Vol.2013-CVIM-187 No.52
2013/5/31

Reproducing Human Manipulative Movements on a Robotic Hand using Tangle Topology

P. Vinayavekhin[1,a) S. Kudoh[2,b) J. Takamatsu[3,c) Y. Sato[1,d) K. Ikeuchi[1,e)

Abstract: This paper proposes a methodology to control these robotic hands through human demonstration. The method focuses on a dexterous manipulation of a hand referred to as regrasping. It allows a robot to observe a human perform regrasping movements, recognise, and finally reproduce the regrasping movement. The proposed method is based on a representation referred to as tangle topology. Tangle topology is a representation that derived from a numerical invariant that describes a relation between two curves called Gauss Linking Integral (GLI). When hand and a manipulated object are considered as strands, it allows regrasping movements to be perceived as a change of tangle relation over time. Using this topological information, a task model (task primitives and skill parameters) for recognising and mapping human regrasping movement to a robotic hand is defined. The proposed methodology is validated by reproducing a regrasping movement of a pen-like object in a robotic hand.

1. Introduction

There are many ways for a robot to interact with the world. One of the ways is to use a *hand*, which is biologically inspired by human. A hand is an important human body part. Throughout a day, human uses its hands numerous times, either with prehensile or non-prehensile movements [1], to alter the environment. A robot that possesses a hand with such dexterity as human hand would be able to interact well with its physical surroundings.

Many anthropomorphic robotic hands has been developed in recent years. With the advancement in technologies that used to construct them, these hands have become very complex in term of their hardware functionalities and have also become more and more similar to a human hand. Unfortunately, there are only a few evidences that has displayed the capabilities of their usages in everyday environment, except for those of the teleoperation application. One of the main issues is the lack of efficient approach to control these robot hands.

This work focuses an approach to control a robotic hand by observing how human move their hand. This is based on the assumption that if the robotic hands were to build to emulate how human hands are maneuvered and manipulate objects, the best way to control the robot hand is to simply imitate human. Although the imitation approach might seem reasonable, in reality, it is not as trivial and straightforward as it may sound. The most critical question is how to model and represent human hand movement, so that it can be applied to robot hands. Since a general human hand movement is very complicated and highly sophisticated, this work will focus on one type of human dexterous manipulative movement called *regrasping* or sometimes referred to a *in-hand manipulation*.

Regrasping is an ability of a hand to change its grasping posture with an object by moving the fingers or, in other words changing the contact location of fingers. Humans usually have more than one grasping posture for one type of tool. A human makes a decision on which type of grasping posture to use based on their intention of how to use the grasped object at that particular moment [2]. For example, an artist grasps a pen or a paintbrush differently depending on whether they want to draw or measure the length of the scene, or people also grasp a hammer differently depending on whether they want to strike or pull out a nail, or people grasp mobile phones differently depending on whether they want to press a button or talk on the phone. This makes regrasping an important function for robotic hands, if they were to be used to interact with tools created for humans.

In this paper, a novel method on how to control a robotic hand to regrasp object is presented. It is based on a *Learning from Observation (LFO)* [3, 4] paradigm. In this paradigm, a robot to observe a human perform regrasping movements, recognise, and finally reproduce the regrasping movement. A task model is composed of two fundamental components: *task primitives* and *skill parameters*. Task primitives are an abstract representation used to represent an observed movement. Skill parameters are a set of knowledge required to map each task primitive to the robot and execute it. A task model of the proposed method is based on a representation referred to as tangle topology. Tangle topology is a representation that derived from a numerical invariant

[1] Institute of Industrial Science, The University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 JAPAN
[2] Graduate School of Information Systems, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 182-8585 JAPAN
[3] Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, NARA 630-0192 JAPAN
a) pv-milk@cvl.iis.u-tokyo.ac.jp
b) kudoh@is.uec.ac.jp
c) j-taka@is.naist.jp
d) yoshi@cvl.iis.u-tokyo.ac.jp
e) ki@cvl.iis.u-tokyo.ac.jp

that describes a relation between two curves called Gauss Linking Integral (GLI). When hand and a manipulated object are considered as strands, it allows regrasping movements to be perceived as a change of tangle relation over time. The tangle relation is described by an attribute called *writhe matrix*. Writhe matrix is categorised into two types, which distinguish a contact relation between the hand and object. Using this topological information, both elements of the task model, task primitives and skill parameters are defined to recognise and map regrasping movement to a robotic hand.

2. Related Works

Giving two grasping configurations, initial and final grasping configurations, regrasping planning is defined as a problem of finding trajectories of hand and its joint angles to move from one configuration to the other. Several approaches have been proposed in the literature to complete a regrasping movement in a robotic hand. They can be divided into three major categories: a.) teleoperation, b.) automatic programming, and c.) learning from human.

A teleoperation is the most intuitive approach to control a robotic hand. It is usually referred to as a master-slave architecture of a control system, where a human directly controls a slave robot hand to manipulate an object through a master device. There are applications in many areas for this control scheme. For example, a robotic surgical system in medical [5], a maintenance humanoid robot in space [6], a mobile manipulation robot in hazardous areas [7].

Automatic programming is the most common approach in the literature. In this approach, a regrasping planning is treated as a search problem. First, a configuration space for a robot hand and objects are defined, together with the models of their relation, e.g. a contact or a friction model. Then various constraints, including collision avoidance, grasp equilibrium etc. are mapped to confine the configuration space. Finally, the algorithms are proposed to search a path within the space so as to connect two grasp postures/configurations together. Some work on this approach with various system set-ups can be found in [8–11].

The original idea of imitating human hand movement has originated from Cutkosky [2]. The author proposed that an appropriate grasp for a particular tool should depend mainly on how the tool is intended to be used, and this information can be obtained from human operator directly. Based on this notion, various methods regarding grasp planning from observation have been proposed [12–14].

On the other hand, the research field of dexterous manipulation from observation has only emerged recently. There are two important components in the dexterous manipulation planning from observation: a.) a representation and recognition phase, b.) a mapping phase. However, same grasp taxonomy might not be a good representation as it is a taxonomy for every kind of objects. For a specific object, the taxonomy could be far too incomplete to represent all hand postures.

Steffen *et al.* [15] use a continuous manifold of hand postures to represent a regrasping movement. They construct a manifold by applying Unsupervised Kernel Regression (UKR) and its modification [16] on finger joint angle space. The method is applied to turning bottle cap in their example. A demonstration is done using various sizes of bottle cap in a virtual environment and the movement of the same robot hand is automatically reproduced with different cap sizes. They extend their UKR representation framework to closed-loop control and use it to imitate the task of swapping Chinese health balls where balls position are used as feedback [17]. The drawback of their framework is that the hands using during demonstration and reproduction must be the same hand.

To overcome the hand structure dependency, the representation is generally derived from a contact information, either on the hand or on the object. Lam *et al.* combined information observed from human demonstration with motion planning algorithms to recover a sequence of contact points on the object that represents the original movement [18]. Human demonstrates a regrasping movement using data-glove and motion tracker. In order to recover contact points in every frames, he starts by recovering initial and final contact points based on observed data from data-glove. Once initial and final contact points are found, contact points on the object of all intermediate frames is generated using motion planning while the demonstrated movement is used to limit the search region of the algorithm. Kondo *et al.* attaches a tactile sensor sheet on an object in order to observe changes in contact state during human regrasping movement [19, 20]. They manage to discover a contact state on a human hand at a particular moment by processing a pressure distribution image. A state transition diagram is created based on these contact states. A target robotic hand is preprogrammed with the movement that changes from one contact state to another. Human regrasping movement is recognised and represented as a transition of contact states using Dynamic Programming algorithm. The transition of contact states is sent as a command to the robotic hand to reproduce the movement. Martins *et al.* came up with a similar approach but a tactile sensor is directly attached to a human hand [21, 22].

It can be seen that for a dexterous manipulation planning from observation, most frameworks only pass a high level knowledge to map to a robotic hand. Other knowledge obtained from the observation has been abstracted out and not used during mapping at all.

3. Outline of RPO System

The goal of a Regrasping Planning from Observation (RPO) system is to teach a robot to regrasp an object using knowledge obtained from human demonstration. In a RPO system, a robot imitates a regrasping movement through the following processes:

(1) *Observation:* is a process of acquiring a target regrasping movement. A sequence of human hand postures performing the movement is captured using sensors.
(2) *Movement Recognition:* is a process of acquiring a necessary knowledge to reproduce the observed movement. The movement is recognised with a pre-defined representation (or model), referred to as a *Task Model*.
(3) *Movement Mapping:* is a process of reproducing the observed movement onto a robotic hand. A movement of the robotic hand is generated to mimic the observed movement

by incorporating the recognised knowledge.

During *Observation*, human regrasping movement is captured using a data glove and a motion tracker. At a particular moment, hand configuration, hand location and object location are captured. A combination of these data is referred to as a grasping posture. Human regrasping movement is observed as a sequence of grasping postures.

Movement Recognition is composed of three sub-procedures: state recognition, task primitive recognition, and skill parameter extraction. States, task primitives and skill parameters are part of a pre-defined Task Model. An observed regrasping movement is recognised into a sequence of grasping states. Grasping state is defined by a combination of tangle relation between fingers and the object. Then, every two consecutive grasping states are compared to determine a task primitive connecting between them. Finally, skill parameters for each task primitive are extracted from the observed movement. These parameters will be used to generate a movement of the same task primitive for a robotic hand during *Movement Mapping*.

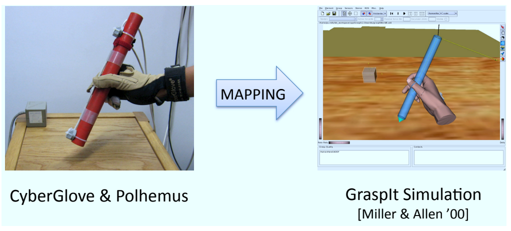
Human regrasping movement is reproduced onto the robotic hand in *Movement Mapping*. Skill parameters obtained from the original movement are refined against constraints which are essential for the movement to be mapped onto a robotic hand. Then, all task primitives are sequentially reproduced in order to deplicate the observed regrasping movement.

3.1 Concept of Designing a RPO System

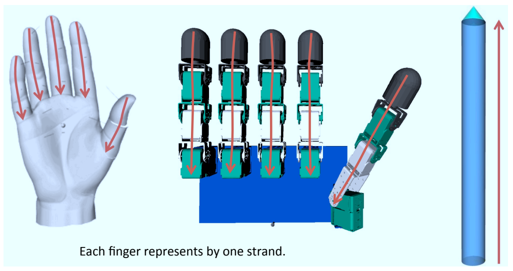
The most crucial component to design a system using LFO paradigm is a Task Model for imitating a task or a movement. A task model is a representation that connects an observation phase and a mapping phase together. An observed movement is recognised into a representation defined by a task model, which in turn is used to generate a similar movement on the robot. To define a task model for RPO system, there are three fundamental questions to be answered:

- What are intermediate grasping states (or key frames), and how to represent them?
- What are task primitives for representing a regrasping task? A task primitive represents a movement that connects between two intermediate states.
- What are skill parameters for each task primitive? The skill parameters describe how to execute each task primitive on a target robot hand.

Task model of a RPO system is designed based on a tangle topology. Tangle topology is a geometric property that describes the relation between two strands. Hand and the manipulated object are considered as zero-width strands. This allows each grasping posture to be considered as a tangle relation between strands representing the hand and object. Taxonomy of grasping postures is constructed based on the tangle relations, which in turn, use to classify intermediate grasping states in the regrasping movement. Three task primitives are used to connect between these states, and represent a regrasping movement. Skill parameters for each task primitive are defined based on parameters that used to describe the tangle relation.



(a) Demonstration system used during an *observation* phase.



(b) Hands and an object are represented with zero-width strands.

Fig. 1 Outline of Observation Phrase.

3.2 Outline of Observation Phase

A demonstration system shown in Figure 1(a) is used to capture a regrasping movement. Humans demonstrate a regrasping task by controlling a model in a virtual environment [23] through a *CyberGlove* data glove and *Polhemus* motion tracker. Information from the glove and motion tracker could have been used directly. However, due to its lack of accuracy to illustrate the hand configuration in the real world, an information in a virtual environment is used instead. An alternative approach to demonstrate task directly in the real world environment is to use an optical marker system [24], or markerless hand tracking system [25].

To model a human regrasping movement using tangle topology, the structure of a human hand and a manipulated object must be represented as strands. In a virtual environment, a demonstrating hand and pen are modelled to resemble the real world and broken down into a group of points in three dimensional space in order to extract their strand information. Hand is substituted with a system of five strands. Each strand represents each finger; it starts at the proximal joint and end at the tip or another way around. A manipulated object, in this case a pen-like object, is substituted with a single strand.

4. Movement Representation: Task Model

In this section, a task model used to represent and recognise human regrasping movement is explained. The task model is designed based on the derivation of Gauss Linking Integral (GLI) called *writhe matrix*. Different grasping postures in regrasping movement can be classified into different types of writhe matrices. These writhe matrices change their type according to the change of the contact state and the orientation between fingers and the object during the movement. Task primitives are modelled after the changes of types of writhe matrices during the hand movement. The proposed task model composes of three task primitives: *detaching*, *attaching*, and *crossover*. Each type of writhe matrices can be characterized by a different set of parameters. A combination of these parameters is, in turn, used as skill parameters for each task primitive.

To simplify a complexity of the problem, a proposed task model is designed for a regrasping movement of a pen-like object. It should, however, be applicable with any object that could be represented with one strand. Designing a task model of regrasping movement for an arbitrary object is our final goal, but this raises various difficult issues, such as how the object should be represented by strands, or how to mapping the movement that interacts with more than on strands at the time.

According to what had been described in Section 3.1, in order to define a set of task primitives for a specific task or movement, there must be a taxonomy to classify intermediate grasping states. Then a transition or movement between these states can be classified or grouped into one type of task primitive. The section starts by giving a brief explaination on tangle topology. Then, a grasp taxonomy used to classify human grasping posture is described. Finally, the definition of task primitive and skill parameters are followed respectively.

4.1 Tangle Topology

GLI is a classical measure for entanglement between two curves. It is the number of average crossings between two curves when viewing from all directions. GLI, or sometimes referred to as writhe, for two curves γ_1 and γ_2 can be calculated by

$$GLI(\gamma_1, \gamma_2) = \frac{1}{4\pi}\int_{\gamma_1}\int_{\gamma_2}\frac{d\gamma_1 \times d\gamma_2 \cdot (\gamma_1 - \gamma_2)}{|\gamma_1 - \gamma_2|^3}. \qquad (1)$$

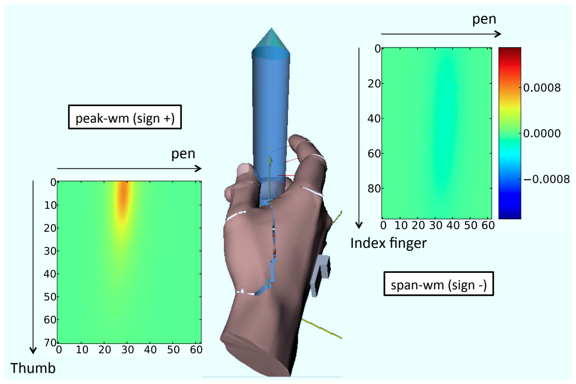
A **writhe matrix** (T) [26] is a by-product from the calculation of GLI between two curves that are represented as a chain of line segments. When two curves are represented by line segments, the writhe between two curves can be estimated by an analytical solution found in [27]. Considering two curves S_1 and S_2, each curve is represented by a chain of line segments of n_1 and n_2 segments respectively. A writhe matrix (T) is a $n_1 \times n_2$ matrix whose element T_{ij} is the GLI between segment i of S_1 and segment j of S_2. GLI between two curves, or sometimes referred to as **total writhe**, is a sum of the GLI of every pair of segment. Note that the terms *total writhe* and *writhe* might be used interchangeably.

4.2 Characterisation of Grasping Postures

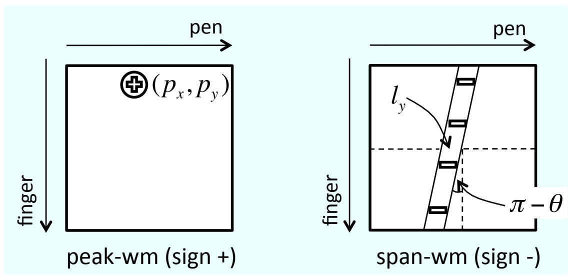
In general, writhe matrix could be in various shapes depending how the two strands are tangled together. Ho and Komura proposed a method to encode writhe matrices of motions of characters, and use it to retrieve similar motions [28]. However, in a very specific case where a finger can barely tangled around the object more than one round, writhe matrices could be examine differently.

Based on an observation and analysis of writhe matrices of various human manipulative movements, e.g. movements in the performance of Japanese tea ceremony, movements of artists during painting, writhe matrices of hand manipulative movement can be broadly classified into two types: peak-type writhe matrix (peak-wm) and span-type writhe matrix (span-wm).

- **Peak-wm** (T^p) is a writhe matrix whose majority of non-zero elements are concentrated at a specific area. This type of writhe matrices is used to represent grasping postures where the corresponding finger is in contact with the object.
- **Span-wm** (T^s), in contrast, is a writhe matrix whose non-zero elements are spread across the matrix. It represents grasping postures where the corresponding finger is located at some distance from the object.



(a) Example of two types of writhe matrices. Writhe matrix between thumb and pen is peak-wm (sign+), while writhe matrix between index finger and pen is span-wm (sign-).



(b) Two different set of parameters used to describe each type of writhe matrix. Peak-wm is described with a location of a peak $\boldsymbol{p} = (p_x, p_y)$, while span-wm is described by a span-line $\boldsymbol{l} = (\theta, l_y)$.

Fig. 2 Two types of writhe matrices are used to characterise a contact relation between fingers and the object. A peak-wm is used to represent a writhe matrix of the finger that is in contact with object, and a span-wm otherwise.

Figure 2(a) illustrates an example for each types of writhe matrices, where the thumb finger of grasp posture in the figure are in contact with a pen and the index finger is not.

The classification of writhe matrices is also comprehensible from Eq. (1). The denominator of GLI is (a power of three of) a distance between two curves or two line segments, when curves are represented with line segments. It greatly affects the value of the writhe. When a finger is in contact with a manipulated object, portions of line segments that represent the finger and the object are close together. This leads to higher writhe values in a specific area of the writhe matrix which is referred to as *peak* area.

For both types of writhe matrices, writhe is not only reflecting the average distance of the finger from the object, but also indicating the orientation of the finger to the object with its value and its sign. In Eq. (1), each strand is assigned with the direction. Sign of writhe is determined by the numerator. Depending on the direction of strands of the finger and the object, writhe changes its sign.

4.3 A Taxonomy of Grasping Postures

The taxonomy for RPO system is developed based on different types of writhe matrices of the hand. To reduce the complexity, the taxonomy will consider grasping postures that relate to only three fingers: thumb, index finger, and middle finger. At the particular moment, writhe matrix of each finger could be in one of the following four states: peak-wm(sign+), peak-wm(sign-), span-wm(sign+), and span-wm(sign-). This mean that, in general, for three finger case, the taxonomy would possess $4^3 = 64$ different states of grasping posture. However, when all other constraints applied, the taxonomy ends up having only 18 feasible states.

Before describing the taxonomy, a notation is given here for a concise descriptive purpose.

- F_T^S : represents writhe matrix of finger F, with a type T, having a sign S.
  - $F \in \{t, i, m\}$: Only thumb, index finger, and middle finger are considered.
  - $T \in \{p, s\}$: Type of writhe matrix can only be either peak-wm or span-wm.
  - $S \in \{+, -\}$: Sign of writhe matrix can only be either plus or minus.
- $t_p^+ i_p^+ m_p^- = m_p^- t_p^+ i_p^+$: Order of fingers makes no difference, but *tim* will be used throughout.

The taxonomy is derived as the followings:

(1) Consider when the direction of the pen is fixed, sign of the writhe matrix indicates the side of the corresponding finger regarding to the pen. Since there are only three fingers involved, two of them must be on the opposite side in order to hold the pen. In other word, all three writhe matrices cannot have the same sign, regardless of their types. This makes all the possibilities to be $2^3 - 2 = 6$ cases, which are $t^+i^+m^-$, $t^+i^-m^+$, $t^-i^+m^+$, $t^-i^-m^+$, $t^-i^+m^-$, $t^+i^-m^-$.

(2) Among three writhe matrices, at least two must have the peak-wm type. Furthermore, those particular two must have the opposite sign. In other word, at least two fingers must be in contact and holding the pen. Therefore, there are three possibilities for each one of six cases listed previously, e.g.
- for $t^+i^+m^-$, all possible states are $t_p^+ i_p^+ m_p^-$, $t_s^+ i_p^+ m_p^-$, and $t_p^+ i_s^+ m_p^-$.

Combining these two constraints, the taxonomy ends up having a totally of $6 \times 3 = 18$ possible states, or 18 categories of grasping posture used to manipulate the pen.

4.4 Task Primitives

Based on the taxonomy, grasping postures are classified into 18 states as described in Section 4.3. In theory, there should be a totally of ${}_{18}P_2 = 306$ transitions between every pair of states. However, when all constraints are taken into consideration (e.g. at least two fingers have to be in contact and hold the pen at all moment), there are only 36 feasible transitions.

Each transition represents a movement that connects between two states of grasping postures. Thirty six transitions can be classified into three categories based on the changes of writhe matrices during the movement. These categories of transition are referred to as task primitives in the RPO system. Original regrasping movement is represented as a sequence of these task primitives, which will then transfer to the robot to inform which movements should be imitated. All three task primitives are described as the following:

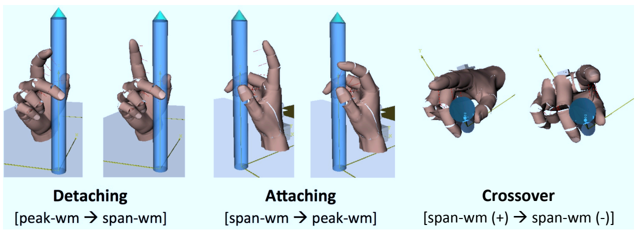


Fig. 3 Grasping postures show an example of task primitives (index finger).

(1) **Detaching** represents a movement where a finger moves away from an object. The writhe matrix changes from peak-wm to span-wm, within the same sign of writhe matrix.

(2) **Attaching**, on the contrary, represents a movement where a finger moves toward an object in order to make a contact. The writhe matrix changes from span-wm to peak-wm, within the same sign of writhe matrix.

(3) **Crossover** represents a movement where a finger changes from one side to another side of the corresponding strand that representing the object. The writhe matrix changes from span-wm to span-wm with the different sign of writhe matrix.

These task primitives are defined based on changes in finger level. This is a reason why 36 transitions of movement can be grouped into three types of task primitive. Two different transitions may have the same task primitives for different fingers, e.g. $t_p^+ i_p^+ m_p^- \to t_s^+ i_p^+ m_p^-$ is detaching on thumb, while $t_p^- i_p^- m_p^+ \to t_p^- i_s^- m_p^+$ is detaching on index finger. Figure 3 shows an example of all three task primitives for the index finger when the manipulated object is a pen.

4.5 Skill Parameters

Skill parameter is also a main component of the task model. It describes how each task primitive should be executed on the robot. Each task primitive requires a different set of skill parameters. In this section, a set of skill parameters for each task primitive is explained. Skill parameters for each task primitive are adopted from writhe matrices of its initial and final grasping states. The seciton will start by outlining a set of parameters used to describe each type of writhe matrix. Then by combining these informations, a set of skill parameters for each task primitive are described.

Ho and Komura used writhe matrix to represent motions between characters [26]. All writhe matrices are treated the same in their system. A topological coordinate is used to describe all writhe matrices. However, this approach is not suitable for representing a regrasping movement as it needs a distintion of a contact relation between hand and object. An evidence for this could be seen in our previous work [29].

In RPO system, writhe matrices are characterised based on their appearances. Writhe matrices are difference, depends whether there is a contact between a finger and the objects as described in Section 4.2. Figure 2(b) shows two types of writhe

Type	Parameter	Description
Peak-wm	$w \in \mathbb{R}$	total writhe
	$p_x \in [-1, 1]$	location of contact on a finger
	$p_y \in [-1, 1]$	location of contact on an object
Span-wm	$w \in \mathbb{R}$	total writhe
	$\alpha \in [0, \pi]$	average orientation between a finger and an object
	$l_y \in [-1, 1]$	average location on an object that is tangled

Table 1 Summary of parameters for each type of writhe matrix.

matrix in an ideal situation where two strands are close or apart from each other. To describe each type of writhe matrix, a different set of parameters are used.

- **Peak-wm** is described by two attributes: writhe w and a location of a peak $\boldsymbol{p} = (p_x, p_y)$. The location of a peak indicates the area where contact occurs. p_x indicates location of contact on the finger, while p_y indicates location of contact on the object. For peak-wm T^p, its parameters will be referred to as $\mathcal{P}(T^p) = (w, p_x, p_y)$.
- **Span-wm** is described by two attributes: writhe w and a span-line $\boldsymbol{l} = (\alpha, l_y)$. The first part of the span-line, slope α, reflects the average orientation between the finger and the object. The other part l_y indicates the average location on the object that is tangled by the finger. A straight line is used to represent the orientation of the finger in this situation because when the finger is not in contact with the object. For span-wm T^s, its parameters will be referred to as $\mathcal{S}(T^s) = (w, \alpha, l_y)$.

As mentioned earlier in the section, skill parameters for each task primitives are taken from its initial and final grasping postures. States of the grasping postures are taken as two of skill parameters of each task primitives. In addition, each task primitives requires parameters that describes writhe matrices of all corresponding fingers and the object. These are different depending on their type of task primitive. These values determine the starting configurations of the robotic hand in a mapping phase.

Table 2 summarizes skill parameters for each type of task primitives. As mentioned in Section 4.4, the major movement may occur in different fingers, even when referring to the same task primitives. This finger is reffered to as a *major moving finger*. It is a finger whose writhe matrix changes its type or its sign to the opposite one between the task primitive. On the other hand, a *minor moving finger* is a finger that may move slightly, but its writhe matrix is kept as peak-wm with the same sign throughout the task primitive. For instance,

- $t_p^+ i_p^+ m_p^- \rightarrow t_s^+ i_p^+ m_p^-$ is considered as detaching, and the major moving finger is thumb. The less are minor moving fingers.
- $t_p^- i_p^- m_p^+ \rightarrow t_p^- i_s^- m_p^+$ is also considered as detaching. However, its major moving finger is index finger, while the less are minor moving fingers.

5. Movement Recognition

A method to recognise a task model is presented in this section. An observed regrasping movement is recognised against the task model and be represented as a sequence of task primitives and skill parameters. A recognition of the task model is divided into two parts: a recognition of task primitives from an observed movement, and an extraction of skill parameters for each task primitives.

Primitive	Parameter	Description
All	s_i	initial states (e.g. $t_p^+ i_p^+ m_p^-$)
(common-	s_f	final states (e.g. $t_s^+ i_p^+ m_p^-$)
parameters)	$\mathcal{P}_{i_1}$	params. of initial peak-wm of 1st minor finger
	$\mathcal{P}_{f_1}$	params. of final peak-wm of 1st minor finger
	$\mathcal{P}_{i_2}$	params. of initial peak-wm of 2nd minor finger
	$\mathcal{P}_{f_2}$	params. of final peak-wm of 2nd minor finger
Detaching	$\mathcal{P}_{i_0}$	params. of initial peak-wm of major finger
	$\mathcal{S}_{f_0}$	params. of final span-wm of major finger
Attaching	$\mathcal{S}_{i_0}$	params. of initial span-wm of major finger
	$\mathcal{P}_{f_0}$	params. of final peak-wm of major finger
Crossover	$\mathcal{S}_{i_0}$	params. of initial span-wm of major finger
	$\mathcal{S}_{f_0}$	params. of final span-wm of major finger

Table 2 Summary of skill parameters for each type of task primitives.

5.1 Recognition of Task Primitives

Task primitives are defined based on the change of type of writhe matrices during hand regrasping movement. In order to recognise a sequence of task primitives of each finger, a sign and a type of all writhe matrices in the regrasping movement have to be identified. A sign of the writhe matrices can be easily determined from the calculation of writhe matrix. A type of writhe matrices are identified by segmenting the movement into many shorter movements, where all writhe matrices in every shorter movements are the same type. The segmentation is done independently for a sequence of writhe matrices of each finger. In other word, this can be perceived as a method to segment a regrasping movement based on the changes of contact relation between hand and object. Though, tactile sensor is not used, but tangle relation between a finger and a pen is used to detect this instead.

In short, the following process is used to segment a regrasping movement. After all writhe matrices of grasping postures in the sequence are calculated, each of them is fitted to a non-singular Bivariate Gaussian distribution. Parameters obtained from the fitting is use to characterise one writhe matrix to the others. The segmentation method makes use of these parameters to decide when the changes in type of writhe matrices occurs.

Consider two strands S_1 and S_2, each strand is represented by line segments of n_1 and n_2 segments respectively. As explained in Section 4.1, a writhe matrix (T) is a $n_1 \times n_2$ matrix whose element T_{ij} is the writhe between segment i of S_1 and segment j of S_2. The distribution of these T_{ij}s is modelled as a non-singular Bivariate Gaussian function. Parameters of a particular writhe matrix is defined as parameters of the function. In order to discover these parameters, a writhe matrix is fitted to a Bivariate Gaussian function using a least square method.

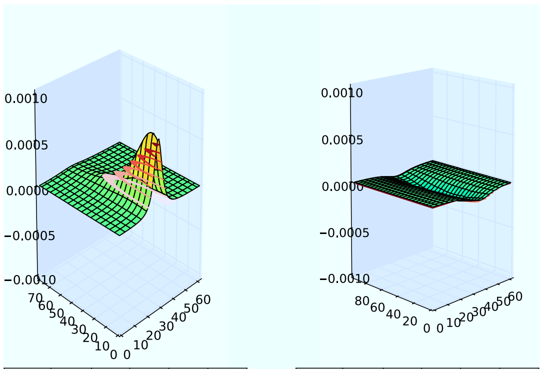
A non-singular Bivariate Gaussian function is described as the following [30]:

$$f(x, y) = A\mathrm{e}^{-(a(x-\mu_x)^2 + 2b(x-\mu_x)(y-{}^{m}u_y) + c(y-\mu_y)^2)},$$

where

$$\begin{aligned} a &= \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2} \\ b &= -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2} \\ c &= \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2}. \end{aligned} \quad (2)$$

© 2013 Information Processing Society of Japan



A	μx	μy	σx	σy	θ
8.04e-4	-10.50	29.17	26.67	3.78	52.94

A	μx	μy	σx	σy	θ
1.56e-4	41.73	35.60	42.06	7.830	0.05

Fig. 4 Result of fitting bivariate Gaussian distribution to writhe matrices shown in Figure 2(a). Resulting parameters are shown under each graph, and the layer in brown, are the Gaussian distribution reconstructed from the resulting parameters.

Parameter	Description
A	amplitude of the peak
μ_x	mean in x direction
μ_y	mean in y direction
σ_x	standard deviation in x direction
σ_y	standard deviation in y direction
θ	rotation angle (clockwise)

Table 3 Description of parameters for Bivariate Gaussian function.

Eq. (2) have six parameters, $\left(A, \mu_x, \mu_y, \sigma_x, \sigma_y, \theta\right)$. The explanation of all parameters are summarized in Table 3.

A least-square method is used to find the most appropriate parameters to fit a particular writhe matrix.A Levenberg-Marquardt algorithm is used to find the optimized parameters. For fitting writhe matrix $T_{n_1 \times n_2}$, the objective function for the optimization is given in Eq. (3) as

$$\underset{(A,\mu_x,\mu_y,\sigma_x,\sigma_y,\theta)}{\text{minimize}} \qquad \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\left(T_{i,j} - f(i,j)\right)^2, \qquad (3)$$

where $f(i, j)$ is a Bivariate Gaussian function defined in Eq. (2). Notice from Eq. (3) that the objective function allows the writhe matrix to be partially fitted to the Gaussian function.

The segmentation is done so that all writhe matrices in every shorter movement are the same type (either peak-wm or span-wm). When considering writhe matrices in regrasping movement, the change in writhe matrix are continuous. In fact, there is no clear line to separate between peak-wm and span-wm, especially when a peak-wm is about to change to span-wm during detaching. To solve this issue, thresholding is used to create a clear separation for two types of writhe matrices.

Two parameters, a standard deviation of writhe matrix along a finger σ_x obtained the fitting and a total writhe w, are considered in this segmenting process because of the way writhe matrices are categorised into two types in the first place. In the case of peak-wm, when the finger are in contact with the object, the value of total writhe is usually higher than that of the span-wm. On the contrary, the value of σ_x of peak-wm is supposed to be lower than that of the span-wm. This is because when a finger is in contact with the object, the high values of writhe along the finger direction seems to be concentrated only in the vicinity of the contact location. The relationship of both types of writhe matrices

Type	σ_x	$\lvert\mathbf{w}\rvert$
Peak-wm	low	high
Span-wm	high	low

Table 4 Relationship between two types of writhe matrices and their parameters.

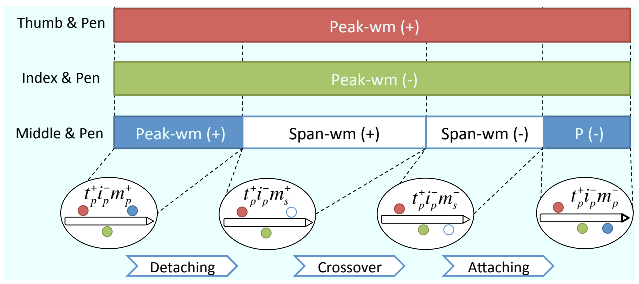


Fig. 5 Example of identifying state of grasping posture from type of writhe matrix. Note that blue edges of task primitives indicates that they belong to the movement of the middle finger.

and theirs parameters is summarized in Table 4.

A combination of the two parameters $\sigma_x / |w|$ is used as a feature for a segmentation. Captured regrasping movement is turned into a sequence of writhe matrices. Hysteresis thresholding is applied on a temporal sequence of features. The higher threshold is related to span-wm, while the lower threshold is related to peak-wm.

The reasons why $\sigma_x / |w|$ is used, are comprehensible from the way writhe matrices are categorised. In the case of peak-wm, when the finger are in contact with the object, the value of total writhe is usually higher than that of the span-wm. On the contrary, the value of σ_x of peak-wm is supposed to be lower than that of the span-wm. This is because when a finger is in contact with the object, the high values of writhe along the finger direction are concentrated only in the vicinity of the contact location.

Once a sign and a type of all writhe matrices of all three fingers are known, state of all grasping postures, explained in Section 4.3, can be easily determined. Finally, task primitives are recognised based on the change of these states. Figure 5 illustrates how task primitives are recognised from the sequence of grasping states.

5.2 Extracting Skill Parameters

After task primitives are recognised, the human regrasping movement can be semantically represented by a sequence of task primitives. In this section, a method to extract skill parameters for each task primitives is described. These skill parameters will be used to reproduce the movement for the corresponding task primitives on a robotic hand.

In Section 5.1, task primitives are recognised from the change of one state to another. However, a state is representing a range of grasping postures. In other word, many consecutive grasping postures are binded to the same states. The issue is that skill parameters of the task primitive are extracted from the initial state and final state, but they are not constant across the same state excepts parameter that indicates type of state itself. Therefore, before skill paramters of each task primitive can be extracted, the range of the task primitive is needed to be specified. Then skill parameters can then be extracted from the first and last grasping postures of the range.

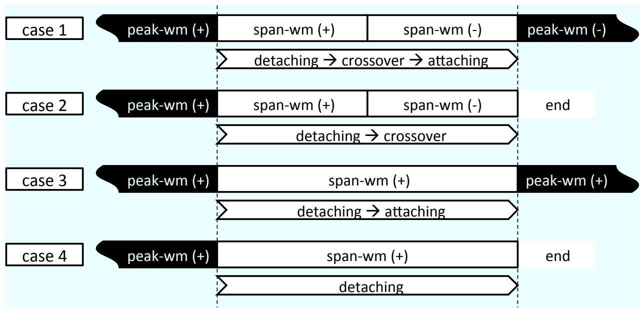
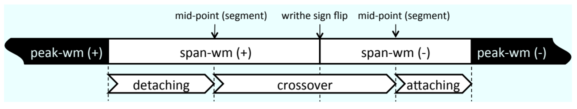
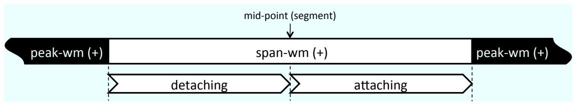


Fig. 6 Four possible cases of task primitives in a span-wm segment.



(a) Extracting skill parameters when sign of span-wm changes.



(b) Extracting skill parameters when sign of span-wm does not change.

Fig. 7 Skill parameters for each task primitives are extracted from the corresponding writhe matrix of the first and last frames of the segment that represents it.

5.2.1 Specifying a Range of Task Primitive

All task primitives always occur and are detected when a writhe matrix of one finger is span-wm. In other word, every time when there is a segment of span-wm occurs, it indicates that there is at least one task primitive within that segment.

For all span-wm segments, there are four possible cases, as shown in Figure 6.

case 1 If the span-wm segment is in between two peak-wm segments and the sign of writhe changes to the opposite one, there are three task primitives occurred in the segment which are *detaching* → *crossover* → *attaching*.

case 2 If the span-wm segment is connected to only one peak-wm segment and the sign of writhe changes to the opposite one, there are two possibilities in this case. When the segment is at the beginning of the hand movement, there is an occurrence of a sequence of *crossover* → *attaching* task primitives. On the other hand, when the segments is at the end of the hand movement, thus connected to peak-wm at the beginning of the segment, there is a sequence of *detaching* → *crossover* task primitives occurred.

case 3 If the span-wm segment is in between two peak-wm segments and the sign of writhe value does not change, there is an occurrence of a sequence of *detaching* → *attaching* task primitives.

case 4 If the span-wm segment is connected to only one peak-wm segment and the sign of writhe value does not change, there is one movement occurred in the segment. When the segment is at the beginning of the hand movement, it is a *attaching* primitive. On the other hand, if the segment is at the end of the hand movement, it is a *detaching* primitive.

When a span-wm segment contains more than one task primitive, the boundary of each movement primitive is needed to be specified. The ways the boundary is chosen are different depending whether there is a *crossover* primitive involved. When there is a *crossover* primitive in between a span-wm segment (in **case 1** and **case 2**), the span-wm segment is first divided into two shorter segments at the frame where the sign of its writhe value flips. Then, the first and last grasping posture of *crossover* are taken from the middle frame of the two smaller segments. On the other hand, when there is no *crossover* primitive (thus no sign change in span-wm segment), the middle frame of span-wm segment is chosen as a segmented frame if required. Notice that when the span-wm segment is segmented, the middle frame is used in this case, as opposed to the ideal case where the frame that the finger is farthest from the object would be chosen.

Once a range of each task primitive are specified, the less of their skill parameters can be obtained by considering its first and last grasping postures.

5.2.2 Extracting Parameters of a Writhe Matrix

For each task primitive, parameters of writhe matrices of its first and last grasping postures (frame) are used as its skill parameters. Parameters for each writhe matrix are different depending on its types, as described in Section 4.5. This section explains how to extract parameters from a given writhe matrix for both peak-wm and span-wm.

For peak-wm T^p of size $n_1 \times n_2$, its parameters are referred to as $\mathcal{P}(T^p) = (w, p_x, p_y)$. Writhe is a sum of all elements in the matrix. The less of the parameters can be extracted by fitting the writhe matrix to the Bivariate Gaussian. The same method explained in Section 5.1 is used. The value of peak location are taken directly from the expectation value of the fitting result.

$$\boldsymbol{p} = (p_x, p_y) = (\mu_x, \mu_y). \tag{4}$$

This is also the incentive behind the name *peak* writhe matrix.

In Table 1, it can be seen that the range of (μ_x, μ_y) is different from the range of (p_x, p_y). Therefore, the final value of $\boldsymbol{p}$ is changed from $[1, n_1] \times [1, n_2]$ to $[-1, 1] \times [-1, 1]$ by linearly scaling as

$$(p_x, p_y) \leftarrow \left(2 \cdot \frac{p_x}{n_1} - 1, 2 \cdot \frac{p_y}{n_2} - 1\right). \tag{5}$$

For span-wm T^s of size $n_1 \times n_2$, its parameters are referred to as $\mathcal{S}(T^s) = (w, \alpha, l_y)$. Writhe is a sum of all elements in the matrix, while the less of the parameters is also derived from the result of the Bivariate Gaussian fitting explained in Section 5.1. They are derived as the followings:

- A line equation l represents a line that passes through μ_x, μ_y and having a slope θ (another parameter from the Gaussian fitting).
- A span-line of span-wm is actually the same line as l. They have the same slope,

  $$\alpha = \theta, \tag{6}$$

  but with a different passing point. The other part l_y is defined as a y-coordinate of the point $({}^{n_1}/_2, l_y)$, whom l passes through. Therefore,

  $$l_y = \left(\frac{n_1}{2} - \mu_x\right) \tan \alpha + \mu_y. \tag{7}$$

This indicates that the location where the mid of the finger tangled with the object is considered as the average location on the object that is tangled by the whole finger.

Similarly with peak-wm, the range of the coordinate system change from $[1, n_1] \times [1, n_2]$ to $[-1, 1] \times [-1, 1]$. This effects in the linear scaling in l_y, and also the value of α as,

$$\alpha = \arctan\left(\frac{n_2}{n_1} \tan\theta\right). \tag{8}$$

6. Movement Mapping

This section starts with an explanation of how to refine skill parameters obtained from the recognition. Then, a framework on how to map each type of task primitives on to the robotic hand is described.

6.1 Refinement of Skill Parameters

This section explains a method to refine skill parameters obtained from human demonstration. The main objective of this refinement is to modify skill parameters, so that they are suitable to be used to map the movement to a robotic hand. The refinement is necessary due to two main reasons. First, the inaccuracy of a data acquisition system dues to the demonstration in virtual environment. This makes captured skill parameters violate some physical constraints. Second, the differences between structures of the human hand and the target robotic hand. This make the skill parameters without any refinement impossible to be used in the target robotic hand.

The section starts by introducing constraints that restrict skill parameters. Then a refinement method for each task primitives based on these constraints are described.

6.1.1 Constraints for Skill Refinement

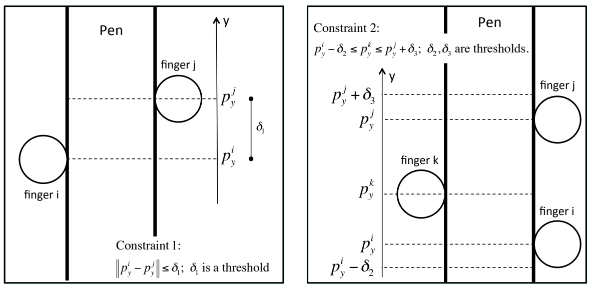
There are three constraints for skill refinement described in this subsection. The first two constraints are defined to force the skill parameters to obey the physical constraints when executing a movement with a robotic hand in the real world. The last constraint is defined to adjust skill parameters to be used in the target robotic hand. These constraints are defined based on the following preconditions:

- At least two fingers out of the three are in contact with the object, while contacts are modeled as a soft finger frictional contact [31].
- an object is a pen-like object.
- Within each task primitive, there is no movement of the object.
- The problem is simplified into two dimensional problem.
- There are only some specific area on the fingers of robotic hand that can be contacted with the object.

These preconditions lead to the following constraints.

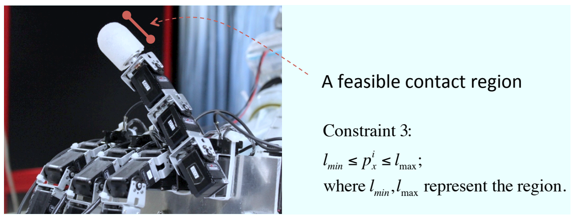
Constraint 1 When only two fingers are in contact with the pen, their contact location on the pen must be on the opposite side of the pen and near to each other, in order to kept the pen in stable state and not moving. Considering $\mathcal{P}_i(w^i, p_x^i, p_y^i)$ and $\mathcal{P}_j(w^j, p_x^j, p_y^j)$ as the skill parameters of two fingers that are in contact with the pen, these constraints can be written as

$$\|p_y^i - p_y^j\| \leq \delta_1, \tag{9}$$



(a) Visualisation of **Constraint 1** (b) Visualisation of **Constraint 2**

Fig. 8 Visualisation of constraints that are related to contact locations.


A feasible contact region
Constraint 3:
$l_{min} \leq p_x^i \leq l_{max}$;
where l_{min}, l_{max} represent the region.
Fig. 9 Limited area of contact on a robotic hand.

where $\delta_1 \geq 0$ is a small threshold value.Figure 8(a) illustrates the constraint in a better visualisable manner.

Constraint 2 When all three fingers are in contact with the pen, two of the contact will be in one side of the pen while another contact will on the other side. A contact location on the pen of the lone contact must be in between the others. Considering $\mathcal{P}_i(w^i, p_x^i, p_y^i)$, $\mathcal{P}_j(w^j, p_x^j, p_y^j)$ and $\mathcal{P}_k(w^k, p_x^k, p_y^k)$ as the skill parameters of all fingers that are in contact with the pen where $\mathcal{P}_k$ is a lone contact and $p_y^i < p_y^j$ is assumed without loss of generality, this constraint can be written as

$$p_y^i - \delta_2 \leq p_y^k \leq p_y^j + \delta_3, \tag{10}$$

where $\delta_2, \delta_3 \geq 0$ are small threshold values. Figure 8(b) shows more comprehensive illustration of this constraint.

Constraint 3 For a particular robotic hand, the area on fingers that can be in contact with the pen can be limited. The related skill parameters must be adjusted accordingly. Considering $\mathcal{P}_i(w^i, p_x^i, p_y^i)$ as skill parameters of the finger that is in contact, and $[l_{min}, l_{max}] \in [-1, 1]$ are the area of the robotic finger that could be in contact, this constraint can be written as

$$l_{min} \leq p_x^i \leq l_{max}. \tag{11}$$

6.1.2 Skill Refinement for Task Primitives

A method for refining skill parameters of each task primitive is based on preconditions and constraints explained in Section 6.1.1. Constraints are applied to the initial and final writhe matrices of the task primitives to initially modify their skill parameters. Then a different refinement method is applied depending on a type of the task primitive.

For every type of task primitive, both initial or final grasping postures has three writhe matrices involved as described in Table 2: one for major finger and two for two minor fingers. Without loss of generality, assuming that

- $\mathcal{F}_0(w^0, \ldots)$ is parameters of writhe matrix (either peak-wm or span-wm) of the major finger m_0,

- $\mathcal{P}_1(w^1, p^1_x, p^1_y)$ is parameters of peak-wm of the first minor finger m_1 where $w^0w^1 > 0$ (same sign of writhe value),
- and $\mathcal{P}_2(w^2, p^2_x, p^2_y)$ is parameters of peak-wm of the second minor finger m_2 where $w^0w^2 < 0$ (opposite sign of writhe value),

constraints modify these parameters as the following:

- if $\mathcal{F}_0$ is parameters of span-wm, **Constraint 1** is applied by modifying

$$p^2_y = \begin{cases} p^2_y & \|p^2_y - p^1_y\| \le \delta_1 \\ p^1_y - \delta_1 & p^2_y < p^1_y - \delta_1 \\ p^1_y + \delta_1 & p^2_y > p^1_y + \delta_1 \end{cases}, \tag{12}$$

where $\delta_1 \le 0$ is a small threshold value.

- if $\mathcal{F}_0$ is parameters of peak-wm and $p^0_y < p^1_y$ is assumed without loss of generality, **Constraint 2** is applied by modifying

$$p^2_y = \begin{cases} p^2_y & p^0_y - \delta_2 \le p^2_y \le p^1_y + \delta_3 \\ p^0_y - \delta_2 & p^2_y < p^0_y - \delta_2 \\ p^1_y + \delta_3 & p^2_y > p^1_y + \delta_3 \end{cases}, \tag{13}$$

where $\delta_2, \delta_3 \ge 0$ are small threshold values.

- Notice that **Constraint 3** is applied to any peak-wm of any finger to modify a contact location on the finger to be feasible for a target robotic hand.

For each type of task primitive, skill parameters are further refined so that when the movement is mapped to the robotic hand, it does not affect on the movement of the pen. Fundamentally, the refinement methods are based on the assumption that if the contact location on the pen of both minor fingers are the same in both initial and final grasping postures and the location of the pen is fixed when the task primitive is interpolated for the robotic hand, a location (position and orientation) of the pen should not change much during the real execution. Therefore, it depends on whether to use the contact location of the initial or the final grasping posture for both minor fingers. These refinement methods are different for each type of task primitive, which are described below.

- For *Detaching*, contact locations on the pen of the two minor fingers of the final grasping posture are used. The contact locations on the pen the initial grasping posture are then changed and refined to these values.
- For *Crossover* and *Attaching*, contact locations on the pen of the two minor fingers of the initial grasping posture are used. The contact locations on the pen the final grasping posture are then changed and refined to these values.

Note that the refinement is applied sequentially by the order of task primitives. For two consecutive task primitives that share common grasping posture and skill parameters, the changes in skill parameters of the prior task primitive will affect the changes of the skill parameters of the latter task primitive.

6.2 Mapping of Task Primitives

The section describes a framework to map task primitives to a robotic hand. The purpose of this mapping framework is to generate trajectories of joints/hand of the robotic hand that is resembled to a demonstrated movement. Skill parameters of each task primitive are used as a fundamental knowledge to ensure that the generated task primitive is similar to the demonstrated one.

Generating (or interpolating) a trajectory of the robotic hand can be considered as solving a series of Inverse Kinematics (IK) problem. In this framework, the tangle relation of hand and object are used as a goal constraints. To be more specific, the trajectory of desired writhe matrices are used to lead the interpolation of the robotic hand. The method to generate the trajectory of desired writhe matrices that connects between two writhe matrices is also proposed in the framework. In fact, the idea of planning a trajectory in topological space was originally used to synthesise a whole body motion of a character [26]. The method is adopted and modified to be used in this framework.

To generate a trajectory of a robotic hand for each task primitive, the following steps are used.

step 1 Mapping skill parameters to the initial and final robot grasping postures. This step maps an abstract information obtained from human demonstration to the robotic hand. It creates the initial and final grasping postures of the robotic hand that have the same skill parameters as the movement primitive.

step 2 Interpolating the robotic hand from its initial grasping posture to its final grasping posture. This step generates a trajectory of the robotic hand to connect between its initial and final grasping postures. The motion is synthesised by planning a trajectory in the topology space.

The less of the section is organized as the followings. First, the method for solving IK problem of the robotic hand in the topology space is described. It is used heavily in both steps for the mapping framework. Then both steps of the mapping framework are explained respectively.

6.2.1 Inverse Kinematics in Topology Space

The method is first used by Ho and Komura [26] to solve IK problem for a character. This section describes the method and how to formulate it to solve an IK problem for a robotic hand.

A generalized coordinate of the robotic hand, $\mathbf{r}$ at the specific moment is described by its joint angles and its location (position and orientation). Writhe matrix T represents tangle relation between the robotic hand and the object. It represents either a writhe matrix between a specific finger and the object, or a concatenation of them. To simplify the representation, let $\mathbf{t}$ be a vector representation of writhe matrix T. For writhe matrix $T_{m\times n}$, $\mathbf{t}$ is a vector of size $m \times n$ where,

$$\mathbf{t} = [T_{(1,1)}, \ldots, T_{(1,n)}, T_{(2,1)}, \ldots, T_{(2,n)}, \ldots, T_{(m_1)}, \ldots, T_{(m,n)}]^{\mathsf{T}} \tag{14}$$

Assume a location of the object is fixed, writhe matrix is a function of a generalized coordinate of the robotic hand alone; this fact can be expressed as $\mathbf{t} = \mathbf{t}(\mathbf{r})$, where $\mathbf{t}$ is a

Given a target tangle relation between the robotic hand and the object as writhe matrix $\mathbf{t}^d$, the IK problem is to find a hand generalized coordinate, $\mathbf{r}$ so that

$$\mathbf{t}^d = \mathbf{t}(\mathbf{r}). \tag{15}$$

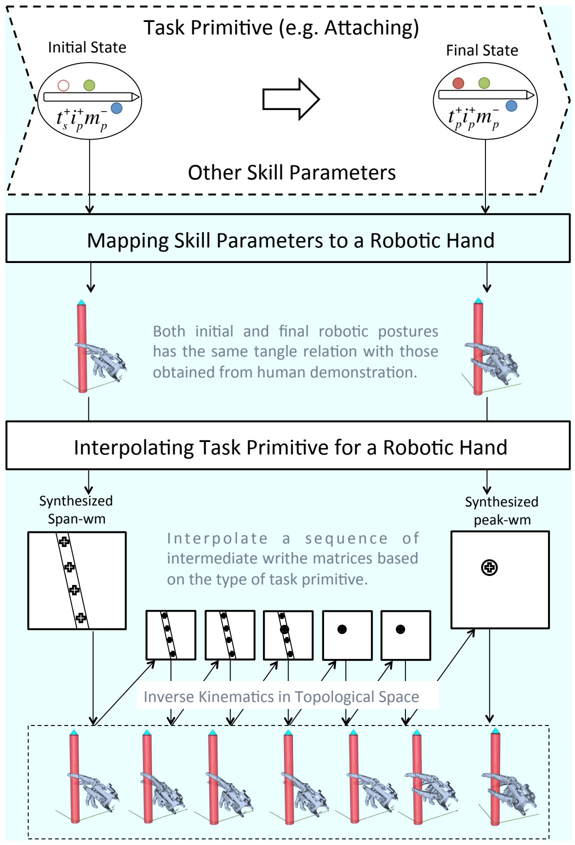

Task Primitive (e.g. Attaching)
Initial State
Final State
$t_s^+ i_p^+ m_p^-$
$t_p^+ i_p^+ m_p^-$
Other Skill Parameters
Mapping Skill Parameters to a Robotic Hand
Both initial and final robotic postures has the same tangle relation with those obtained from human demonstration.
Interpolating Task Primitive for a Robotic Hand
Synthesized Span-wm
Synthesized peak-wm
Interpolate a sequence of intermediate writhe matrices based on the type of task primitive.
Inverse Kinematics in Topological Space

Fig. 10 Framework for mapping task primitives to a robotic hand. A task primitive is passed to a system together with its skill parameters obtained from human demonstration. A corresponding skill parameters is first mapped to a robotic hand to create initial and final grasping postures. Then a trajectory of a robotic hand is created to connect between the two grasping postures by interpolating in a topological space.

An iterative method is used to approximate a solution for Eq. (15). A Jacobian matrix is used to linearly approximate the function of $\mathbf{t}$. The small change in the generalized coordinate of the robotic hand relates to the change in writhe matrix through this Jacobian matrix as,

$$\Delta\mathbf{t} = J(\mathbf{r})\,\Delta\mathbf{r}. \tag{16}$$

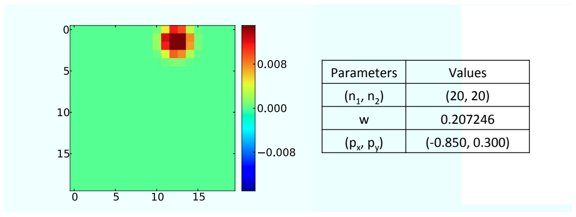
Starting with a generalized coordinate of the robotic hand $\mathbf{r}$ with a writhe matrix $\mathbf{t}$ and a target writhe matrix $\mathbf{t}^d$, the key idea of solving this IK problem is to update a generalized coordinate with $\Delta\mathbf{r}$ so that it effects in a change of writhe matrix $\Delta\mathbf{t}$ which is approximately equal to $\mathbf{t}^d - \mathbf{t}$.

To find the most appropriate value of $\Delta\mathbf{t}$, the damped least squares method [32, 33] is used. The objective function to search for $\Delta\mathbf{t}$ is given as the following:

$$\underset{\Delta\mathbf{r}}{\text{minimize}} \quad \|J(\mathbf{r})\,\Delta\mathbf{r} - \left(\mathbf{t}^d - \mathbf{t}\right)\|^2 + \lambda^2\|\Delta\mathbf{r}\|^2, \tag{17}$$

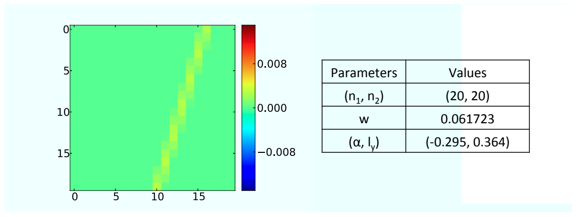
where $\lambda \in \mathbb{R}$ is a non-zero damping constant. Damping constant λ can also be chosen to be different for each finger or each element in the generalized coordinate.

Other constraints can be included in Eq. (17) in order to restrain the generated regrasping movement. In this system, two additional constraints are added; a collision avoidance and a kinematic limitation of the robot hand.



Parameters	Values
(n_1, n_2)	(20, 20)
w	0.207246
(p_x, p_y)	(-0.850, 0.300)

(a) Synthesised peak-wm



Parameters	Values
(n_1, n_2)	(20, 20)
w	0.061723
(α, l_y)	(-0.295, 0.364)

(b) Synthesised span-wm

Fig. 11 Examples of synthesised writhe matrices of size $n_1 \times n_2$.

Collision between a robotic hand and manipulated object can be avoided by limiting the maximum writhe between segments of concerned strands. When two line segments approach each other, the absolute value of their writhe increases and becomes 0.5 at the moment of crossing. This can be described as :

$$\|J(\mathbf{r})\,\Delta\mathbf{r} + \mathbf{t}\| \leq \sigma, \tag{18}$$

where σ is a threshold vector.

Kinematic limitation of finger/hand are restricted by assigning a feasible search space for $\Delta\mathbf{r}$. Assume that $\mathbf{r}_{min,k}$ and $\mathbf{r}_{max,k}$ are a minimum and maximum possible value for joints and movement of robot hand, the constraint can be written as :

$$\mathbf{r}_{min} - \mathbf{r} \leq \Delta\mathbf{r} \leq \mathbf{r}_{max} - \mathbf{r}. \tag{19}$$

Skill parameters obtained from human demonstration is passed to the robot system in the form of parameters of writhe matrices, $\mathcal{P}$ or $\mathcal{S}$. In order to use these informations for robot mapping, it is necessary that these parameters are converted back to the writhe matrix so that the method to solve IK problem in topology space can be used. The writhe matrix will be referred to as a *synthesised writhe matrix*. Considering peak-wm with parameter set $\mathcal{P} = (w, p_x, p_y)$, synthesised writhe matrix is initialised as a singular bivariate Gaussian distribution with the mean at (p_x, p_y). Standard deviations for both axes are set empirically, depending on the target robot hand and the object. These values are used to control the distance between the strands of finger and object of the resulting grasp posture. On the other hand, for span-wm with parameter set $\mathcal{S} = (w, \alpha, l_y)$, synthesised writhe matrix is initialised with all elements set to zero, except those that pass through by line (α, l_y) are set to 1. In both cases, the writhe matrix is normalised to the corresponding writhe value w. Figures 11(a) and 11(b) illustrates an example of synthesised peak-wm and synthesised span-wm respectively.

6.2.2 Mapping Skill Parameters to Robot Postures

Before generating a trajectory of a robotic hand for each task primitive, initial and final grasping postures of the robotic hand are created to have the same skill parameters as of the task primitive. This can be done by solving the IK problem, individually for

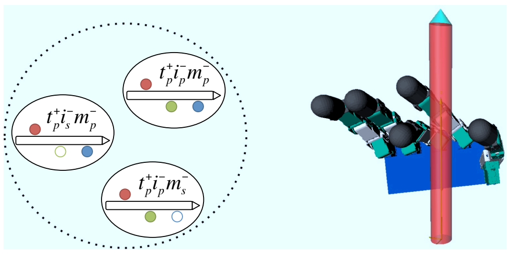


Fig. 12 An initial robotic grasping posture for all states with $t^+i^-m^-$. It is used to initialise a robotic configuration before it is mapped to skill parameters obtained from human demonstration.

initial and final postures, given parameters of their writhe matrices which are part of the skill parameters. The remaining question is how to initialise the robotic grasping posture so that it can be used as an initial generalized coordinated of the IK problem.

A configuration of the robotic hand for IK problem is initialised with state information i.e. s_i or s_f in Table 2. For each group of states with the same sign of writhe matrices for all three fingers, a robotic grasping posture is prepared and given as an input to the system. The grasping posture does not need to be in a particular type of writhe matrix or to have a specific skill parameters of the writhe matrices. It is only needed to have the same correspondence of the sign of all writhe matrices with states it represented. This will provide an initial configuration of the robotic hand and also a relative location of the hand to the pen, which can then be adjusted accordingly to the location of the pen. Figure 12 shows examples of an initial grasping posture for states with $t^+i^-m^-$.

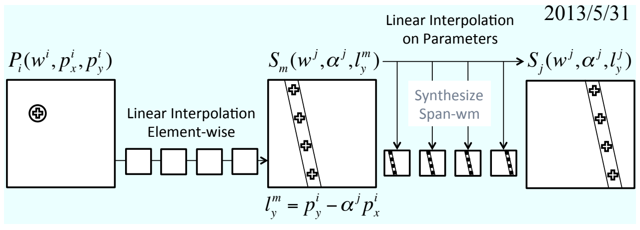
Notice that in this step, the trajectory to reach the desired writhe matrices is not important. The hand configuration and location are updated iteratively, using method explained in Section 6.2.1, until writhe matrices between the robotic hand and then pen are similar to the desired writhe matrices.

6.2.3 Interpolate Hand Motion between Initial and Final States

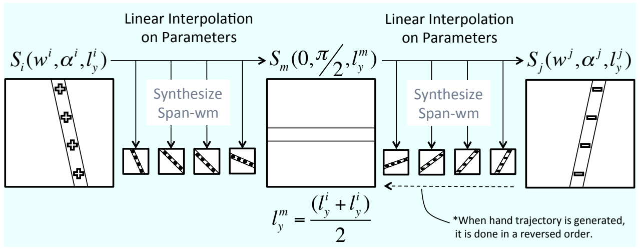
Once initial and final grasping postures of a robotic hand for the task primitive is created, the next step is to generate a trajectory of the robotic hand that connects them together. An IK solver explained in Section 6.2.1 is used to generate the trajectory of the robotic hand. The method tries to move the robotic hand so that its writhe matrices are similar to the desired writhe matrices. Therefore, to interpolate a trajectory for the robotic hand, a sequence of intermediate writhe matrices to lead the trajectory interpolation is required. In other words, this sequence of intermediate writhe matrices dictates how the movement of robot hand turns out to be for each task primitives. In this section, a method to create a sequence of intermediate writhe matrices for each task primitives is explained.

A sequence of intermediate writhe matrices of major finger for each task primitive is created as follows:

- *Detaching* : Consider $\mathcal{P}_i(w^i, p_x^i, p_y^i)$ and $\mathcal{S}_j(w^j, \alpha^j, l_y^j)$ as skill parameters for initial peak-wm and final span-wm writhe matrix. In this case, a mid span-wm writhe matrix is necessary. It is assigned with skill parameters $\mathcal{S}_m(w^m, \alpha^m, l_y^m)$, where


Linear Interpolation on Parameters
$P_i(w^i,p_x^i,p_y^i)$
$S_m(w^j,\alpha^j,l_y^m)$
$S_j(w^j,\alpha^j,l_y^j)$
Synthesize Span-wm
Linear Interpolation Element-wise
$l_y^m = p_y^i - \alpha^j p_x^i$

(a) Task primitive : *Detaching* (equivalent to a reversal of *Attaching*).


Linear Interpolation on Parameters
Linear Interpolation on Parameters
$S_i(w^i,\alpha^i,l_y^i)$
$S_m(0,\pi/2,l_y^m)$
$S_j(w^j,\alpha^j,l_y^j)$
Synthesize Span-wm
Synthesize Span-wm
$l_y^m = \frac{(l_y^i + l_y^j)}{2}$
*When hand trajectory is generated, it is done in a reversed order.

(b) Task primitive : *Crossover*.

Fig. 13 Methods to generate a sequence of intermediate writhe matrices for a major finger for each task primitives.

$$\begin{aligned} w^m &= w^j \\ \alpha^m &= \alpha^j \\ l_y^m &= p_y^i - \alpha^j p_x^i. \end{aligned} \tag{20}$$

A pair (α^m, l_y^m) represents a span-line that pass through point (p_x^i, p_y^i), but has the same direction as a span-line (α^j, l_y^j). Then a process of creating a sequence of intermediate writhe matrices for this task primitives is divided into two steps. First, intermediate writhe matrices between $\mathcal{P}_i$ and $\mathcal{S}_m$ are created. This can be done by synthesising both writhe matrices, T^i, T^m,from their skill parameters, and then element-wise linearly interpolating between them to create all intermediate writhe matrices. Assume that $T^r, r \in [1, n]$ is one of the n intermediate writhe matrices that would be created between T^i and T^m, T^r of size $n_1 \times n_2$ is calculated as

$$T_{p,q}^r = r\frac{T_{p,q}^m - T_{p,q}^i}{n+1} + T_{p,q}^i \tag{21}$$

$\forall(p,q) \in [1,n_1] \times [1,n_2]$. Second, intermediate writhe matrices between $\mathcal{S}_m$ and $\mathcal{S}_j$ by linearly interpolating between their skill parameters to obtain skill parameters for all intermediate span-wm, $\mathcal{S}_s$. Then each intermediate writhe matrix is synthesised from this parameter $\mathcal{S}_s$.

- *Attaching* : It uses the same algorithms as *Detaching*, but in the reversed direction.
- *Crossover* : Consider $\mathcal{S}_i(w^i, \alpha^i, l_y^i)$ and $\mathcal{S}_j(w^j, \alpha^j, l_y^j)$ as skill parameters for initial span-wm and final span-wm writhe matrix. Mid span-wm is also necessary in this case. It is assigned with skill parameters $\mathcal{S}_m(w^m, \alpha^m, l_y^m)$, where

$$\begin{aligned} w^m &\approx 0 \\ \alpha^m &= \pi/2 \\ l_y^m &= (l_y^i + l_y^j)/2. \end{aligned} \tag{22}$$

This $\mathcal{S}_m$ represents a zero-wm, a special kind of span-wm, where α^m is a slop of span-line that results in the horizontal line in the writhe matrix. In the similar manner as *detaching*, the process of creating a sequence of intermediate writhe matrix is divided into two steps: creating intermediate

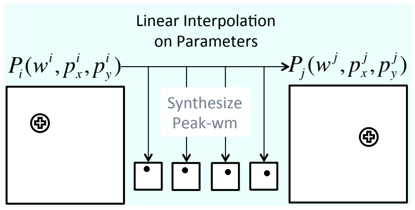


Fig. 14 Methods to generate a sequence of intermediate writhe matrices for minor fingers for all task primitives.

writhe matrices between S_i and S_m and creating intermediate writhe matrices between S_m and S_j. Both steps can be done by linearly interpolating between skill parameters of initial and final writhe matrices to obtain skill parameters for all intermediate span-wm writhe matrices. Then, each intermediate writhe matrix is synthesised from its skill parameters.

Notice that the process of generate a trajectory of the robotic hand for *crossover* is also divided into two steps. The first steps is to generate the trajectory to follow intermediate writhe matrices from S_i and S_m, and the second steps is to generate the trajectory to follow intermediate writhe matrices from S_j and S_m. Then the first trajectory is connected with the reversal of the second trajectory to create a trajectory for *crossover* primitive. This is because the IK solver tends to fail to find a global optimum solution when the total writhe value of the writhe matrix is small.

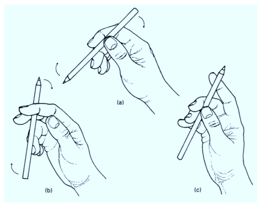
Although there is not much change in skill parameters of minor fingers in each task primitive after the refinement, it is still necessary to provide a method to create a sequence of intermediate writhe matrices for them. Changes in minor fingers are from peak-wm to peak-wm within the same sign of writhe. Consider $\mathcal{P}_i(w^i, p_x^i, p_y^i)$ and $\mathcal{P}_j(w^j, p_x^i, p_y^i)$ as skill parameters for initial peak-wm and final peak-wm writhe matrix. A sequence of intermediate writhe matrices can be created by first linearly interpolating between skill parameters of initial and final peak-wm to obtain skill parameters for all intermediate peak-wm. Then each intermediate writhe matrix is synthesised from its skill parameters.

Figure 13 illustrates a visualization of how the intermediate writhe matrices are interpolated for the major finger and Figure 14 for the minor fingers of all task primitive. Note that linearly interpolation of skill parameters between two span-wm (with the same sign of writhe), that has been used in the process of creating intermediate writhe matrices, results in rotation of a span-line of the span-wm.

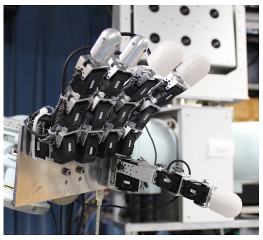
7. Experiments

Interdigital Step task is employed to validate the proposed task model. This movement is selected because it is the most complex movement in the human hand movement classification [34]. Three fingers (thumb, index, middle) are used throughout the movement.

A custom-made robot hand is used for robot mapping. It consists of 20 Futaba RS303MR servo motors connected together. The hand has five fingers and each finger has four joints, thus 20 degrees of freedom. Its thumb has a different structure from the other fingers. The hand is connected to Mitsubishi PA-10 robot arm to be maneuvered around.



(a) Interdigital step [34]



(b) A robot hand

Fig. 15 An experimental data and a target robot hand

7.1 Recognition

Human Interdigital Step movement is recognised against the task model. The movement is first segmented into many smaller segments depending on their type of writhe matrices. The recognition method combines this knowledge and the sign of writhe matrices to determine task primitives that occurs in the movement. Skill parameters for each task primitives are extracted. The sequence of task primitives, together with their skill parameters, are used to represent the observed regrasping movement.

Figure 16 shows a recognition result of Interdigital Step. Figure 16(a) shows a graph of $\sigma_x / |w|$ of the sequence of writhe matrices between the three fingers and the pen and the segmentation result based on two types of writhe matrices of Interdigital Step. Hysteresis thresholding is used to segment each coloured line independently to identify when there is a change in type of writhe matrix, either from peak-wm to span-wm or span-wm to peak-wm. A high threshold of 700 and a lower threshold of 200 is used. At the segmented frames, it indicates a moment when the finger starts to have or to lose a contact with the pen. The segmented frames for the red line are frame 151st and 249th as a changes of contact of the thumb, and for the green line are frame 22nd and 87th as a changes of contact of the index finger. Total writhe value between each finger and the pen indicates the sign of all corresponding writhe matrices. At the particular moment, when a sign and a type of all three writhe matrices are known, a state of the grasping posture can be identified (e.g. $t_p^+ i_s^- m_p^-$ from frame 22nd to 58th). Using this information, task primitives can be easily recognised. In the movement, there are six task primitives. Three of which have an index finger (shown in green), while the others have a thumb (shown in red), as their major finger. Range of each task primitive is calculated using method explain in Section 5.2.1. They indicate frames which the skill parameters of the task primitives should be extracted. It is noticeable that there is no task primitive connecting frame 87th and 151st. All grasping postures in the segment are considered to have the same state, i.e. $t_p^+ i_p^+ m_p^-$. Figure 17 illustrates corresponding grasping postures of initial and final frames of all task primitives.

7.2 Refinement of Skill Parameters

Figure 18 illustrate refinement results for skill parameters obtained the recognition process. Both original and refined skill parameters are shown for comparison. The skill parameter shown is a p_y of peak-wm, which represents a contact location on the pen of the corresponding finger. In each small image, contact

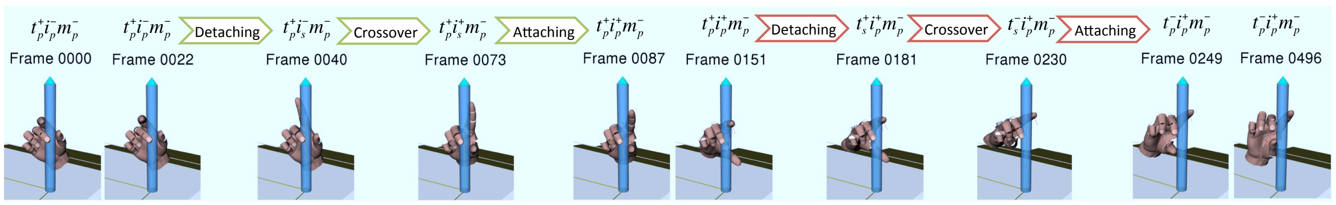
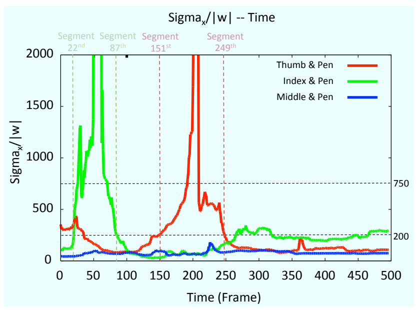
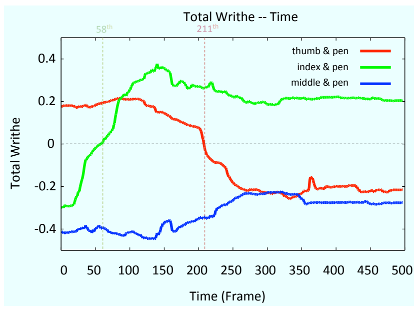


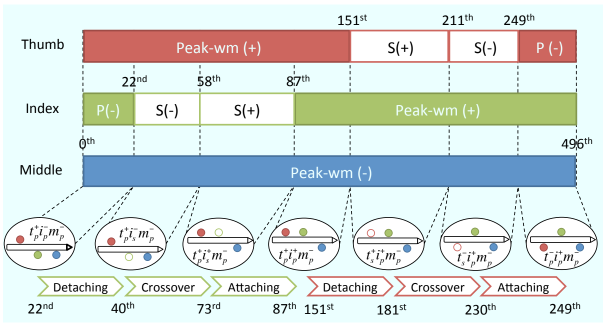
Fig. 17 Grasping postures of the beginning and the end of all task primitives are shown, together with the information on their states (Interdigital Step).



(a) Graph shows $\sigma_x/\,|w|$ of the sequence of writhe matrices of between three fingers and the pen: red = thumb and pen, green = index and pen, blue = middle and pen.The movement is segmented at frame 151st and 249th for the red line, and is segmented at frame 22nd and 87th for the green line.



(b) Graph of total writhe. Three coloured line represent a writhe between pairs of strands: red= thumb and pen, green= index and pen, and blue= middle and pen. The red line crosses a zero value at frame 211th and the green line crosses a zero value at frame 58th.



(c) The regrasping movement is recognised into six task primitives. The colour of the arrows indicate the major finger of the task primitives (red for thumb and green for index finger). Frame numbers of the range of task primitives are also illustrated.

Fig. 16 Recognition results of Interdigital Step.

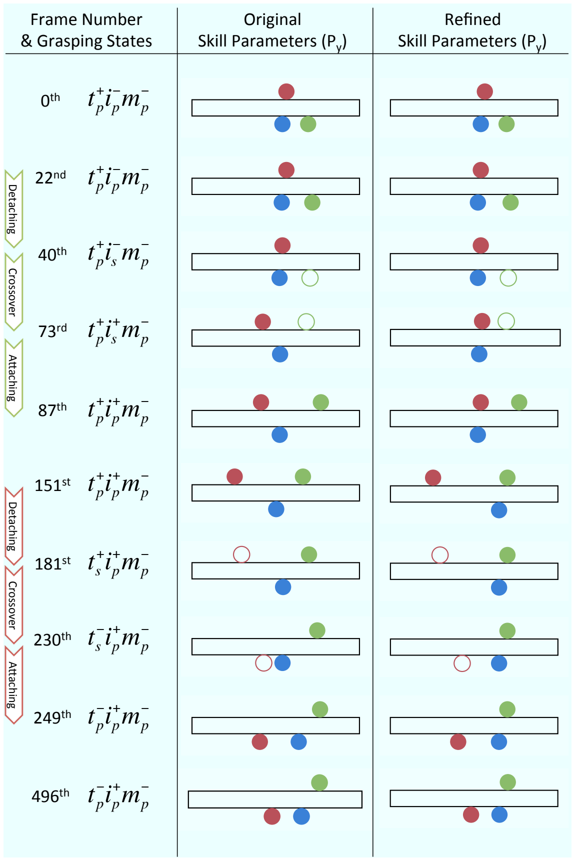

Frame Number & Grasping States
Original Skill Parameters (Py)
Refined Skill Parameters (Py)
Detaching
Crossover
Attaching
Detaching
Crossover
Attaching

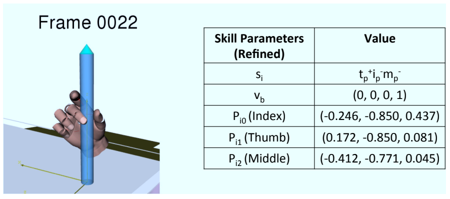
Fig. 18 Refinement of the skill parameters obtained human demonstration of Interdigital Step. The left column shows original skill parameters, where the right column shows the refined version. For a peak-wm (a filled circle), a contact location on the pen p_y is shown, where for a span-wm (an unfilled circle) an average tangled location on the pen l_y is shown instead (red=thumb, green=index finger, blue=middle finger).

locations of all three fingers are shown, except when the corresponding writhe matrix is span-wm, a l_y is displayed instead.

It is noticeable that within a task primitive, contact locations of both minor fingers are refined to be unchanged from the initial state to the final state. As a result, for all consecutive task primitives, the contact locations of their minor fingers are unchanged throughout, e.g. contact location of thumb (red) and middle finger (blue) are unchanged from frame 22nd to 87th in the refined version of Figure 18.

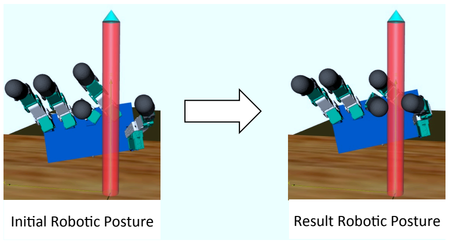
7.3 Mapping to a Robotic Hand

To map a human regrasping movement onto the robotic hand, trajectories of all task primitives are generated and connected together. In order to generate a trajectory for each task primitive, initial and final grasping postures of the robotic hand are first created from the corresponding skill parameters. Then a trajectory that connected between these two grasping postures is generated.



Skill Parameters (Refined)	Value
s_i	$t_p^+ i_p^- m_p^-$
v_b	(0, 0, 0, 1)
P_{i0} (Index)	(-0.246, -0.850, 0.437)
P_{i1} (Thumb)	(0.172, -0.850, 0.081)
P_{i2} (Middle)	(-0.412, -0.771, 0.045)

(a) a human grasping posture (frame 22nd) and its skill parameters.



(b) Initial robotic posture and the result robotic posture of the mapping.

Fig. 19 Skill parameters obtained from human is mapped to a robotic hand (frame 22nd of Interdigital Step).

7.3.1 Mapping Skill Parameters to Robot Postures

Figure 19 illustrates a robotic posture that is mapped to skill parameters that obtained from human demonstration (frame 22nd of Interdigital Step). Three writhe matrices, all peak-wms in this case, are synthesised from their corresponding skill parameters using method explained in Section 6.2.1. Then, they are used as a target tangle relation for solving an IK problem. The robotic posture shown in the left of Figure 19(b) is used as a initialising hand configuration. The IK solver then iteratively searches for a solution, until the criteria is met which yields a resulting posture on the right of the figure. Note that the criteria used for a stop condition is $||\Delta\mathbf{r}||$, where $\Delta\mathbf{r}$ is a small change in the generalized coordinate of the robotic hand.

7.3.2 Mapping of Regrasping Movement

Figure 20 show results of a robot imitating human demonstration of Interdigital Step. Trajectories of all task primitives are generated and connected together. The connected trajectory is passed onto the robot to be executed. In current implementation, the robot uses no force or visual feedback during an execution of the trajectory.

8. Conclusion

This paper describes a novel approach to teach a regrasping movement to a robot. The approach is based on a LFO paradigm, which allows the robot to learn how to perform a certain task by observing and imitating human. Robot plans its own regrasping movement by observing human demonstrating a movement, segmenting and recognising the movement based on a pre-defined task model. This allows an observed regrasping movement to be represented by a sequence of task primitives. The robot then imitates the movement by sequentially executing these task primitives using skill parameters obtained in the planning process.

The task model for imitation is based on the changes of topological relation, referred to as tangle topology, between a hand and an object in a regrasping movement. Hands and a manipulated object are substituted with a set of zero-width strands. Tangle relation examines a relation between these strands. A task model is consisted of two components: task primitives and skill parameters. In the RPO system, task primitives are defined based on changes in types of writhe matrices between fingers and the object. A set of skill parameters for each type of task primitive are a necessary information needed to transfer the task primitive from a human hand to a robotic hand. Both task primitives and skill parameters are recognised and extracted from human demonstration, which turns a captured regrasping movement into a strategy or a plan for a robot to execute it.

A sequence of task primitives together with their skill parameters are used by a robot to imitate the captured regrasping movement. Skill parameters of all task primitives are refined to comply with physical constraints of a target robotic hand and environment before the movement primitives are sequentially executed. A mapping framework to map each task primitive to a robotic hand is described. It is based on the interpolation of a robotic hand in a topological space and composed of two steps: 1.) mapping skill parameters to create an initial and a final postures of a robotic hand and, 2.) creating a trajectory of the hand to connect between the two postures.

Once trajectories for each task primitives are created, they are connected together to create a regrasping movement for a robotic hand. A robot imitates the movement by simply following the pre-constructed trajectory. A robot system used to execute the movement comprised of a 20 DOFs custom-made robotic hand, attached to the Mitsubishi PA-10 robotic arm to maneuver around. The hand is build to resemble the DOF structure of the human hand.

Acknowledgments This work was supported by MEXT/JSPS KAKENHI Grant Number 24240034.

References

[1] Jones, L. and Lederman, S.: *Human Hand Function*, Oxford University Press, USA (2006).
[2] Cutkosky, M.: On grasp choice, grasp models, and the design of hands for manufacturing tasks, *IEEE Trans. Robot. Autom.*, Vol. 5, No. 3, pp. 269–279 (online), DOI: 10.1109/70.34763 (1989).
[3] Ikeuchi, K. and Suehiro, T.: Toward an assembly plan from observation. I. Task recognition with polyhedral objects, *IEEE Trans. Robot. Autom.*, Vol. 10, No. 3, pp. 368 –385 (online), DOI: 10.1109/70.294211 (1994).
[4] Kuniyoshi, Y., Inaba, M. and Inoue, H.: Learning by watching: extracting reusable task knowledge from visual observation of human performance, *IEEE Trans. Robot. Autom.*, Vol. 10, No. 6, pp. 799 – 822 (online), DOI: 10.1109/70.338535 (1994).
[5] Bann, S., Khan, M., Hernandez, J., Munz, Y., Moorthy, K., Datta, V., Rockall, T. and Darzi, A.: Robotics in surgery, *Journal of the American College of Surgeons*, Vol. 196, No. 5, pp. 784 – 795 (online), DOI: 10.1016/S1072-7515(02)01750-7 (2003).
[6] Ambrose, R., Aldridge, H., Askew, R., Burridge, R., Bluethmann, W., Diftler, M., Lovchik, C., Magruder, D. and Rehnmark, F.: Robonaut: NASA's space humanoid, Vol. 15, No. 4, pp. 57 –63 (online), DOI: 10.1109/5254.867913 (2000).
[7] Taylor, M.: Robots for nuclear power plants, *IAEA Bulletin*, Vol. 27, No. 3 (1985).
[8] Trinkle, J. and Hunter, J.: A framework for planning dexterous manipulation, *Proc. IEEE Int'l Conf. on Robot. Autom., ICRA*, Vol. 2, pp. 1245 –1251 (online), DOI: 10.1109/ROBOT.1991.131782 (1991).
[9] Cherif, M. and Guptam, K.: Planning quasi-static motions for reconfiguring objects with a multi-fingered robotic hand, *Proc. IEEE Int'l Conf. on Robot. Autom., ICRA*, Vol. 2, pp. 986 –991 (online), DOI: 10.1109/ROBOT.1997.614263 (1997).
[10] Sudsang, A. and Phoka, T.: Regrasp planning for a 4-fingered hand manipulating a polygon, *Proc. IEEE Int'l Conf. on Robot. Autom., ICRA*, Vol. 2, pp. 2671 – 2676 (online), DOI:

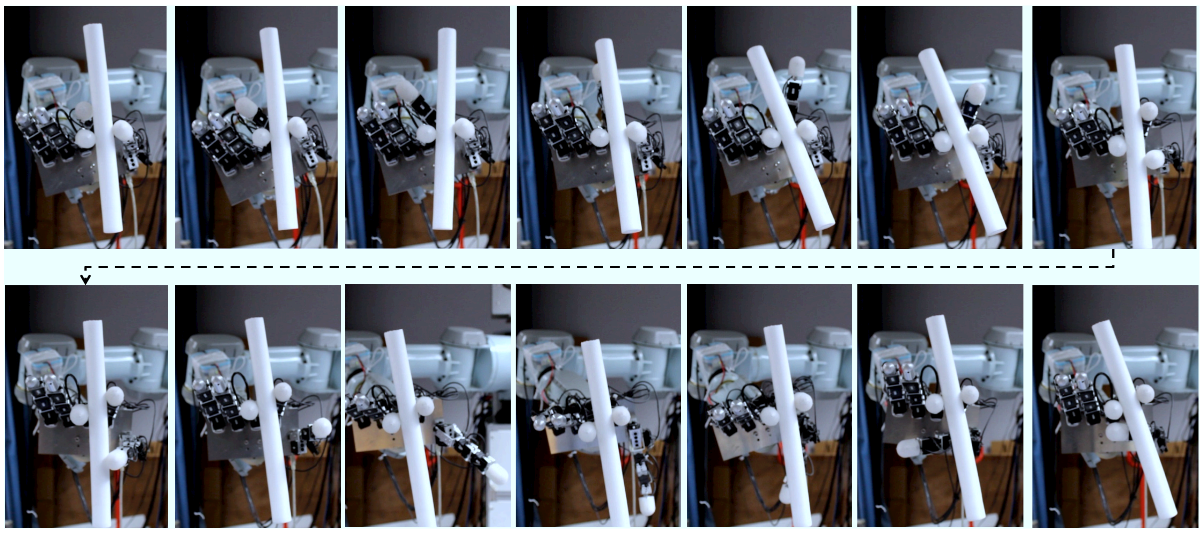


Fig. 20 A result of the robotic hand imitates a human regrasping movement (Interdigital Step).

10.1109/ROBOT.2003.1241996 (2003).

[11] Saut, J.-P., Sahbani, A., El-Khoury, S. and Perdereau, V.: Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces, *IEEE/RSJ Int'l Conf. on Intell. Robot. Syst., IROS*, pp. 2907 –2912 (online), DOI: 10.1109/IROS.2007.4399090 (2007).

[12] Iberall, T.: Human prehension and dexterous robot hands, *Int. J. Rob. Res.*, Vol. 16, pp. 285–299 (1997).

[13] Kang, S. B. and Ikeuchi, K.: Toward automatic robot instruction from perception-mapping human grasps to manipulator grasps, *IEEE Trans. Robot. Autom.*, Vol. 13, No. 1, pp. 81 –95 (online), DOI: 10.1109/70.554349 (1997).

[14] Do, M., Romero, J., Kjellstrom, H., Azad, P., Asfour, T., Kragic, D. and Dillmann, R.: Grasp recognition and mapping on humanoid robots, *IEEE-RAS Int't Conf. on Human. Robot., Humanoids*, pp. 465 –471 (online), DOI: 10.1109/ICHR.2009.5379538 (2009).

[15] Steffen, J., Haschke, R. and Ritter, H.: Towards dextrous manipulation using manipulation manifolds, *IEEE/RSJ Int'l Conf. on Intell. Robot. Syst., IROS*, pp. 2738 –2743 (online), DOI: 10.1109/IROS.2008.4650720 (2008).

[16] Steffen, J., Klanke, S., Vijayakumar, S. and Ritter, H.: Realising Dextrous Manipulation with Structured Manifolds using Unsupervised Kernel Regression with Structural Hints, *ICRA 2009 Workshop: Approaches to Sensorimotor Learning on Humanoid Robots* (2009).

[17] Steffen, J., Oztop, E. and Ritter, H.: Structured unsupervised kernel regression for closed-loop motion control, *IEEE/RSJ Int'l Conf. on Intell. Robot. Syst., IROS*, pp. 75 –80 (online), DOI: 10.1109/IROS.2010.5651216 (2010).

[18] Lam, P. C., Liu, Y.-H., Li, D. and Leung, M.: Generating dextrous manipulation for multifingered robot hands by combining motion planning and teaching, Vol. 3, pp. 1419–1424 vol.3 (online), DOI: 10.1109/CCECE.1999.804914 (1999).

[19] Kondo, M., Ueda, J., Matsumoto, Y. and Ogasawara, T.: Recognition of In-Hand Manipulation along with Rolling Contact using Orbital Motion of Contact Points on Object Surface, pp. 167 –172 (online), DOI: 10.1109/MFI.2006.265662 (2006).

[20] Kondo, M., Ueda, J. and Ogasawara, T.: Recognition of in-hand manipulation using contact state transition for multifingered robot hand control, *Robot. Auton. Syst.*, Vol. 56, No. 1, pp. 66–81 (online), DOI: http://dx.doi.org/10.1016/j.robot.2007.09.018 (2008).

[21] Martins, R., Faria, D. R. and Dias, J.: Symbolic level generalization of in-hand manipulation tasks from human demonstrations using tactile data information, *IROS 2010 Workshop: Grasp Planning and Task Learning by Imitation* (2010).

[22] Faria, D. R., Martins, R., Lobo, J. and Dias, J.: Extracting data from human manipulation of objects towards improving autonomous robotic grasping, *Robotics and Autononous Systems*, Vol. 60, No. 3, pp. 396–410 (online), DOI: 10.1016/j.robot.2011.07.020 (2012).

[23] Miller, A. and Allen, P.: Graspit! A versatile simulator for robotic grasping, *IEEE Robot. Autom. Mag.*, Vol. 11, No. 4, pp. 110 – 122 (online), DOI: 10.1109/MRA.2004.1371616 (2004).

[24] Chang, L. Y., Pollard, N., Mitchell, T. and Xing, E. P.: Feature Selection for Grasp Recognition from Optical Markers, *IEEE/RSJ Int'l Conf. on Intell. Robot. Syst., IROS*, pp. 2944 – 2950 (2007).

[25] Oikonomidis, I., Kyriazis, N. and Argyros, A.: Efficient model-based 3D tracking of hand articulations using Kinect, *British Machine Vision Conference (BMVA)* (2011).

[26] Ho, E. S. L. and Komura, T.: Character Motion Synthesis by Topology Coordinates, *Computer Graphics Forum (Proc. Eurographics 2009)* (Dutr'e, P. and Stamminger, M., eds.), Vol. 28, No. 2, Munich, Germany (2009).

[27] Klenin, K. and Langowski, J.: Computation of writhe in modeling of supercoiled DNA, *Biopolymers*, Vol. 54, pp. 307–317 (2000).

[28] Ho, E. S. and Komura, T.: Indexing and Retrieving Motions of Characters in Close Contact, *IEEE Trans. Vis. Comput. Graphics*, Vol. 15, pp. 481–492 (online), DOI: http://doi.ieeecomputersociety.org/10.1109/TVCG.2008.199 (2009).

[29] Vinayavekhin, P., Kudoh, S. and Ikeuchi, K.: Towards an automatic robot regrasping movement based on human demonstration using tangle topology, *Proc. IEEE Int'l Conf. on Robot. Autom., ICRA*, pp. 3332 –3339 (2011).

[30] Hagen, N. and Dereniak, E. L.: Gaussian profile estimation in two dimensions, *Appl. Opt.*, Vol. 47, No. 36, pp. 6842–6851 (online), DOI: 10.1364/AO.47.006842 (2008).

[31] Kao, I., Lynch, K. and Burdick, J. W.: Contact Modeling and Manipulation, *Springer Handbook of Robotics* (Siciliano, B. and Khatib, O., eds.), Springer, pp. 647–669 (2008).

[32] Wampler, C.: Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods, *IEEE Trans. Syst., Man, Cybern.*, Vol. 16, No. 1, pp. 93 –101 (online), DOI: 10.1109/TSMC.1986.289285 (1986).

[33] Nakamura, Y. and Hanafusa, H.: Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control, *Journal of Dynamic Systems, Measurement, and Control*, Vol. 108, No. 3, pp. 163–171 (online), DOI: 10.1115/1.3143764 (1986).

[34] Elliott, J. and Connolly, K.: A classification of manipulative hand movements, *Developmental medicine and child neurology*, Vol. 26, No. 3, pp. 283–296 (1984).