Chapter from the book *Motion Planning*
Downloaded from: http://www.intechopen.com/books/motion_planning

Interested in publishing with InTechOpen?
Contact us at book.department@intechopen.com
Planning for Unraveling Deformable Linear Objects Based on Their Silhouette

Hidefumi Wakamatsu¹, Eiji Arai¹ and Shinichi Hirai²

¹Osaka University, ²Ritsumeikan University
Japan

1. Introduction

Deformable linear objects such as tubes, cords, cables, wires, and threads are used widely for fixing, fastening, wrapping, packing, suturing, and ligating of objects including themselves. In such manipulative tasks, knotting of linear objects is required. At the same time, their raveling must be avoided. If unexpected ravel occurs, it takes much time to unravel. For example, raveling of earphone/headphone cord of a portable audio player would puzzle you sometimes. So, efficient unraveling is important as well as avoidance of such raveling.

Knotting manipulation by robots has been studied. Inoue et al. reported tying a knot in a rope with a manipulator utilizing visual feedback (Inoue & Inaba, 1984). Hopcroft et al. devised an abstract language to express various knotting manipulations and performed knot-tying tasks with a manipulator (Hopcroft et al., 1991). Matsuno et al. realized a task consisting of tying a cylinder with a rope using a dual manipulator system (Matsuno et al., 2001). Takamatsu et al. have been developing a system for knot planning from observation of human demonstrations (Takamatsu et al., 2006). Saha and Isto proposed a motion planner for manipulating ropes and realized tying several knots using two cooperating robotic arms (Saha & Isto, 2006). Yamakawa et al. proposed a new strategy for making knots with one high-speed multifingered robot hand having tactile sensors (Yamakawa et al., 2007). Unknotting manipulation, i.e., the inverse of knotting manipulation, has been also studied. We have realized automatic planning and execution of knotting/unknotting manipulation (Wakamatsu et al., 2006). Ladd and Kavraki developed an untangling planner for mathematical knots represented as closed piecewise linear curves (Ladd & Kavraki, 2004).

Unraveling is equivalent to unknotting. However, the state of a raveled object can become more complex than that of a knotted object. Moreover, it is difficult to recognize the state of a raveled object completely because it may twine itself. Therefore, recognition of the object state and manipulation planning are both important for unraveling. In this paper, we propose a planning method for unraveling a linear object when 3D information about the object state is unknown. First, an unknotting process of a linear object, which is equivalent to its unraveling process, is represented as a sequence of crossing state transitions. The object state is categorized according to three properties with respect to self-crossings of the object. State transitions are defined by introducing four basic operations. Then, possible
unknotting processes can be generated if the current crossing state is completely identified. Second, the crossing sequence of a linear object, which is related to its silhouette, is considered. The crossing sequence can be categorized into two types: unravelable and not-unravelable. Third, a procedure to generate efficient unrawing processes based on unravelability of the crossing sequence is explained. An object with an unravelable crossing sequence can be unrewound by pulling its both endpoints. Finally, examples of unrewinding process generation with our developed system are demonstrated.

2. Unknotting Process Generation

In this section, we briefly explain a method to generate possible processes for unknotting of a linear object, which is equivalent to its unrewinding. First, the state of a linear object can be topologically represented using three properties after projecting its shape on a projection plane. The first property is the crossing sequence. It is determined by numbering a crossing met first with tracing along the projected curve from one endpoint to the other. The $i$-th crossing is represented as symbol $C_i$. One endpoint where tracing starts is referred to as the left endpoint $E_l$ and that where tracing ends as the right endpoint $E_r$. The second property is the location of a pair of points at each crossing, that is, which point is upper/lower. The upper point of $i$-th crossing is described as symbol $u_i$ and the lower point of that as symbol $l_i$. The third property is the helix of each crossing. Let us define a crossing where the upper part overlaps first on the right side of the lower part and then overlaps on its left side as a left-handed helical crossing. Conversely, in a right-handed helical crossing, the upper part first overlaps on the left side of the lower part and then overlaps on its right side. The symbols $C_i^-$ and $C_i^+$ represent the $i$-th left- and right-handed helical crossing, respectively.

Next, we introduce basic operations described in Fig.1, corresponding to state transitions. Crossing operations $CO_{I_{II}}$, $CO_{II}$, and $CO_{IV}$ increases the number of crossings, while uncrossing operations $UO_{I_{II}}$, $UO_{II}$, and $UO_{IV}$ decrease the number. Arranging operation $AO_{III}$ does not change the number of crossings but permutes their sequence. Each basic operation can be applied to specific subsequences of crossings. Let us investigate subsequences to which each operation is applicable. Operation $UO_1$ is applicable to a subsequence represented as follows:

$$\cdots C_i^{u/I} - C_i^{l/u} - \cdots .$$ (1)

That is, two crossing points corresponding to one crossing $C_i$, should be adjacent to each other in applying $UO_1$. Operation $UO_1$ is applicable to subsequences described as follows:

$$\cdots C_i^{u/I} - C_i^{u/I} - \cdots C_i^{l/u} - C_i^{l/u} - \cdots ,$$ (2)

$$\cdots C_i^{u/I} - C_i^{u/I} - \cdots C_i^{l/u} - C_i^{l/u} - \cdots .$$ (3)
That is, two upper crossing points $C_i^u$ and $C_j^u$, should be adjacent to each other and the corresponding lower crossing points $C_i^l$ and $C_j^l$, should also be adjacent to each other. Operation $UO_{IV}$ is applicable to subsequences represented as follows:

$$E_i - C_i^u/l - \cdots - C_i^l/u - \cdots,$$

$$\cdots - C_j^u/l - \cdots - C_j^l/u - E_r.$$  

That is, a crossing adjacent to an endpoint can be deleted by operation $UO_{IV}$. Operation $AO_{III}$ is applicable to a subsequence represented as permutation of the following three subsequences: $\alpha$, $\beta$, and $\gamma$, e.g., $\cdots - \beta - \gamma - \alpha - \cdots$:

$$\alpha: \cdots - C_i^u/l - C_i^l/u - \cdots,$$

$$\beta: \cdots - C_j^u/l - C_j^l/u - \cdots,$$

$$\gamma: \cdots - C_k^u/l - C_k^l/u - \cdots.$$  

That is, three crossings consisting of three segments one of which overlaps the others can be permuted by operation $AO_{III}$. Uncrossing operations $UO_{I}$, $UO_{II}$, and $UO_{IV}$ and arranging operation $AO_{III}$ are applicable to their specific crossing subsequences indicated above. Once the initial and the objective crossing states of a linear object are given, we can generate possible sequences of crossing state transitions, that is, possible processes of unknotting manipulation by repeating detection of applicable subsequences of individual operations and deletion/permutation of relevant crossings.

Fig. 2 shows an example of automatic generation of possible unknotting processes by our developed system. Required manipulation corresponds to untying a slipknot. Assuming that only uncrossing operations can be used, i.e., without operation $AO_{III}$, 14 crossing states and 39 state transitions are derived as shown in Fig. 2. Including operation $AO_{III}$, we can derive 21 crossing states and 68 state transitions.
3. Introduction of Unravelable Crossing Sequence

Once the current crossing state of a linear object is identified, we can unravel the object using the method proposed in the previous section. To identify the crossing state completely, the location at each crossing should be known. Morita recognized the state of a linear object with a 9-eye stereo camera (Morita et al., 2003) and Matsuno identified it utilizing variance of luminance at crossings (Matsuno et al., 2005). Now, let us assume that only silhouette, that is, 2D information about the object state is available. It means that the location and the helix of any crossing cannot be identified. Then, we can perform operation $\text{UO}_i$ even if the location at crossing $C_i$ shown in Fig.1-(a) is unknown. Operation $\text{UO}_{IV}$ can also be realized regardless of the location at crossing $C_i$ shown in Fig.1-(d). Contrary, whether operations $\text{UO}_{II}$ and $\text{AO}_{III}$ can be applied depends on the location of crossings. Fig.3 shows examples of crossings with a subsequence to which operations $\text{UO}_{II}$ and $\text{AO}_{III}$ are applicable but with locations to which they cannot be applied. Any knot can be unknotted by applying operations $\text{UO}_{IV}$ alone (Wakamatsu et al., 2006). Note that the state transition graph shown in Fig.2 includes unknotting processes consisting of only $\text{UO}_{IV}$ operations. This implies that a raveled linear object can be unraveled by applying operations $\text{UO}_{IV}$ alone regardless of the location at each crossing. Recall that we often search for an endpoint and manipulate it to unravel a self-entwined rope. However, such manipulation may be not efficient when the object is raveled intricately, i.e., it has many crossings. In this section, we propose a method for generating efficient unraveling processes of a linear object based on its crossing sequence, i.e., its silhouette.
First, we define a knot in which some crossings remain even if all possible operations $UO_{I}$, $UO_{II}$, and $AO_{III}$ are applied as a \textit{tightenable knot}. For example, an overhand knot and a figure-of-eight knot are tightenable knots. Contrary, a knot which can be unknotted completely by applying operations $UO_{I}$, $UO_{II}$, and/or $AO_{III}$ is defined as an \textit{untightenable knot}. The untightenable knot is unknotted when its both endpoints are pulled away from each other. We can check whether a knot is tightenable or untightenable from its crossing state description (Wakamatsu et al., 2006).

Fig. 4-(a-1) illustrates the silhouette of a knot with 3 crossings. Its crossing sequence is described as follows:

\[ E_{I} - C_{1} - C_{2} - C_{3} - C_{1} - C_{2} - C_{1} - E_{I}. \]  

(9)

Knots shown in Fig. 4-(a-2) through (a-4) have the same crossing sequence. They include subsequence $\cdots - C_{3} - C_{3} - \cdots$ to which operation $UO_{I}$ can be applied. When crossing $C_{3}$ is deleted, it is found that crossing $C_{2}$ can also be deleted by application of operation $UO_{I}$. After deletion of crossing $C_{2}$, we can delete crossing $C_{1}$ by applying operation $UO_{I}$ once more. This means that knots shown in Fig. 4-(a-2) through (a-4) are untightenable. Any knot with the crossing sequence described by eq. (9) can be unraveled by pulling its both endpoints regardless of the location at each crossing. In this paper, we define such crossing sequence as an \textit{unravelable crossing sequence}. An untightenable knot has an unravelable crossing sequence.

A knot shown in Fig. 4-(b-1) also has 3 crossings, sequence of which is as follows:
\[ E_i - C_1 - C_2 - C_3 - C_1 - C_2 - C_3 - E_r . \] (10)

It is equivalent to that of knots shown in Fig. 4-(b-2) through (b-4). Knots in Fig. 4-(b-3) and (b-4) are both untightenable, but the knot in Fig. 4-(b-2) corresponds to an overhand knot, that is, it is tightenable. This implies that a tightenable knot with the crossing sequence described by eq.(10) exists. Consequently, such crossing sequence is not unravelable. Note that knots in Fig. 4-(b-3) and (b-4) can be unraveled, but they can not be distinguished from the knot in Fig. 4-(b-2) when 3D information, i.e., the location at each crossing is not given. Thus, we can categorize the crossing sequence of a knot into two types: unravelable and not-unravelable. The former can be unraveled by pulling its both endpoints regardless of the location at each crossing, while the latter may be tightened according to the location when its both endpoints are pulled.

Fig. 5 shows looped prime knots in knot theory. We can not reduce the number of crossings of these knots even if any operation corresponding to Reidemister move (Adams, 1994) is applied. They are closely related to tightenable knots. Let us discuss the relationship between looped prime knots and unravelable crossing sequences. If the looped prime knot with 3 crossings is cut as shown in Fig. 5-(a), its crossing state is described as follows:

\[ E_i - C_1^{\text{ur}} - C_2^{\text{ur}} - C_1^{\text{ur}} - C_2^{\text{ur}} - C_3^{\text{ur}} - E_r . \] (11)

If the crossing state of an unlooped linear object is described by eq.(11), it is equivalent to an overhand knot. If the object has 3 crossings but their sequence differs from eq.(11), it can be unknotted by applying operation UO_{I}, UO_{II}, and/or A0_{III}. Consequently, a linear object with 3 crossings can be unraveled by pulling both endpoints if and only if it does not have a not-unravelable crossing sequence: \( E_i - C_1 - C_2 - C_3 - C_1 - C_2 - C_3 - E_r . \)

Fig. 5-(b) shows the looped prime knot with 4 crossings. Cutting the knot as shown in Fig. 5-(b-1) and tracing it counterclockwise from one endpoint, the crossing sequence is described as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_1 - C_2 - C_4 - E_r . \] (12)

In the case of Fig. 5-(b-2) and (b-3), the crossing sequence is described as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_2 - C_3 - C_4 - E_r . \] (13)

A figure-of-eight knot has this crossing sequence. A knot with the crossing sequence described by eq.(12) or (13) may be tightened. This implies that a linear object with 4 crossings is unraveled if it does not have the above two sequences.

There are two types of the looped prime knot with 5 crossings as shown in Fig. 5-(c). One type illustrated in Fig. 5-(c-1) has the crossing sequence as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_5 - C_1 - C_2 - C_3 - C_4 - C_5 - E_r . \] (14)
Figure 5. Looped prime knots

This sequence corresponds to that of a double overhand knot. The other type illustrated in Fig. 5-(c-2) through (c-5) has the following crossing sequences:

\[ \text{rl ECCCCCCCCCCE} \]

\[ \begin{align*}
E_7 & - C_1 - C_2 - C_3 - C_4 - C_5 - C_3 - C_2 - C_1 - C_4 - C_5 - E_r, \\
E_8 & - C_1 - C_2 - C_3 - C_4 - C_5 - C_1 - C_2 - C_3 - C_4 - C_5 - E_r, \\
E_9 & - C_1 - C_2 - C_3 - C_4 - C_2 - C_1 - C_5 - C_3 - C_4 - C_5 - E_r, \\
E_{10} & - C_1 - C_2 - C_3 - C_4 - C_5 - C_2 - C_1 - C_3 - C_5 - E_r.
\end{align*} \]

Then, a linear object with 5 crossings but without the crossing sequence described by eqs. (14) through (18) is unravelable. Thus, we can derive not-unravelable crossing sequences from looped prime knots in knot theory. If the crossing sequence of a linear object with \( n \) crossings does not include not-unravelable sequences with 3 through \( n \) crossings, it can be unraveled by pulling its both endpoint instead of applying \( n \) UO \text{IV} operations.
4. Procedure to Generate Efficient Unraveling Processes

In this section, we explain a procedure to generate unraveling processes. Let us assume that the silhouette of a linear object shown in Fig.6-(a-1) is given. Its crossing sequence is described as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_5 - C_3 - C_2 - C_1 - C_4 - C_3 - E_j. \] (19)

The above sequence corresponds to a not-unravelable sequence with 5 crossings. It means that the object may be raveled and tightened if its both endpoints are pulled. Then, let us consider application of operation UO<sub>IV</sub> so that the object does not include any not-unravelable sequence. If we apply operation UO<sub>IV</sub> to the left terminal segment, the object state changes into the state shown in Fig.6-(a-2). Its crossing sequence is described as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_3 - C_2 - C_1 - C_4 - C_3 - E_j. \] (20)

Figure 6. Unraveling processes

A set of closed regions surrounded by a linear object is defined as the inner region, and the other region in the projection plane as the outer region. Moreover, segments touch the outer region are referred to as outer segments, and segments do not touch as inner segments (Wakamatsu et al., 2006). In Fig.6-(a-2), the left terminal segment is an inner segment. When one of terminal segments is inner, we can not pull both endpoints sufficiently without changing the crossing sequence. So, we apply another operation UO<sub>IV</sub> to the left terminal segment. Then, the following sequence is derived:

\[ E_i - C_1 - C_2 - C_3 - C_1 - C_2 - C_3 - C_4 - C_3 - E_j. \] (21)

The above sequence is equivalent to the not-unravelable sequence with 3 crossings. This implies that additional UO<sub>IV</sub> operations are required to unravel the object. Contrary, if we
apply 2 consecutive UO\text{IV} operations to the right terminal segment as shown in Fig.6-(b), the crossing sequence becomes as follows:

\[ E_t - C_1 - C_2 - C_3 - C_4 - C_5 - C_6 - E_r. \]  

(22)

As this sequence differs from the not-unravelable sequence, the knot shown in Fig.6-(b-3), which is equivalent to that in Fig.4-(a-1), can be unraveled by pulling its both endpoints. Consequently, we can conclude that unraveling process shown in Fig.6-(b) is more efficient than that shown in Fig.6-(a). Thus, we can generate efficient unraveling processes of a linear object based on only its crossing sequence, \textit{i.e.,} its silhouette. This indicates that we may unravel a linear object without a stereo camera.

Not-unravelable sequences can be extracted from the list of looped prime knots in knot theory (Rolfsen, 1976). Let us define the following subsequence as a not-unravelable subsequence with 3 crossings:

\[ \ldots - C_i - \ldots - C_j - \ldots - C_k - \ldots - C_{k-1} - \ldots \]  

(23)

If the crossing state includes the above subsequence, the object has a part which may be tightened. The not-unravelable sequence described by eq.(10) is a kind of this subsequence. We can also define not-unravelable subsequences with \( n \) crossings referring to not-unravelable sequences. When such not-unravelable subsequence is detected from the crossing sequence, we delete a crossing included in the subsequence and nearest to one endpoint by applying operations UO\text{IV} repeatedly. Let \( C_{il} \) and \( C_{jr} \) be \( i \)-th and \( j \)-th crossing met first when we trace an object from the left and the right endpoint, respectively. When the object has \( n \) crossings, we assume that only operation UO\text{IV} is applied to delete \( k \) \((k=1, \ldots, n)\) crossings. Then, we check the number of remaining not-unravelable subsequences after deleting crossings \( C_{il} \) \((i=k, k-1, \ldots, 1, 0)\) and \( C_{jr} \) \((j=k-i)\). If the crossing sequence does not include any not-unravelable subsequence by \( n-3 \) crossings are deleted, the rest can be uncrossed by applying one pulling operation instead of some UO\text{IV} operations. This implies that the object can be unraveled efficiently. For example, the crossing sequence described by eq.(19) is equivalent to the not-unravelable sequence with 5 crossings and includes three not-unravelable subsequences with 3 crossings:

\[ \ldots - C_1 - \ldots - C_4 - C_5 - \ldots - C_1 - C_4 - C_5 - \ldots, \]  

(24)

\[ \ldots - C_2 - \ldots - C_4 - C_5 - \ldots - C_2 - \ldots - C_4 - C_5 - \ldots, \]  

(25)

\[ \ldots - C_3 - C_4 - C_5 - C_3 - \ldots - C_4 - C_5 - \ldots. \]  

(26)

In this case, crossing \( C_{i1} = C_1 \) or \( C_{r1} = C_3 \) can be deleted by operation UO\text{IV}. If crossing \( C_5 \) is uncrossed, all these subsequences are deleted. Then, the object becomes unravelable. Contrary, subsequences described by eqs.(25) and (26) remain even if crossing \( C_1 \) is deleted. Consequently, we select application of operation UO\text{IV} to crossing \( C_5 \) as the first process for unraveling. After that, the object is completely unraveled by pulling its both endpoints. Thus, efficient unraveling processes of a linear object can be derived even if only its crossing sequence is identified.
5. Case Study

In this section, we discuss the effectiveness of our proposed method for efficient unraveling with some examples. Fig. 7 shows two examples of a raveled object. They correspond to not-unravelable sequences with 8 crossings. The crossing sequence in case-1 shown in Fig. 7-(a) is described as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_5 - C_6 - C_7 - C_8 - C_6 - C_5 - C_4 - C_3 - C_8 - C_7 - E_r. \]  

In case-2, the crossing sequence shown in Fig. 7-(b) is represented as follows:

\[ E_i - C_1 - C_2 - C_3 - C_4 - C_5 - C_6 - C_7 - C_8 - C_6 - C_7 - C_8 - C_5 - E_r. \]  

Figure 7. Silhouette of raveled objects

We developed a system to detect not-unravelable subsequences from a given crossing sequence. Using this system, it was found that the crossing sequence in case-1 include 49 not-unravelable subsequences and that in case-2 includes 19. Table 1 and 2 show the number of remaining not-unravelable subsequences after deleting crossings \( C_{li} \) and \( C_{ri} \) in case-1 and case-2, respectively. As shown in Table 1, when 3 crossings \( C_1, C_7, \) and \( C_i \) or \( C_4, C_7, \) and \( C_8 \) are deleted, the crossing sequence in case-1 does not include any not-unravelable subsequences. So, we can delete the rest crossings, i.e., we can unravel the object at once pulling away its both endpoints. This indicates that we can perform unraveling with less operations than unraveling in which all 8 crossings are deleted by operation \( UO_{IV} \). In case-2, 5 crossings \( C_4, C_5, C_6, C_7, \) and \( C_8 \) must be deleted to exclude not-unravelable subsequences from the crossing sequence as shown in Table 2. After that, we can unravel the object with one pulling operation. This is a more efficient unraveling process than that consisting of 8 \( UO_{IV} \) operations. But, we have to delete more crossings in case-2 to change the crossing sequence into the unravelable one than in case-1. Thus, it is found that the effectiveness of our proposed method depends on the crossing sequences. Efficient unraveling processes are derived from some crossing sequences but they are not from others. However, we can determine whether 3D information is needed for its efficient unraveling from its crossing sequence, i.e., its silhouette. For more efficient unraveling in case-2, for example, we have to identify the location of some crossings. If crossings \( C_3 \) and \( C_4 \) can be deleted by applying operation \( UO_{II} \), the object can be separated into two parts. Moreover, the left part is unravelable and it can be unraveled independently of the right part. Consequently, we should indentify the location of these crossings first. If their location
satisfies eq.(2) or (3), the object can be separated actually. Thus, we can rank crossings the location of which should be identified in order of importance for efficient unraveling. 3D information of not all crossings have to be identified when the unravelability, which is determined from the silhouette of a linear object, is considered.

| deleted crossings | remaining not-unravelable subsequences |
|-------------------|---------------------------------------|
| none              | 49                                    |
| $C_1$             | 21                                    |
| $C_7$             | 21                                    |
| $C_3, C_2$        | 7                                     |
| $C_1, C_7$        | 7                                     |
| $C_2, C_8$        | 7                                     |
| $C_1, C_2, C_3$   | 3                                     |
| $C_1, C_2, C_7$   | 0                                     |
| $C_1, C_7, C_8$   | 0                                     |
| $C_3, C_7, C_8$   | 3                                     |

Table 1. Unraveling process for case-1

| deleted crossings | remaining not-unravelable subsequences |
|-------------------|---------------------------------------|
| none              | 19                                    |
| $C_1$             | 11                                    |
| $C_3$             | 11                                    |
| $C_1, C_2$        | 7                                     |
| $C_1, C_5$        | 7                                     |
| $C_5, C_8$        | 8                                     |
| $C_1, C_2, C_3$   | 2                                     |
| $C_1, C_2, C_5$   | 5                                     |
| $C_1, C_5, C_8$   | 5                                     |
| $C_5, C_7, C_8$   | 4                                     |
| $C_1, C_2, C_3, C_4$ | 1                                |
| $C_1, C_2, C_3, C_5$ | 2                                |
| $C_1, C_2, C_5, C_8$ | 4                                |
| $C_1, C_5, C_7, C_8$ | 2                                |
| $C_5, C_6, C_7, C_8$ | 2                                |
| $C_1, C_2, C_3, C_4, C_5$ | 1                                |
| $C_1, C_2, C_3, C_5, C_8$ | 1                                |
| $C_1, C_2, C_5, C_7, C_8$ | 1                                |
| $C_1, C_5, C_6, C_7, C_8$ | 1                                |
| $C_4, C_5, C_6, C_7, C_8$ | 0                                |

Table 2. Unraveling process for case-2
6. Conclusions

A planning method for unraveling deformable linear objects based on their silhouette was proposed. First, an unknotting process of a linear object, which is equivalent to its unraveling process, was represented as a sequence of crossing state transitions. It can be generated on a computer if 3D information about the current crossing state is given. Second, the crossing sequence of a linear object, which corresponds to its 2D information, was categorized into two types: unravelable and not-unravelable. Third, a procedure to generate efficient unraveling processes based on unravelability of the crossing sequence was explained. An object with an unravelable crossing sequence can be unraveled by pulling its both endpoints. Finally, examples of unraveling process generation with our developed system were demonstrated. The crossing sequence is not sufficient information for deriving efficient unraveling processes, but it is useful for that.

7. References

Adams, C. C. (1994). The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots, Henry Holt & Co.
Hopcroft, J. E., Kearney, J. K., and Krafft, D. B. (1991). A Case Study of Flexible Object Manipulation. *Int. J. Robotics Research*, Vol.10, No.1, pp.41-50.
Inoue, H. and Inaba, M. (1984). Hand-eye Coordination in Rope Handling. *Robotics Research: The First International Symposium*, MIT Press, pp.163-174.
Ladd, A. M. and Kavraki, L. E. (2004). Using Motion Planning for Knot Untangling. *Int. J. Robotics Research*, Vol.23, No.7-8, pp.797-808.
Matsuno, T., Fukuda, T., and Arai, F. (2001). Flexible Rope Manipulation by Dual Manipulator System Using Vision Sensor, *Proc. of Int. Conf. Advanced Intelligent Mechatronics*, pp.677-682.
Matsuno, T., Tamaki, D., Arai, F., and Fukuda, T. (2005). Rope Structure Recognition for Manipulation Using Topological Model and Knot Invariant, *J. SICE*, Vol.41, No.4, pp.366-372, (in Japanese).
Morita, T., Takamatsu, J., Ogawara, K., Kimura, H., and Ikeuchi, K. (2003). Knot Planning from Observation, *Proc. of IEEE Int. Conf. Robotics and Automation*, pp.3887--3892.
Rolfsen, D. (1976). Knots and Links, Publish or Perish, Inc.
Saha, M. and Isto, P. (2006). Manipulation Planning for Deformable Linear Objects, *J. IEEE Transactions on Robotics*, Vol.23 No.6, pp.1141-1150.
Takamatsu, J., Morita, T., Ogawara, K., Kimura, H., and Ikeuchi, K. (2006). Representation for Knot-Tying Task, *J. IEEE Transactions on Robotics*, Vol.22 No.1, pp.65-78.
Wakamatsu, H., Arai, E., and Hirai, S. (2006). Knotting/Unknotting Manipulation of Deformable Linear Objects, *Int. J. Robotics Research*, pp.371-395.
Yamakawa, Y., Namiki, A., Ishikawa, M., and Shimojo, M. (2007). One-handed Knotting of a Flexible Rope with a High-speed Multifingered Hand having Tactile Sensors, *Proc. of IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp.703-708.
In this book, new results or developments from different research backgrounds and application fields are put together to provide a wide and useful viewpoint on these headed research problems mentioned above, focused on the motion planning problem of mobile robots. These results cover a large range of the problems that are frequently encountered in the motion planning of mobile robots both in theoretical methods and practical applications including obstacle avoidance methods, navigation and localization techniques, environmental modelling or map building methods, and vision signal processing etc. Different methods such as potential fields, reactive behaviours, neural-fuzzy based methods, motion control methods and so on are studied. Through this book and its references, the reader will definitely be able to get a thorough overview on the current research results for this specific topic in robotics. The book is intended for the readers who are interested and active in the field of robotics and especially for those who want to study and develop their own methods in motion/path planning or control for an intelligent robotic system.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hidefumi Wakamatsu, Eiji Arai and Shinichi Hirai (2008). Planning for Unraveling Deformable Linear Objects Based on Their Silhouette, Motion Planning, Xing-Jian Jing (Ed.), ISBN: 978-953-7619-01-5, InTech, Available from: http://www.intechopen.com/books/motion_planning/planning_for_unraveling_deformable_linear_objects_based_on_their_silhouette