?
- What kinds of movement primitives should be defined in advance?
- How should an appropriate movement primitive be specified from a knot-state transition?

IV. Knot-State Representation

In this section, we describe a method to illustrate a knot in a 3-D space on a 2-D plane, and a method to convert the knot image on a 2-D plane into P-data representation, which is a representation of a knot state.

A. Knot Projection

In the knot theory, a knot is defined as a simple[2] closed curve, that is, tangled loop, with no thickness in a 3-D space. In this paper, we redefine a knot as a simple open curve, that is, a simple curve with two terminals.

In the same manner as the knot theory does, we illustrate a knot in a 3-D space on a 2-D plane by applying orthographic or perspective projection from an appropriate viewpoint, as shown in Fig. 5. In such a projected image, several intersections appear. At each intersection, the farthest strand relative to the viewpoint is drawn with gaps around the intersection.

For recoverability of topological information of a knot from a projected image, this drawing must satisfy the following conditions:

- each intersection is a point intersection, not a line intersection;
- one strand crosses the other strand at all intersections;
- any three strands do not intersect at the same point;
- any terminals are not on a strand.

By applying such a drawing style, we can correctly recover topological information of a knot from this drawing. This drawing is defined as a knot projection. Fortunately, such a projected image

[2]Simple means that a curve has no branch.

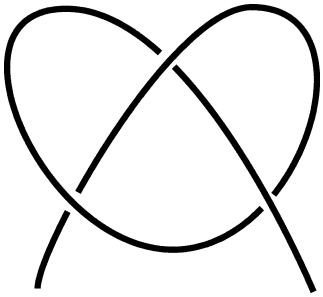


Fig. 5. Knot projection.

can always be obtained from any knot by selecting an appropriate viewpoint [22].

Each intersection integrates points on the nearer strand and on the farther strand. If we need to distinguish these two points, we refer to them as upper and lower intersections, respectively. In this paper, the word *point* indicates either an intersection or a terminal, and a segment indicates a strand surrounded by two points.

B. P-Data

In designing knot-state representation, the following two conditions are required:

1) a knot state is an abstract data structure that does not depend on parametric information;
2) the transformation from a knot projection to a knot state is reversible.

The first condition should be satisfied because it is unfavorable to translate the state by locally moving an intersection, locally transforming the shape of a strand, and so on. For example, two knot projections, as shown in Fig. 6, are different from each other in the case that the first condition is not satisfied. These two knots are essentially equivalent with respect to knot tying. The second condition should be satisfied for solving a so-called path-planning problem. In actuality, we prove the reversibility in the Appendix. We choose the P-data representation [8],[3] because it is simple but satisfies both conditions. In the P-data representation, a knot projection is converted into P-data through the following process.

1) At first, we choose one of two terminals. We refer to chosen and not-chosen terminals as *start* and *end* terminals.
2) From the start terminal, we follow the knot projection until reaching the end terminal. If we encounter an intersection, we number it according to the order of encounter. Therefore, each intersection has two numbers.
3) We refollow the knot projection. When we encounter an intersection, we record both numbers. We also determine a sign of an intersection and its vertical position (upper or lower intersection).
4) Finally, we code them using a number (referred to as an *attribute*) as follows: 1) upper/-; 2) lower/-; 3) upper/+; 4) lower/+.

[3]Reference [8] calls this presentation perfect P-data, but in this paper, we simply call this P-data. P-data presentation in [8] deals with just a tangled loop.

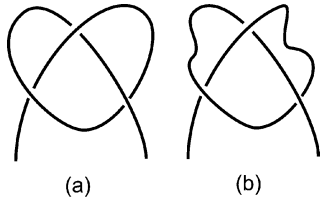


Fig. 6. Two equivalent knot projections.

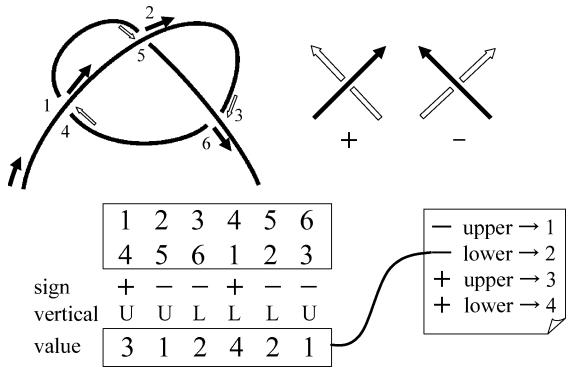


Fig. 7. P-data.

The sign is determined by the sign of the following equation:

$$(\vec{l}_{\text{upper}} \times \vec{l}_{\text{lower}}) \cdot \vec{e_z}$$

where $\vec{l}_{\text{upper}}$ and $\vec{l}_{\text{lower}}$ are the following directions of the nearer and farther strands, respectively, at their intersection, and e_z is a unit vector parallel to the z axis (upward from this paper). If the nearer strand passes from the left to the right of the farther strand, the sign is plus. And if the nearer strand passes from the right to the left, the sign is minus.

From the above process, P-data for the knot in Fig. 7 is obtained as follows:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 3 & 1 & 2 & 4 & 2 & 1. \end{array}$$

Here, we define several terminologies and functions.

Definition 1: The ith intersection is defined as the intersection of which the encountering order is i, in the process of converting a knot projection into P-data. In the same manner, the ith segment is defined as the segment of which the encountering order is i in the process.

Definition 2: Given P-data P, $n(P)$ returns the number of columns of the P-data, that is, twice the number of intersections in the knot projection.

Definition 3: $\sigma(i \mid P)$ returns the number assigned to the ith intersection other than i. When it is clear what the state P is, we briefly denote it as $\sigma(i)$.

Definition 4: $\mathrm{attr}(i \mid P)$ returns the attribute 1) upper/-, ... of the ith intersection. When it is clear what the state P is, we briefly denote it as $\mathrm{attr}(i)$.

For example, in the above P-data, $n(P) = 6$, $\sigma(2 \mid P) = 5$, and $\mathrm{attr}(3 \mid P) = 2$. If $\sigma(i \mid P) = j$ is satisfied, $\sigma(j \mid P) = i$ must be satisfied, that is, the commutative law is always satisfied.

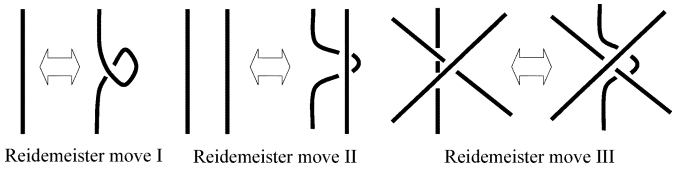


Fig. 8. Reidemeister moves.

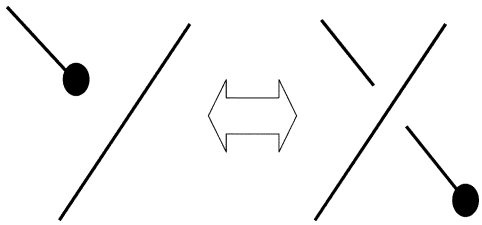


Fig. 9. Cross move.

V. Definition of Movement Primitives

In this section, we first define four movement primitives: three types of Reidemeister moves [22] and the Cross move. And we illustrate how to transform a knot shape after executing each movement primitive. Next, we illustrate sufficiency of the movement primitives for reproducing any knot-tying tasks.

At first, we define characteristics of the movement primitives as follows:

- they move only one segment at a time;
- they directly translate P-data to another P-data without intermediate P-data.

A. Four Movement Primitives

1) Three Types of Reidemeister Moves: Two knots[4] are equivalent if one knot can be deformed to the other knot without cutting it. Note that a knot is allowed to stretch and shorten.

Reidemeister proved that any equivalent knot can be obtained from the knot by stretching, shortening, and finite repetitions of three types of moves referred to as Reidemeister moves [22]. Note that stretching and shortening do not translate P-data, but these three types of Reidemeister moves do. Fig. 8 shows the Reidemeister moves.

The Reidemeister move I adds or removes one intersection by creating or destroying a simple loop. The Reidemeister move II adds or removes two intersections, while a strand crosses to another strand. The Reidemeister move III moves a strand through an intersection. Therefore, the number of intersections is not changed. Note that these moves satisfy the characteristics mentioned above.

From characteristics of movement primitives, one segment only moves at a time. Such moves are classified into three types as follows:

- move across some intersection;
- move across some segment, not across any intersection;
- move not across any segment or intersection.

These moves correspond to Reidemeister moves III, II, and I, respectively.

2) Cross Move: A knot in the knot theory is a simple closed curve, that is, a tangled loop, and has no terminal. However, an actual rope has two terminals, so that it is allowed to move terminals without cutting a rope itself. Such a move is not considered in the knot theory. Therefore, we additionally define a Cross move as a movement primitive. The Cross move adds or removes one intersection, while one terminal of a rope crosses some segment, as shown in Fig. 9.

[4]These knots are ones in the knot theory, that is, tangled loops.

B. Sufficiency of the Movement Primitives

Every P-data transition, that is, change of P-data caused by a movement primitive, can be divided into the following two types:

1) a terminal of a rope crosses some segment;
2) no terminal crosses any segment.

In the first case, every transition can be realized by the Cross move. In the second case, every transition can be realized by several repetitions of three types of Reidemeister moves, and this fact is easily introduced from Reidemeister's proof about knot equivalence.[5] Therefore, these four movement primitives can reproduce any knot-tying tasks.

In actuality, by repetitions of these four movement primitives, one can make typical knots, including the overhand knot, eight knot, bowline knot, harness hitch, bow tie, single-loop bow, two-half knot, and taut-line hitch.

VI. Specification of Appropriate Movement Primitives From P-data Transitions

In this section, we describe a method to specify an appropriate one of the four movement primitives from a P-data transition. In the following, P_t denotes the P-data obtained from a knot projection at time t. Without any loss of generality, we assume that a knot projection before the transition has fewer intersections than a knot projection after the transition, that is

$$n(P_{t-1}) \leq n(P_t).$$

P-data depends on the selection of a start terminal; the selection is important when considering a method for specifying an appropriate movement primitive. Now we assume that a start terminal is not changed before and after the transition.

A. Reidemeister Move I

Fig. 10 shows an example of Reidemeister move I. After applying Reidemeister move I to a knot projection at time $t-1$, one intersection is added to the knot projection at time t. Therefore, (1) must be satisfied

$$n(P_t) = n(P_{t-1}) + 2. \tag{1}$$

Now we assume that Reidemeister move I is applied to the ith segment (the third segment in the case, as shown in Fig. 10),

[5]However, there is no guarantee that such a transition is realized by only one of the three types of Reidemeister moves (see the definition of movement primitives). Although the movement primitive which consists of more than one of them may be discovered, it is easy to implement the system to recognize and execute the primitive, because it is equivalent to several repetitions of three types of Reidemeister moves.

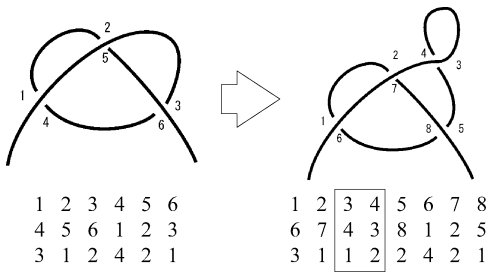


Fig. 10. Change of P-data under Reidemeister move I.

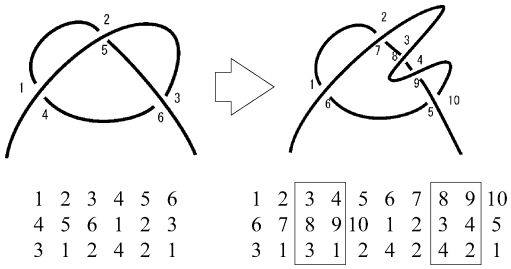


Fig. 11. Change of P-data under Reidemeister move II.

where $1 \leq i \leq n(P_{t-1}) + 1$. The additional intersection has two continuous numbers. Therefore, (2) must be satisfied

$$\sigma(i \mid P_t) = i + 1. \tag{2}$$

Because situations of other intersections are not changed, when we remove the ith and $i+1$th columns from P_t and reorder the intersection numbers, that is, subtract the intersection numbers from $i+2$ to $n(P_t)$ by two, P_t must become equal to P_{t-1}. Such removing and reordering is defined as $R_I(P_t, i)$.

Conversely, considering the reversibility from P-data to a knot projection, Reidemeister move I must occur at the ith segment $(1 \leq i \leq n(P_{t-1}) + 1)$ at time $t-1$, when $\sigma(i \mid P_t) = i + 1$ and $R_I(P_t, i) = P_{t-1}$ are satisfied. Using this fact, we can specify when and where Reidemeister move I occurs from a P-data transition. In the case shown in Fig. 10, because $\sigma(3 \mid P_t) = 4$ and $P_I(P_t, 3) = P_{t-1}$ are satisfied, we can specify that Reidemeister move I occurs at the third segment at time $t-1$.

B. Reidemeister Move II

Fig. 11 shows an example of Reidemeister move II. After applying Reidemeister move II to a knot projection at time $t-1$, two intersections are added in the knot projection at time t. Therefore, (3) must be satisfied

$$n(P_t) = n(P_{t-1}) + 4. \tag{3}$$

Now we assume that Reidemeister move II is applied to the ith and jth segments (the third and sixth segments in the case shown in Fig. 11), where $1 \leq i < j \leq n(P_{t-1}) + 1$. In the process of converting a knot projection into P-data, the two additional intersections are continuously encountered. Therefore, (4) or (5) must be satisfied

$$\sigma(i \mid P_t) = j + 2 \cap \sigma(i+1 \mid P_t) = j + 3 \tag{4}$$

$$\sigma(i \mid P_t) = j + 3 \cap \sigma(i+1 \mid P_t) = j + 2. \tag{5}$$

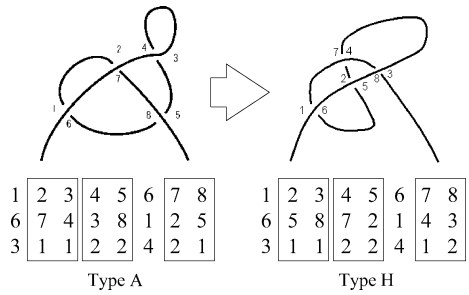


Fig. 12. Change of P-data under Reidemeister move III.

Furthermore, these vertical positions (upper/lower) must be the same, and these signs must be different. Therefore, (6) must be satisfied

$$|\mathrm{attr}(i \mid P_t) - \mathrm{attr}(i+1 \mid P_t)| = 2. \tag{6}$$

Because situations of other intersections are not changed, when we remove the ith, $(i+1)$th, $(j+2)$th, and $(j+3)$th columns from P_t and reorder the intersection numbers, i.e., subtract the intersection numbers from $i+2$ to $j+1$ by two and from $j+4$ to $n(P_t)$ by four, P_t must become equal to P_{t-1}. Such removing and reordering is defined as $R_{\mathrm{II}}(P_t, i, j)$.

Conversely, considering the reversibility from P-data to a knot projection, Reidemeister move II must occur between the ith and jth segments $(1 \leq i < j \leq n(P_{t-1}) + 1)$ at time $t-1$, when (4) or (5), (6), and $R_{\mathrm{II}}(P_t, i, j) = P_{t-1}$ are satisfied. Using this fact, we can specify when and where Reidemeister move II occurs from a P-data transition. In the case shown in Fig. 11, because $\sigma(3 \mid P_t) = 8$, $\sigma(4 \mid P_t) = 9$, $|\mathrm{attr}(3 \mid P_t) - \mathrm{attr}(4 \mid P_t)| = 2$, and $R_{\mathrm{II}}(P_t, 3, 6) = P_{t-1}$ are satisfied, we can specify that Reidemeister move II occurs between the third and sixth segments at time $t-1$.

C. Reidemeister Move III

Fig. 12 shows an example of Reidemeister move III. After applying Reidemeister move III to a knot projection at time $t-1$, no intersection is added in the knot projection at time t. Therefore, (7) must be satisfied

$$n(P_t) = n(P_{t-1}). \tag{7}$$

The focus is on three important segments for Reidemeister move III. Now, we name them Segment A, B, and C by the following rules.

- Segment A is surrounded by two upper intersections.
- Segment B is surrounded by one upper and one lower intersections.
- Segment C is surrounded by two lower intersections.

Let Segments A–C correspond to the ith, jth, and kth segments, respectively, where $2 \leq i, j, k \leq n(P_{t-1}) - 1$. Note that i, j, and k are permitted to be in any order. These three segments compose three intersections. Each intersection has two numbers of the set $\{i-1, i, j-1, j, k-1, k\}$.

Here, there are two ways to assign numbers $i-1$, i to two intersections of Segment A, depending on a direction of following a knot in the process of converting a knot projection into P-data. In the same manner, there are two ways to assign numbers at

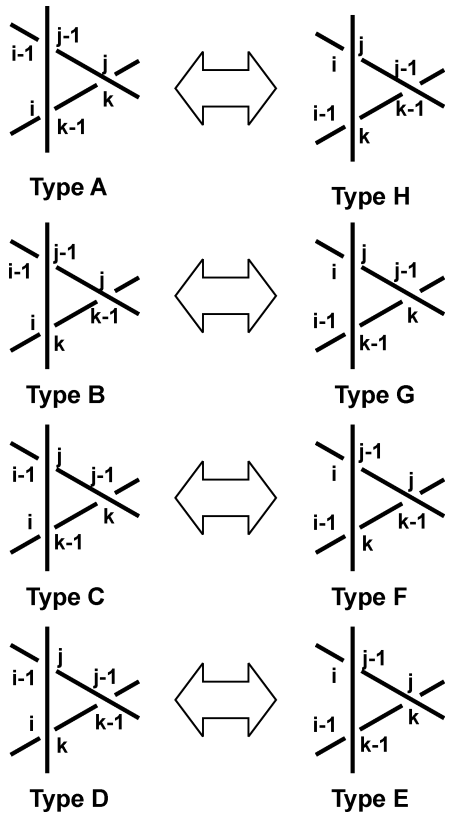


Fig. 13. Eight kinds of assignment of intersection numbers.

Segments B and C. Therefore, the following eight assigning styles exist (see Fig. 13):

A) $\sigma(i-1)=j-1, \sigma(i)=k-1, \sigma(j)=k$;
B) $\sigma(i-1)=j-1, \sigma(i)=k, \sigma(j)=k-1$;
C) $\sigma(i-1)=j, \sigma(i)=k-1, \sigma(j-1)=k$;
D) $\sigma(i-1)=j, \sigma(i)=k, \sigma(j-1)=k-1$;
E) $\sigma(i-1)=k-1, \sigma(i)=j-1, \sigma(j)=k$;
F) $\sigma(i-1)=k, \sigma(i)=j-1, \sigma(j)=k-1$;
G) $\sigma(i-1)=k-1, \sigma(i)=j, \sigma(j-1)=k$;
H) $\sigma(i-1)=k, \sigma(i)=j, \sigma(j-1)=k-1$.

Considering possibility of transitions among the eight, the following four transitions are permitted:

- Type A $\leftrightarrow$ Type H;
- Type B $\leftrightarrow$ Type G;
- Type C $\leftrightarrow$ Type F;
- Type D $\leftrightarrow$ Type E.

Because situations of another intersections are not changed, when we remove the $i-1$th, ith, $j-1$th, jth, $k-1$th, and kth columns from both P_t and P_{t-1}, they become equal to each other. Such removing is defined as $R_{\mathrm{III}}(P,i,j,k)$.

Considering reversibility from P-data to a knot projection, the algorithm to decide when and where Reidemeister move III occurs is as follows.

1) Search the ith, jth, and kth segments which compose a triangle before and after the transition, where $2 \le i,j,k \le n(P_{t-1})-1$.
2) Decide on the assigning style for these three segments before and after the transition.
3) Check the legality of the transition in **Fig. 13**.
4) Check $R_{\mathrm{III}}(P_t,i,j,k) = R_{\mathrm{III}}(P_{t-1},i,j,k)$.

In the case shown in Fig. 12, we can specify that Reidemeister move III occurs among the third, fifth, and eighth segments at time $t-1$. Focusing on the three segments, the assigning styles are Types A and H at time $t-1$ and t, respectively, and $R_{\mathrm{III}}(P_{t-1},3,8,5) = R_{\mathrm{III}}(P_t,3,8,5)$ is satisfied.

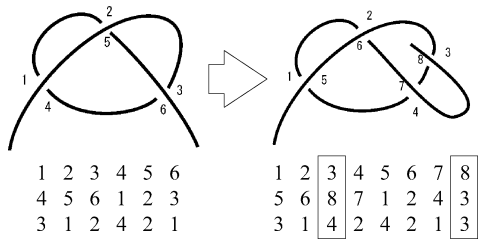


Fig. 14. Change of P-data under the Cross move.

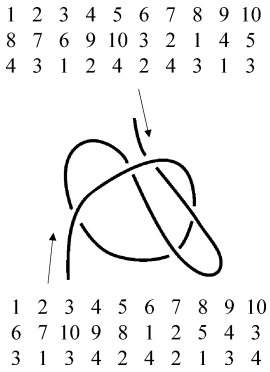


Fig. 15. Two equivalent P-data.

D. Cross Move

Fig. 14 shows an example of the Cross move. After applying the Cross move to a knot projection at time $t-1$, one intersection is added in the knot projection at time t. Therefore, (8) must be satisfied

$$n(P_t) = n(P_{t-1}) + 2. \tag{8}$$

The Cross move is executed by crossing a start or an end terminal across some segment. Therefore, it is easy to specify which segment the Cross move is applied from, $\sigma(1 \mid P_t)$ or $\sigma(n(P_t) \mid P_t)$.

Now we assume that the Cross move is applied to a start terminal and the ith segment. The additional intersection has the number $i+1$. Therefore, $i+1 = \sigma(1 \mid P_t)$ must be satisfied. When we remove the first and $i+1$th columns from P_t and reorder the intersection numbers, that is, subtract the intersection numbers from two to i by one, and from $i+2$ to $n(P_t)$ by two, P_t becomes equal to P_{t-1}. Such removing and reordering is defined as $C_s(P_t)$.

Next, we assume that the Cross move is applied to an end terminal and the jth segment. The additional intersection has the number j. Therefore, $j = \sigma(n(P_t) \mid P_t)$ must be satisfied. In the same fashion, when we remove the last and jth columns from P_t and reorder the intersection numbers, that is, subtract

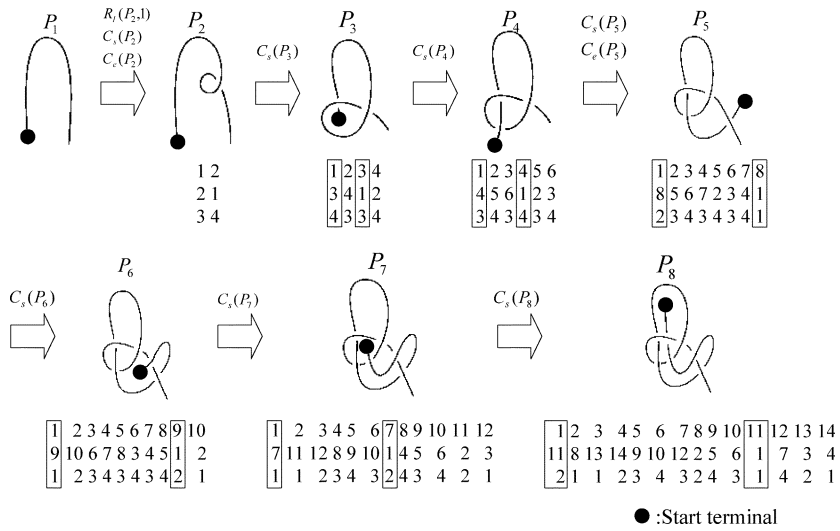


Fig. 16. Analysis of the bowline knot.

the intersection numbers from $j+1$ to $n(P_t)-1$ by one, P_t becomes equal to P_{t-1}. Such removing and reordering is defined as $C_e(P_t)$.

Conversely, considering the reversibility from P-data to a knot projection, the Cross move must occur between a start terminal and the $(\sigma(1 \mid P_t) - 1)$th segment (an end terminal and the $\sigma(n(P_t) \mid P_t)$th segment), when $C_s(P_t) = P_{t-1}$ ($C_e(P_t) = P_{t-1}$) is satisfied. Using this fact, we specify when and where the Cross move occurs. In the case shown in Fig. 14, because $C_e(P_t) = P_{t-1}$ is satisfied, we can specify that the Cross move occurs between an end terminal and the third $\sigma(n(P_t) \mid P_t) = 3$ segment.

E. Ambiguity With Respect to Selection of a Start Terminal

Until the previous section, we assumed that a start terminal in the process of converting a knot projection into P-data is not changed before and after the transition. Now we consider the elimination of this assumption.

As shown in Fig. 15, a knot projection is converted into two different sets of P-data according to the difference of selection of a start terminal. We define these two sets of P-data as equivalent P-data. Between P-data P and the equivalent P-data $P_{\rm eq}$, the following relationships should be satisfied for all i.

Proposition 1: $\sigma(N(P) - i + 1 \mid P_{\rm eq}) = N(P) - \sigma(i \mid P) + 1$.

Proof: The ith intersection in the case of the conversion into P-data P is the $(N(P) - i + 1)$th intersection in the case of the conversion into P-data $P_{\rm eq}$. ■

Proposition 2: $\mathrm{attr}(N(P) - i + 1 \mid P_{\rm eq}) = \mathrm{attr}(i \mid P)$.

Proof: Trivial. The vertical position, that is, upper or lower, of the ith intersection in the case of the conversion into P-data P is equal to the $(N(P) - i + 1)$th intersection in the case of the conversion into P-data $P_{\rm eq}$. Furthermore, the sign of the ith intersection in the case of the conversion into P-data P is equal to that of the $(N(P) - i + 1)$th intersection in the case of the conversion into P-data $P_{\rm eq}$. The following direction of every strand turns back, but

$$(\vec{l}_{\rm upper} \times \vec{l}_{\rm lower}) \cdot \vec{e_z} = (-\vec{l}_{\rm upper} \times -\vec{l}_{\rm lower}) \cdot \vec{e_z}$$

is always satisfied. ■

Using *Propositions 1* and *2*, we can easily convert P-data into the equivalent P-data. Because P-data is reversible to a knot projection, as proved in the Appendix, we can resolve the ambiguity by applying the algorithm to specify an appropriate movement primitive to both P-data and the equivalent P-data.

F. Application to Several Kinds of Knot-Tying Tasks

Fig. 16 shows the result of the analysis of a bowline knot. Note that we correctly select a start terminal that is unchanged, and is shown by the black point in the figure in every knot projection. First, a KPO system converts each knot projection into P-data. The number array under each knot projection is P-data. Next, we apply our proposed method for recognizing knot-tying tasks. In each transition, the number of columns increases by two; therefore, the Cross move ($C_s(*)$ or $C_e(*)$ in the figure) or Reidemeister move I ($R_{\rm I}(*)$ in the figure) may occur.

In all transitions, a KPO system can specify appropriate movement primitives. Among them, in the first and fourth transitions, a KPO system specifies more than one movement primitive. Actually, in the fourth transition, we can find that the Cross move $C_s(P_4)$ is more appropriate by observing the transformation of a knot projection. Because P-data does not include parametric information, the system cannot select one of the two movement primitives. However, both of them can realize the transition; therefore, this ambiguity is not a problem. In the first transition, that is the same.

Fig. 17 shows the efficiency of a KPO system with respect to the ambiguity of the selection of a start terminal. In this example, we select the wrong one of two terminals as a start terminal in the third, sixth, and seventh knot projections. The wrong selection usually leads to the result that a KPO system cannot specify any movement primitive, in spite of the fact that a movement primitive to realize the transition exists.

In the fifth transition, the system cannot specify any movement primitive from the transition P_5 to P_6; however, the system can specify the Cross move $C_s(P_{\rm eq6})$, which is an appropriate one, from the transition P_5 to $P_{\rm eq6}$, which is equivalent P-data of P_6. As a result, we find that the selection of a start terminal is wrong in the sixth P-data.

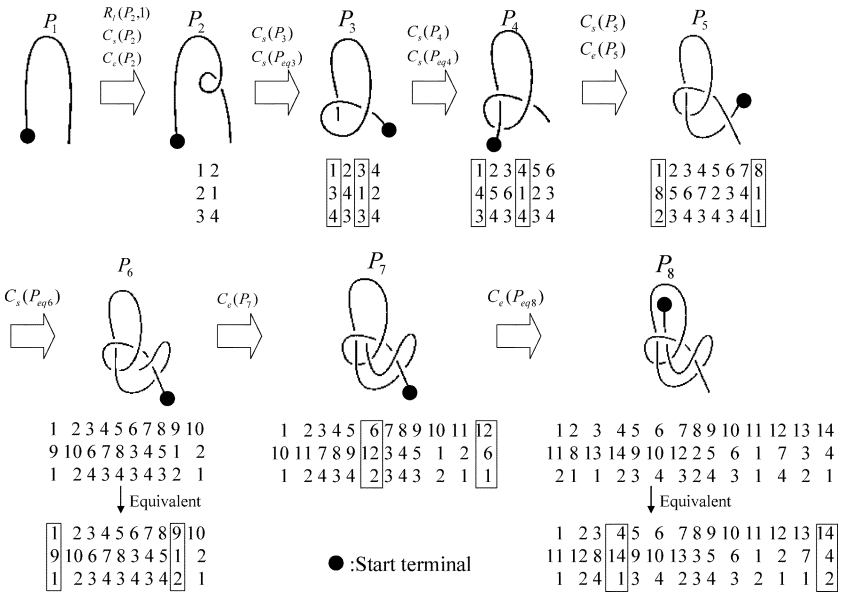


Fig. 17. Analysis of the bowline knot when some start terminals are not correctly selected (P_{eq*} is equivalent P-data of P_*).

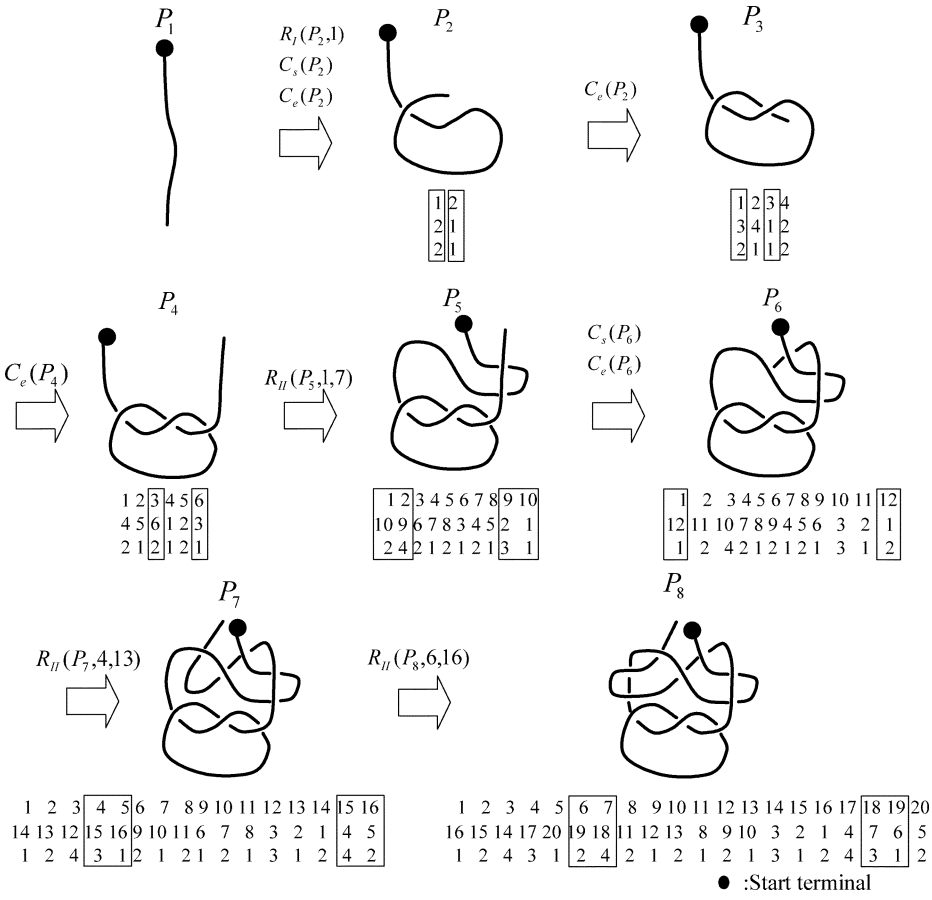


Fig. 18. Analysis of the bow knot.

In the sixth transition, the system can specify an appropriate movement primitive from the transition P_6 to P_7; however, the selection was wrong in the sixth P-data. Therefore, we find that the selection is wrong in the seventh P-data.

In the second transition, the system cannot decide if the selection is correct or wrong, because P_3 is equal to $P_{\mathrm{eq}3}$, which is equivalent P-data of P_3. Note that the system can specify an appropriate movement primitive in this case.

Fig. 18 shows the result of the analysis of a bow knot. This time, we correctly select a start terminal in every knot projection. In this example, Reidemeister move II ($R_{\mathrm{II}}(*)$ in the figure) is employed to realize the fourth, sixth, and seventh tran-

sitions. It is certain that these transitions can be realized by several repetitions of the Cross move. However, it is natural that these transitions, especially for the seventh transition, are realized by Reidemeister move II. The method in [18] considers only the Cross move, that is, it forces a robot to tie a bow knot in an unnatural way.

VII. Summary

In this paper, we have proposed a novel method to recognize knot-tying tasks with one rope. We applied the knot theory, represented a knot state in P-data representation, and converted a knot-tying task into a sequence of movement primitives which we defined.

First, we introduced P-data representation to describe each knot state on a process of knot tying. P-data representation has several efficient characteristics, which are as follows.

- P-data representation is an abstract data structure for representing topological information only. It ignores slight differences, such as a small difference of position of an intersection and local differences in the shape of a knot, that are unimportant to the process of knot tying.
- Topological information of a knot projection can be reversed from only P-data representation. That is important in solving a so-called path-planning problem.

In the Appendix, we prove reversibility by showing the method to convert irreducible P-data into a planar three-connected graph.

Next, we defined the following movement primitives: Cross move and Reidemeister moves I, II, and III. From Reidemeister's proof, a transition between two equivalent knots which are tangled loops can be realized by shortening, stretching, and repetitions of the three types of Reidemeister moves. The proof guarantees that a transition where any terminal of a rope does not cross a strand can be realized by repetitions of them. However, our target rope has two terminals, and the Cross move is necessary to realize a transition where some terminal crosses a strand. We illustrated the sufficiency of the movement primitives for realizing any knot-tying tasks, and confirmed that well-known knot-tying tasks could be represented as a sequence of the movement primitives.

Then, we proposed a method to specify an appropriate movement primitive from a P-data transition. A P-data transition is uniquely decided from the type and parameters (such as, to which segment is a movement primitive applied?) of a movement primitive. Conversely, we can specify an appropriate movement primitive by a whole search against all candidates. At the same time, we also solved a problem of ambiguity of the selection of a start terminal using equivalent P-data.

Finally, we showed examples of analyses of typical knot-tying tasks. A KPO system can specify appropriate movement primitives from P-data transitions, even if the selection of a start terminal is wrong. Especially in a bow knot, the system can naturally tie the knot by using Reidemeister move II.

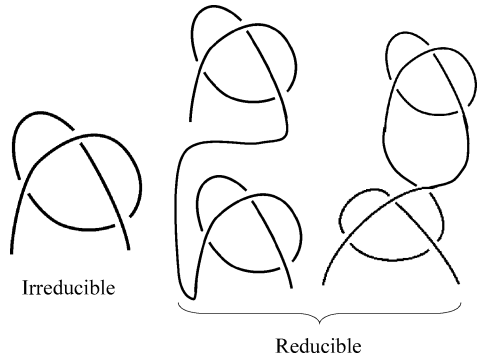


Fig. 19. Knot projections which generate irreducible and reducible P-data.

Appendix
Reversibility

In this section, we prove reversibility from P-data to a knot projection using the graph theory. At the beginning, we define several terminologies and notations.

Definition 5: A *reducible* P-data is defined as one which satisfies the following condition, where $1 \leq i < j \leq n(p) \cap (i \neq 1 \cup j \neq n(p))$ is satisfied:

$$\exists i, j \quad \forall k, i \leq \sigma(k) \leq j (i \leq k \leq j). \tag{9}$$

An *irreducible* P-data is one which is not reducible.

Fig. 19 shows knot projections which generate irreducible and reducible P-data, respectively. It is easy to find out that a knot projection which generates reducible P-data consists of some knot projections which generate irreducible P-data. Therefore, if we can reverse any irreducible P-data to a knot projection, we can reverse any P-data by appropriately connecting the knot projections. From now on, we concentrate on the reversibility of irreducible P-data.

Definition 6: A *knot graph* is defined as a graph of which vertices and edges correspond to points and segments of a knot projection, respectively. However, the first and the last segments are not included in the edges of the knot graph, and the two terminals are not included in the vertices of the knot graph. Note that in a knot graph, an attribute of each intersection is disregarded.

Fig. 20 shows an example of a knot graph. Note that it is easy to convert a knot graph into a knot projection, if an attribute of each intersection is given.

Now, we introduce some propositions necessary to prove the reversibility.

Proposition 3: A graph which is dual to any knot graph is a connected graph, where the external face and its connections to inner faces are disregarded in a dual graph.

Fig. 21 shows an example of a dual graph.

Proof: If the external face and its connections are not disregarded, the dual graph is obviously a connected graph. Therefore, we have only to prove that the disregard does not cause a break in the connectivity. Now we assume that the dual graph is not a connected graph. Consider only all edges and vertices adjacent to the external face in the original graph. This graph is referred to as an external graph. The external graph is obviously a connected graph, because our target knot is created by only one rope. If the external graph is equivalent to one loop, the dual graph is obviously a connected graph.

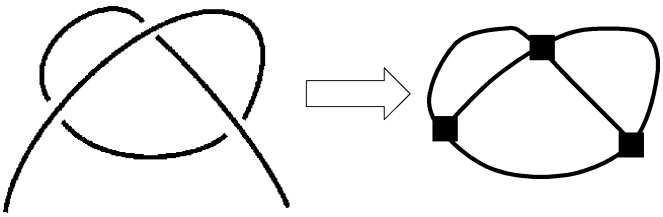


Fig. 20. Knot projection to knot graph.

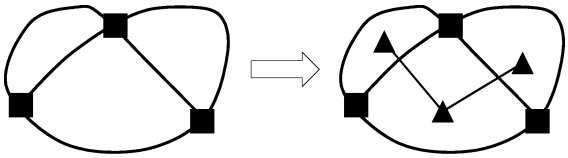


Fig. 21. Dual graph of a knot graph.

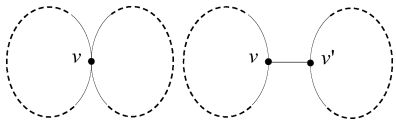


Fig. 22. Necessary condition for disconnect of a dual graph.

Because the external graph is not equivalent to one loop, we assume that two loops exist (see Fig. 22). Two loops are connected to each other by a vertex v or one edge e adjacent to two vertices v and v', as shown in Fig. 22.

Consider the process of converting a knot projection into P-data. When following a knot projection from a start terminal, the following three cases are possible:

1) entering and exiting the loop from a vertex v;
2) beginning from a start terminal inside the loop and exiting the loop from a vertex v;
3) entering the loop from a vertex v and stopping at an end terminal inside the loop.

Let a vertex v have two numbers i and j, where $i < j$. In the first case, the encountering order k for all vertices inside the loop satisfies $i < k < j$. This contradicts that the P-data which is obtained from a knot projection corresponding to the original graph is irreducible. In the second case, the encountering order k for all vertices inside the loop satisfies $1 \leq k < j$ and $j \neq n(P)$. This also contradicts. In the third case, the encountering order k for all vertices inside the loop satisfies $i < k \leq n(P)$ and $i \neq 1$. This also contradicts. In the case that more than two loops exist, we can introduce the contradiction that the original graph is not irreducible by the same way. ■

Proposition 4: A graph obtained by applying barycentric subdivision to a knot graph is a three-connected graph.

The graph, as shown at the bottom in Fig. 23, is the result of applying barycentric subdivision to the graph shown at the top left in the figure. The subdivided graph is obtained by performing the following steps (see Fig. 23).

1) Append a vertex to the middle of each edge and connect it to the two vertices adjacent to the edge. (Note that all original edges are removed.)

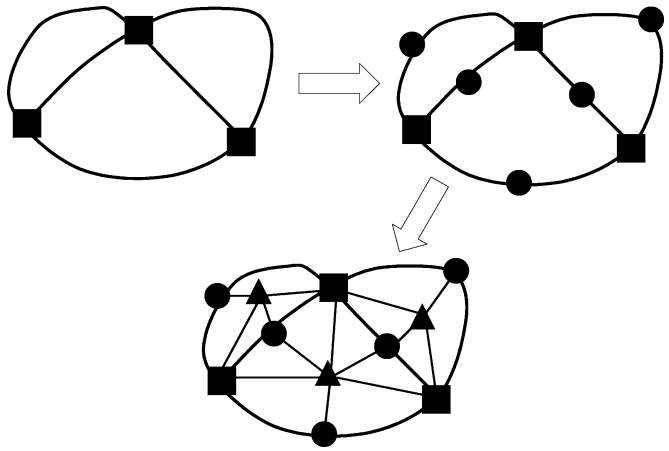


Fig. 23. Barycentric subdivision.

2) Append a vertex to the barycenter of each face, except for the external face, and connect it to all vertices on the boundary of the face.

Proof: Now we try to judge the connectivity of any two vertices in a graph G' which is made from the subdivided graph by removing two vertices. A path between two vertices which are on the same face of the original graph (the graph before the subdivision) always exists for any G', because the graph which corresponds to the face is obviously a wheel graph, which is a three-connected graph.

Next, consider the connectivity of two vertices which are not on the same face. Let the two vertices be on the faces F_1 and F_n. From *Proposition 3*, a sequence $F_1 \rightarrow \cdots \rightarrow F_n$ is obtained, where F_i is adjacent to F_{i+1} with a shared edge for all i. Because each face is a wheel graph, that is, a three-connected graph, and three entrances exist between any two adjacent faces, a path between the two vertices always exists for any G'. Therefore, the subdivided graph is a three-connected graph. ■

Proposition 5: A knot graph obtained from irreducible P-data is uniquely embeddable into a sphere S^2. The way to embed is topologically unique except for its reverse.

Proof: From *Proposition 4*, a graph obtained by applying barycentric subdivision to a knot graph is a three-connected graph, and obviously a planar graph. Unique embedability of a planar three-connected graph has already been proved by Whitney [23]. ■

Fortunately, we resolve the ambiguity of the reverse using an attribute of an intersection. We obtain a knot projection from irreducible P-data through the following steps.

1) Extract all faces by searching shorter loops using the breadth-first search from a knot graph. Each edge is twice employed as a component of a face.
2) Select one of the faces as the external face.
3) Apply barycentric subdivision.
4) Embed it using the algorithm proposed in [8].

There is one caution on the process of extracting all faces, as follows. Consider the vertex which has four adjacent edges. Let the vertex correspond to the ith and jth intersections. This time, these edges correspond to the $i-1$th, ith, $j-1$th, and jth segments. A loop including the $i-1$th and ith segments does not compose a face, because one strand crosses the other strand

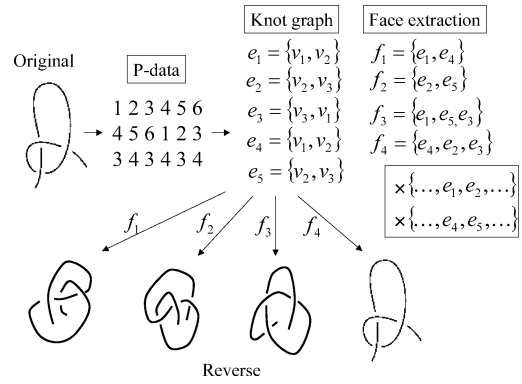


Fig. 24. Convert a knot projection into a knot projection through P-data conversion.

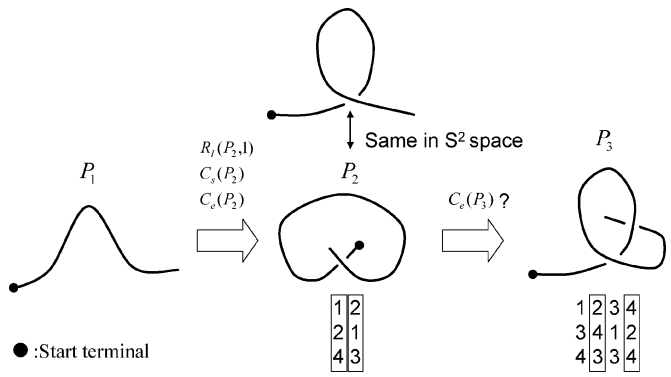


Fig. 25. Erroneous specification by making wrong use of difference between an S^2 space and an R^2 space.

in all intersections, as mentioned in Section IV-A. In the same manner, a loop including the $j-1$th and jth segments does not compose a face.

In Fig. 24, we actually show the result of reversibility of P-data by converting a knot projection into a knot projection through P-data conversion. First, we convert the knot projection as shown at the top left in the figure into P-data. As a result, we obtain the following P-data:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 3 & 4 & 3 & 4 & 3 & 4. \end{array}$$

Next, we convert such irreducible P-data into a knot projection again, as in the algorithm described above. First, we convert P-data into a knot graph, which has three $(= 6/2)$ vertices $\{v_1, v_2, v_3\}$ and five edges

$$\{e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_3\}, e_3 = \{v_3, v_1\}, e_4 = \{v_1, v_2\}, e_5 = \{v_2, v_3\}\}.$$

Second, we extract the following four faces by searching shorter loops using the breadth-first search:

$$f_1 = \{e_1, e_4\}, f_2 = \{e_2, e_5\}, f_3 = \{e_1, e_5, e_3\}, f_4 = \{e_4, e_2, e_3\}.$$

From the caution on extracting all faces, we should not regard $\{e_1, e_2, e_3\}$ and $\{e_3, e_4, e_5\}$ as a face, because these sets include e_1 and e_2, or e_4 and e_5, simultaneously.

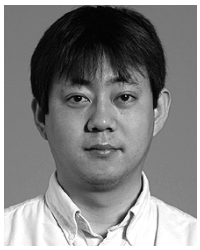
When we select a face f_4 as an external face, we obtain a knot projection, as shown at the bottom right in Fig. 24, through applying the barycentric subdivision and embedding it using the algorithm proposed in [8]. That is an equivalent knot projection to the original one. By selecting one of another three faces as an external face, we obtain anther three knot projections. Note that these three knot projections look quite different from the original knot, because they are illustrated in an R^2 space. Of course, they are equivalent in an S^2 space.

Through this example, one may have the following question. Is P-data representation not available for an R^2 space, which is the usual demonstration space? The answer is yes, in respect to specification of movement primitives, which is our main purpose. P-data is not available only when an operated segment moves across the point at infinity which is necessary to generate one-to-one correspondence between an R^2 space and an S^2 space. In the normal use of our specification algorithm, the point can be assumed to be infinitely far away; there is no possibility to move the segment across the point at all. However, a vicious demonstrator can cause the algorithm to introduce an erroneous movement primitive, as shown in Fig. 25. If one wants to avoid such an error, one should employ both P-data and the external segment information for the specification.

REFERENCES

[1] S. Schaal, "Is imitation learning the route to humanoid robots?," *Trends Cognit. Sci.*, vol. 3, pp. 233–242, 1999.
[2] K. Ikeuchi and T. Suehiro, "Toward an assembly plan from observation—Part I: Task recognition with polyhedral objects," *IEEE Trans. Robot. Autom.*, vol. 10, no. 3, pp. 368–385, Jun. 1994.
[3] J. Takamatsu, H. Tominaga, K. Ogawara, H. Kimura, and K. Ikeuchi, "Extracting manipulation skills from observation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, vol. 1, 2000, pp. 584–589.
[4] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Trans. Robot. Autom.*, vol. 10, no. 6, pp. 799–822, Dec. 1994.
[5] R. Dillmann and M. Bordegoni, "Learning robot behavior and skills based on human demonstration and advice: The machine learning paradigm," in *Proc. Int. Symp. Robot. Res.*, 1999, pp. 229–238.
[6] C. C. Adams, *The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots*. San Francisco, CA: Freeman, 1994.
[7] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot planning from observation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2003, pp. 3887–3892.
[8] M. Ochiai, S. Yamada, and E. Toyoda, *Computer Aided Knot Theory*. Tokyo, Japan: Makino Shoten, 1996.
[9] T. Hasegawa, T. Suehiro, and K. Takase, "A model-based manipulation system with skill-based execution," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 535–545, Oct. 1992.
[10] H. Wakamatsu, A. Tsumaya, K. Shirase, E. Arai, and S. Hirai, "Knotting/raveling manipulation of linear objects," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2003, pp. 3156–3161.
[11] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Netw.*, vol. 9, no. 8, pp. 1281–1302, 1996.
[12] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 1398–1403.
[13] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi, "Extraction of essential interactions through multiple observations of human demonstrations," *IEEE Trans. Ind. Electron.*, vol. 50, no. 4, pp. 667–675, Aug. 2003.
[14] H. Nakagaki, "Insertion task of a flexible beam or a flexible wire," *J. Robot. Soc. Japan*, vol. 16, no. 2, pp. 159–162, 1998.
[15] T. Hisada, "Finite element modeling," *J. Robot. Soc. Japan*, vol. 16, no. 2, pp. 140–144, 1998.

[16] T. Wada, B. J. McCarragher, H. Wakamatsu, and S. Hirai, "Modeling of shape bifurcation phenomena in manipulations of deformable string objects," *Int. J. Adv. Robot.*, vol. 15, no. 8, pp. 833–846, 2001.
[17] M. Inaba and H. Inoue, "Hand–eye coordination in rope handling," *J. Robot. Soc. Japan*, vol. 3, no. 6, pp. 32–41, 1985.
[18] J. E. Hopcroft, J. K. Kearne, and D. B. Krafft, "A case study of flexible object manipulation," *Int. J. Robot. Res.*, vol. 10, no. 1, pp. 41–50, 1991.
[19] J. Wolter and E. Kroll, "Toward assembly sequence planning with flexible parts," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, pp. 1517–1524.
[20] M. Yamada, R. Budiarto, H. Seki, and H. Itoh, "Topology of Cat's Cradle diagrams and its characterization using knot polynomials," *J. Inf. Process. Soc. Japan*, vol. 38, no. 8, pp. 1873–1582, 1997.
[21] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comput. Vis.*, pp. 321–331, 1988.
[22] K. Reidemeister, *Knot Theory*. Moscow, ID: BCS Associates, 1983.
[23] H. Whitney, "Congruent graphs and the connectivity of graphs," *Amer. J. Math.*, vol. 54, pp. 150–168, 1932.



Jun Takamatsu received the Ph.D. degree in computer science from the University of Tokyo, Tokyo, Japan, in 2003.

Currently, he is a Postdoctoral Researcher with the Institute of Industrial Science, University of Tokyo. His research interests include learning from observation, task planning using feasible motion analysis, and 3-D shape analysis.

Takuma Morita received the Master's degree in information and communication engineering from the University of Tokyo, Tokyo, Japan, in 2003.

Currently, he is with Sony Corporation, Tokyo, Japan, as a Software Engineer.

Koichi Ogawara received the Ph.D. degree in information and communication engineering from the University of Tokyo, Tokyo, Japan, in 2002.

Currently, he is a Postdoctoral Researcher with the Institute of Industrial Science of the University of Tokyo. His research interests include vision-based object recognition, manipulation task understanding from multimodal input, intelligence in acquisition of skilled behavior, and imitation learning.

Hiroshi Kimura was born in Hiroshima, Japan in 1961. He received the B.E., M.E., and Dr.E. degrees in mechanical engineering from the University of Tokyo, Tokyo, Japan, in 1983, 1985, and 1988, respectively.

Currently, he is an Associate Professor with the Graduate School of Information Systems, University of Electro-Communications, Tokyo, Japan. His research interests are the dynamics of machinery and intelligence.

Katsushi Ikeuchi (M'78–SM'95–F'98) received the Ph.D. degree in information engineering from the University of Tokyo, Tokyo, Japan, in 1978.

Currently, he is a Professor with the Institute of Industrial Science of the University of Tokyo. After being with the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, Cambridge, for three years, the Electrotechnical Laboratory of the Ministry of International Trade and Industries for five years, and the School of Computer Science of Carnegie-Mellon University, Pittsburgh, PA, for ten years, he joined the University of Tokyo in 1996. He is Editor-in-Chief of the *International Journal of Computer Vision*, and is on the Editorial Board of the *Journal of Computer Vision, Graphics*.

Dr. Ikeuchi was selected as a Distinguished Lecturer of the IEEE Signal Processing Society for 2000–2001. He has served as the Program/General Chairman of several international conferences, including 1995 IEEE-IROS, 1996 IEEE-CVPR, and 1999 IEEE-ITSC. He has received several awards, including the David Marr Prize in computational vision, and IEEE Robotics and Automation Society's K.-S. Fu Memorial Best Transaction Paper award. In addition, in 1992, his paper, "Numerical Shape from Shading and Occluding Boundaries," was selected as one of the most influential papers to have appeared in the *Artificial Intelligence Journal* within the past ten years.