On a system which allows us to simulate smoothing operation on knot projections using dynamic of spring

Meiji University School of Interdisciplinary Mathematical Sciences
The Department of Frontier Media Sciences

Ahara Lab

Rikiishi Yumu(B3)
Ahara Kazushi

Index

Abstract

* This system is named ‘BeadsKnot’.

* In order to draw a planer figure of a complicated knot by connecting some simple knots, we use simulation of physics with spring force simulation and repulsion.

* In this system, we can simulate smoothing operation and connecting operation for knot projections

Project BeadsKnot

* We make a system which allows us to draw & edit knot projections interactively.

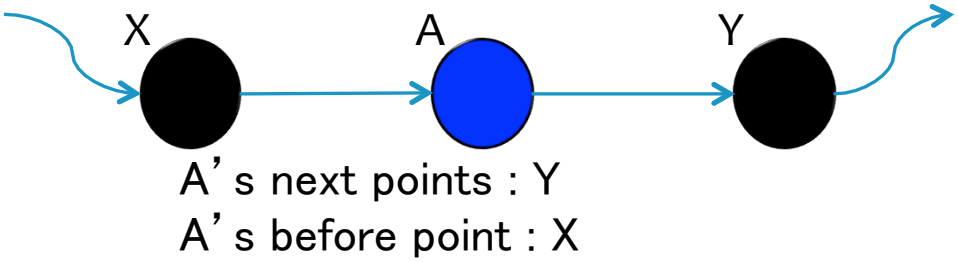
* We want to make this system as its first stage to make a system which allows us to simulate Reidemeister Moves for knot projections.

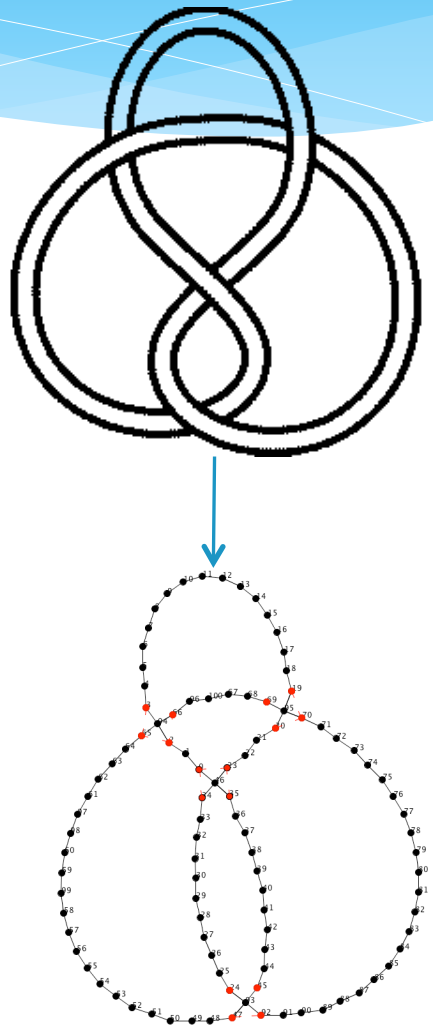
* Our system has operations of smoothing / un-smoothing and we believe that this will lead us to achieve the goal.

Composition of BeadsKnot

- Points

* In BeadsKnot, a knot projection consists of points connected with lines.
* Every point except on crossings has pointers of the next point and the before point.

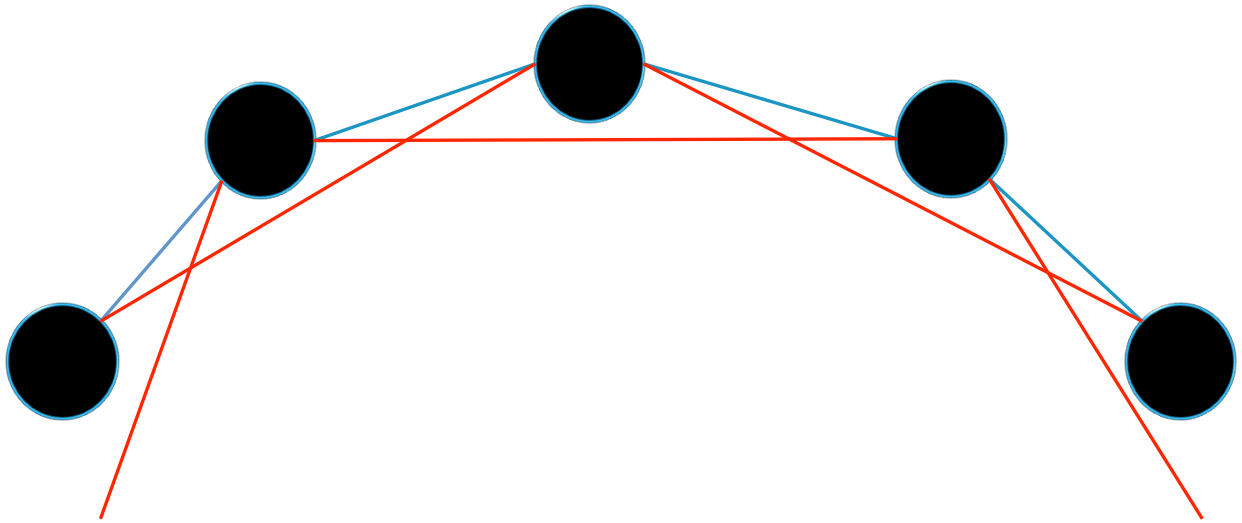




Composition of BeadsKnot

- Points

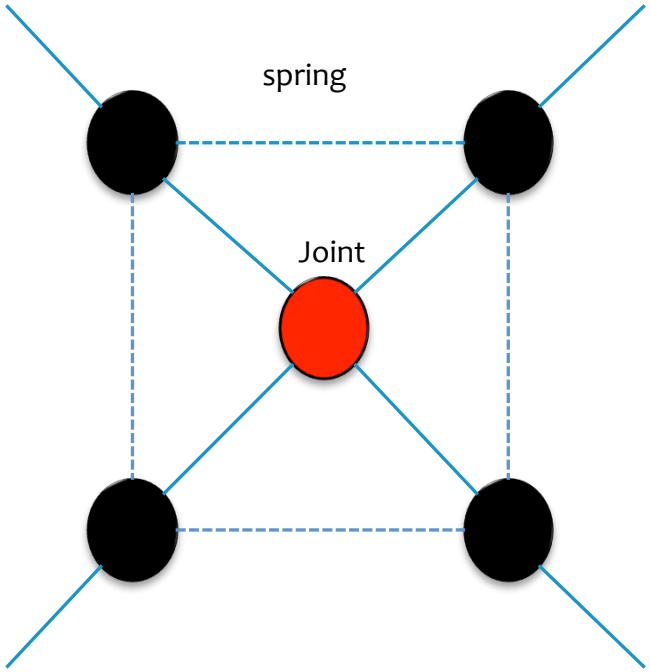
* We set a spring between each point of length a. (blue lines)
* We set a spring between before point and next point of length b .(red lines)
* $b = 2a - \varepsilon$



Composition of BeadsKnot

Special points “Joint”

* We call a point on a crossing ‘joint’.
* A joint has 4 pointers to the points nearby.
* We set additional springs of length $\sqrt{2}a$ among 4 points as in right figure.



Composition of BeadsKnot

Other……

* Repulsion acts between every two points of which the knot projection consists.
* Spring force and repulsion form a scheme of the simulation model.

Functions of BeadsKnot

About input

* By mouse dragging, we can draw a circled beads on the display.

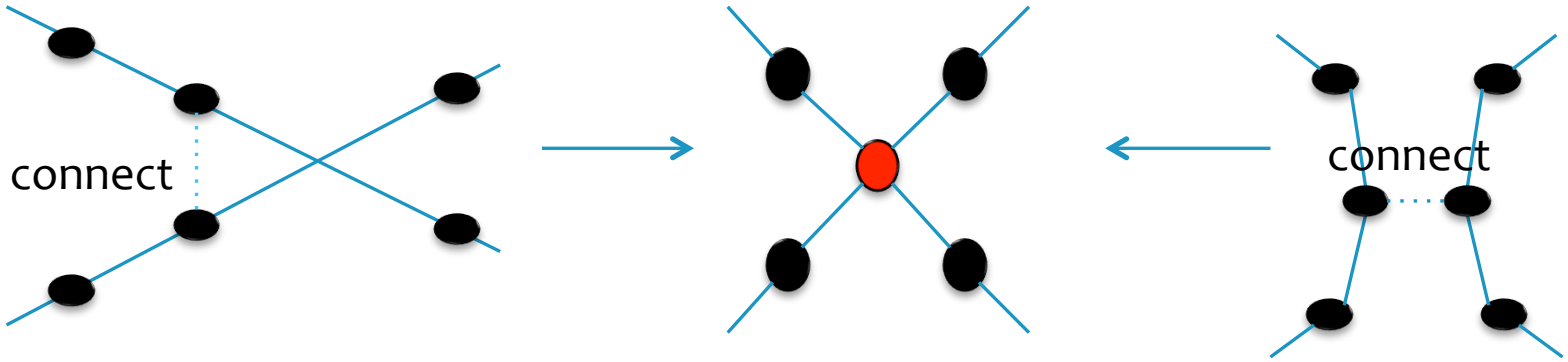
* We can add a circle in any position.

Functions of BeadsKnot

About Creating a Joint

* In case of mouse dragging.
  * →BeadsKnot replaces an intersection to a joint automatically.

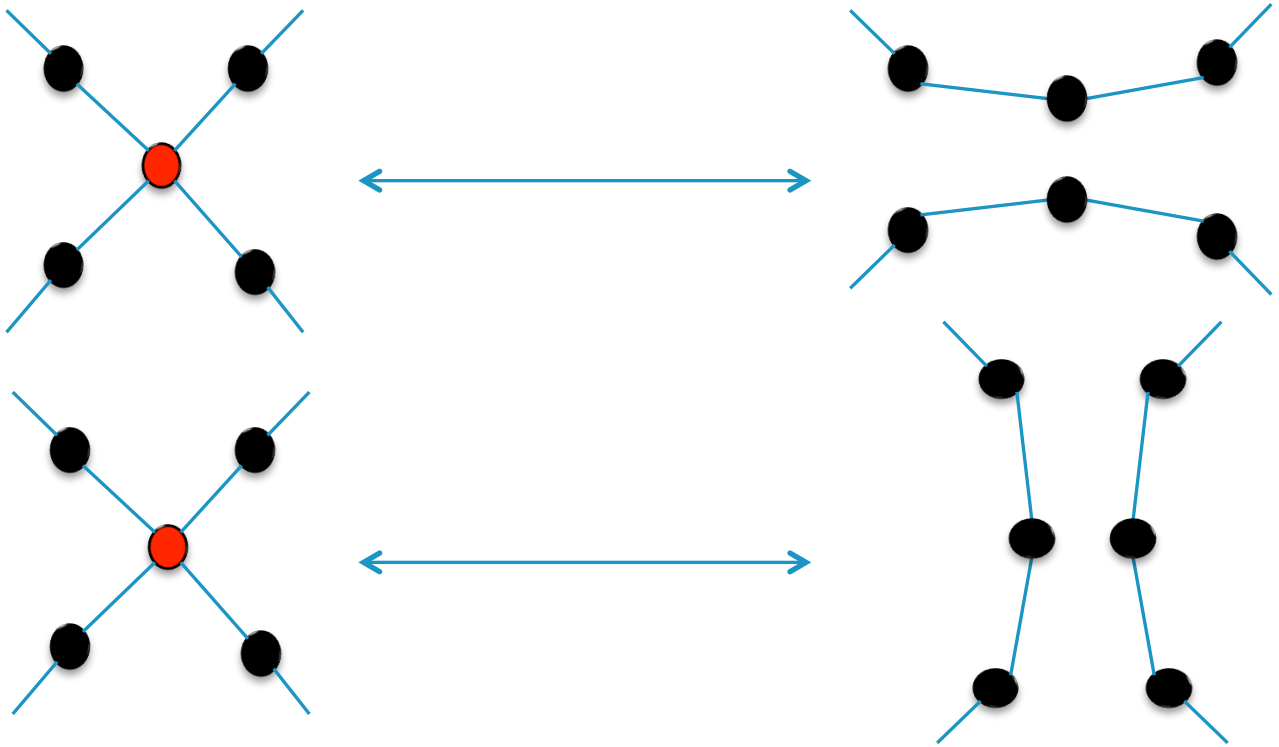
* clicking two points, we can made a joint connecting two points to each other.



Functions of BeadsKnot

Smoothing operation

* Choose a direction and click a joint.



Functions of BeadsKnot

About adding / deleting a point into / from a point sequence

Add a point here

Delete this point

* Demonstration

Algorithm

Basic info

* For each connected component, we prepare an array list of points.

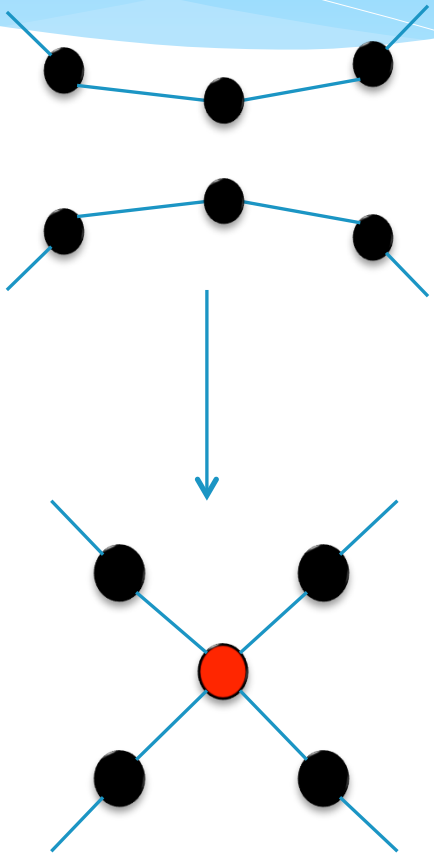
* Operations
* Each sequence of points has an orientation to determine ‘the next’ and ‘the before’.
* When we add a point in a row, the orientation is preserved.

Algorithm

- Creating a joint

1. We delete clicked 2 points and add a joint to Array of points
2. The next/before data of the deleted points are lost, and we check the consistency of the next/before data.

If we connect two connected components, we need to combine two array list of points.

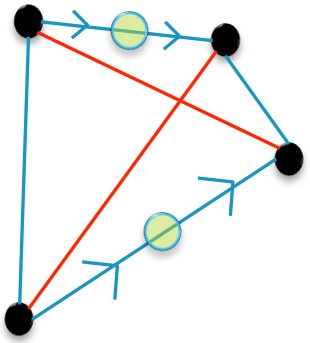


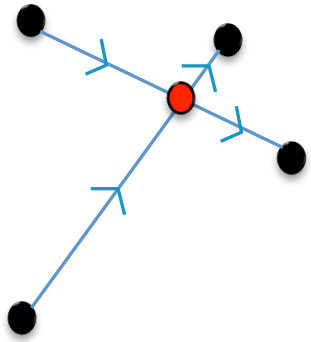
Algorithm

Where is a joint created?

* When we make a joint, neighbor 4 points of a joint is the before and the next points of the clicked 2 points.

* We make a quadrilateral by neighbor 4 points and we set a joint at the intersection of the diagonal of the quadrilateral.





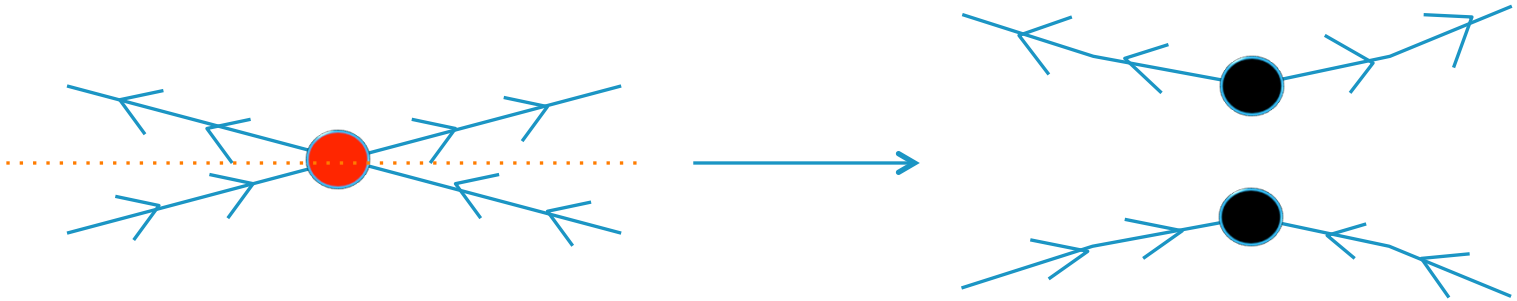
Algorithm

Smoothing

1. Delete a joint which is clicked.
2. Add 2 points to the array list of points in order to fill the break.
3. Determine the orientation around the 4 neighbors.

Algorithm

Arrange the direction of knots



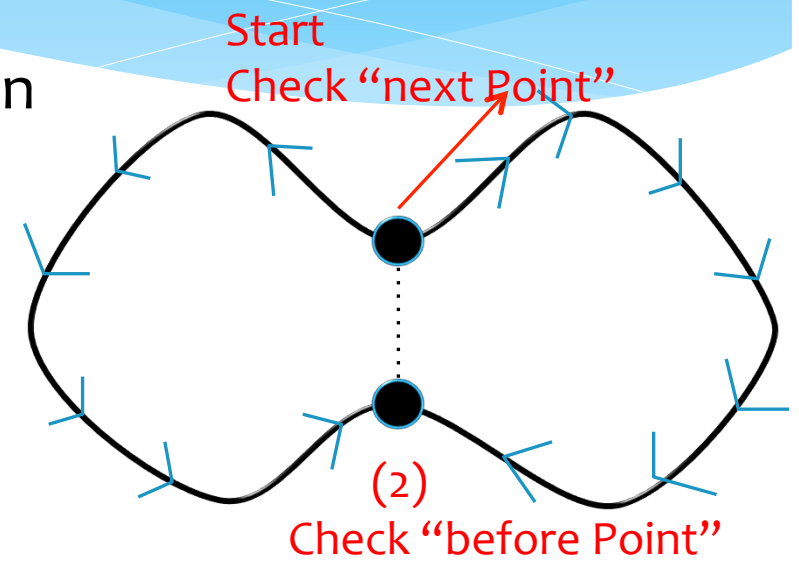
In this figure, the orientation of the knot happens contradiction at the black points
So, we need to rearrange the orientations for each segment.

Algorithm

Rearrangement of the orientation

1. After smoothing, we need to re-arrange the orientation of the knot projection.
2. We start at one of the two new points which was at a joint, we follow a path to the ‘next’ direction.
3. If we meet another joint or the other new point in the way, we check the ‘next’ direction.

If if happens contradiction, we follow a path the ‘before’ direction and re-arrange the before/next.



Conclusion & reflection

* Smoothing simulation works normally (probably),

* Sometimes we cannot operate smoothing smoothly.

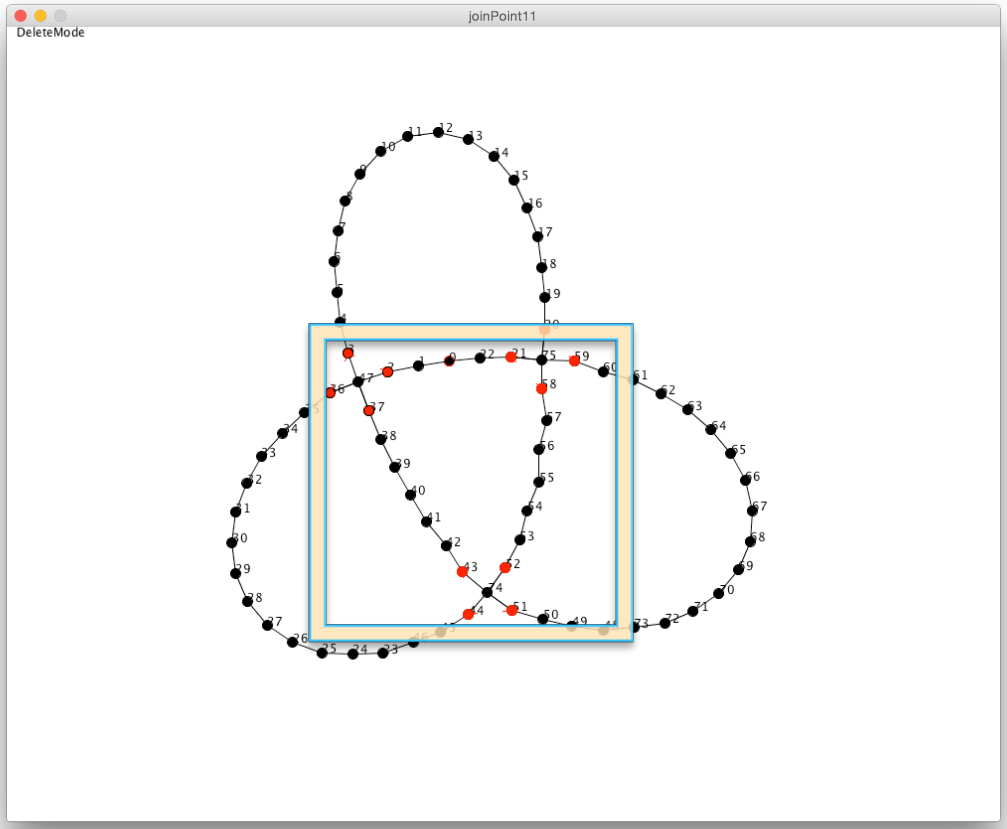
  We need more highly reliability.

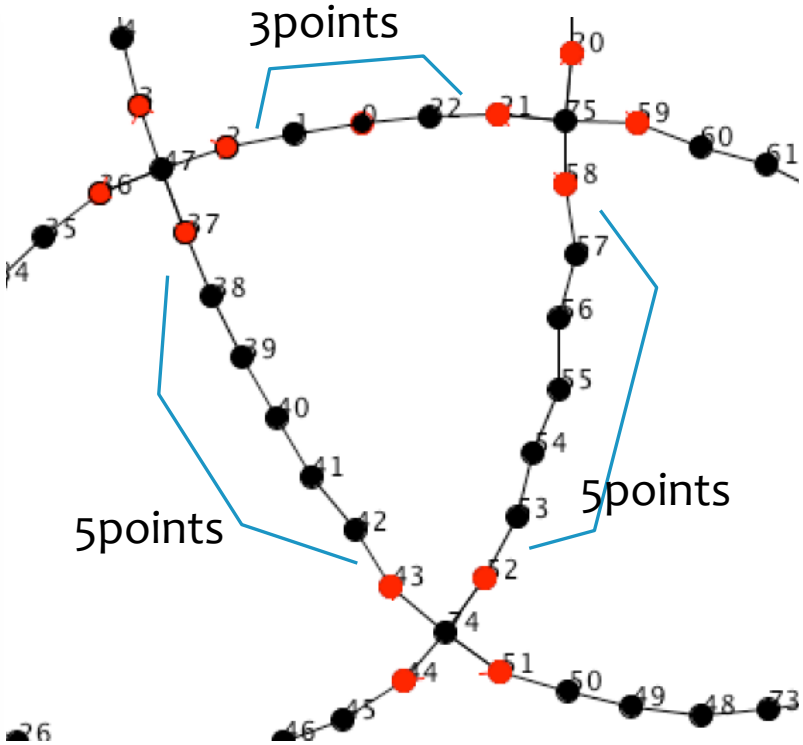
Future plan(1)

* We cannot determine which up or down edge is.

Future plan(2)

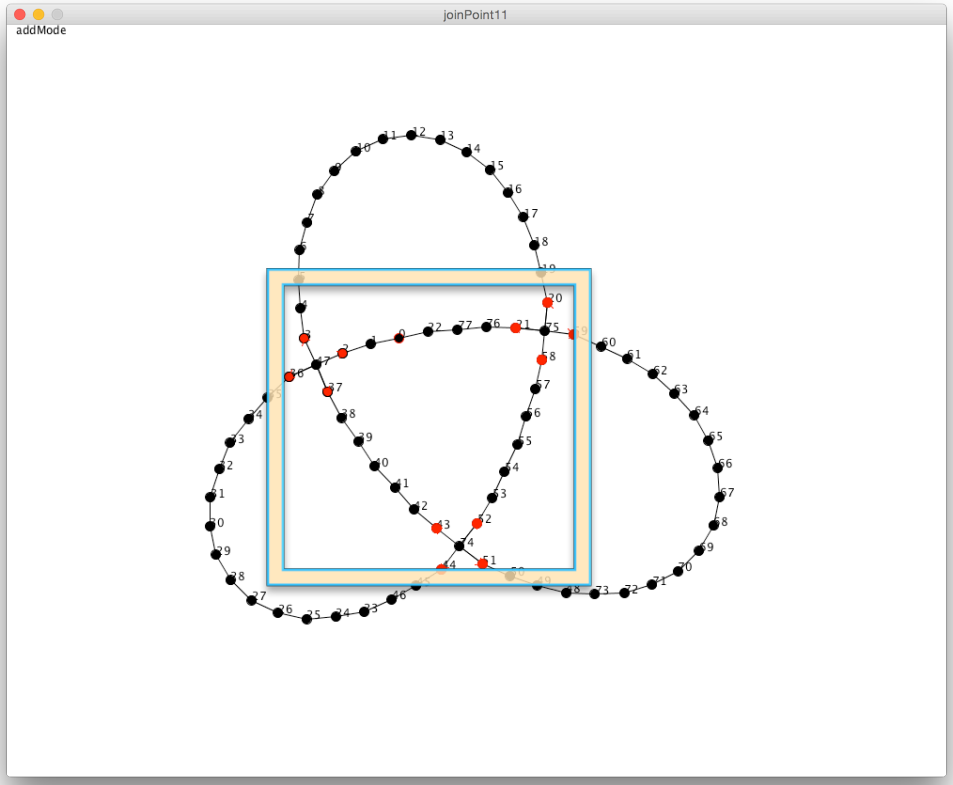
* We need to adjust the number of points automatically for each points of the knot projection.

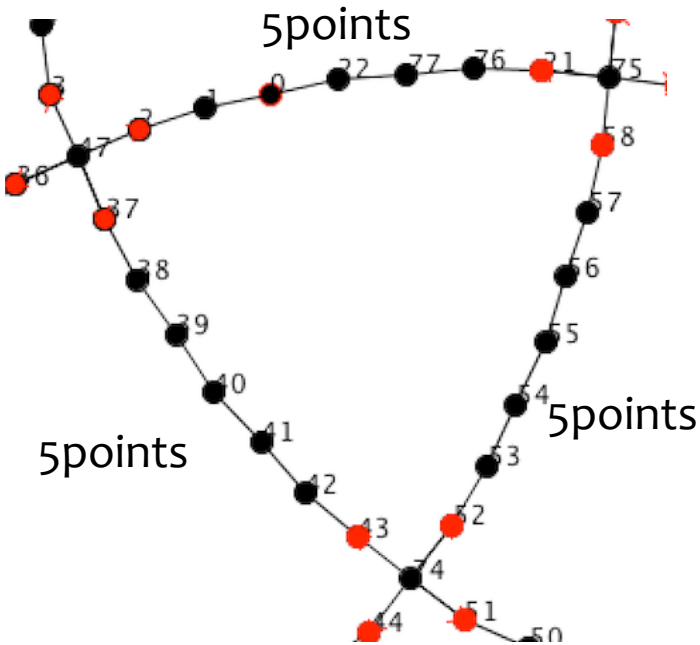




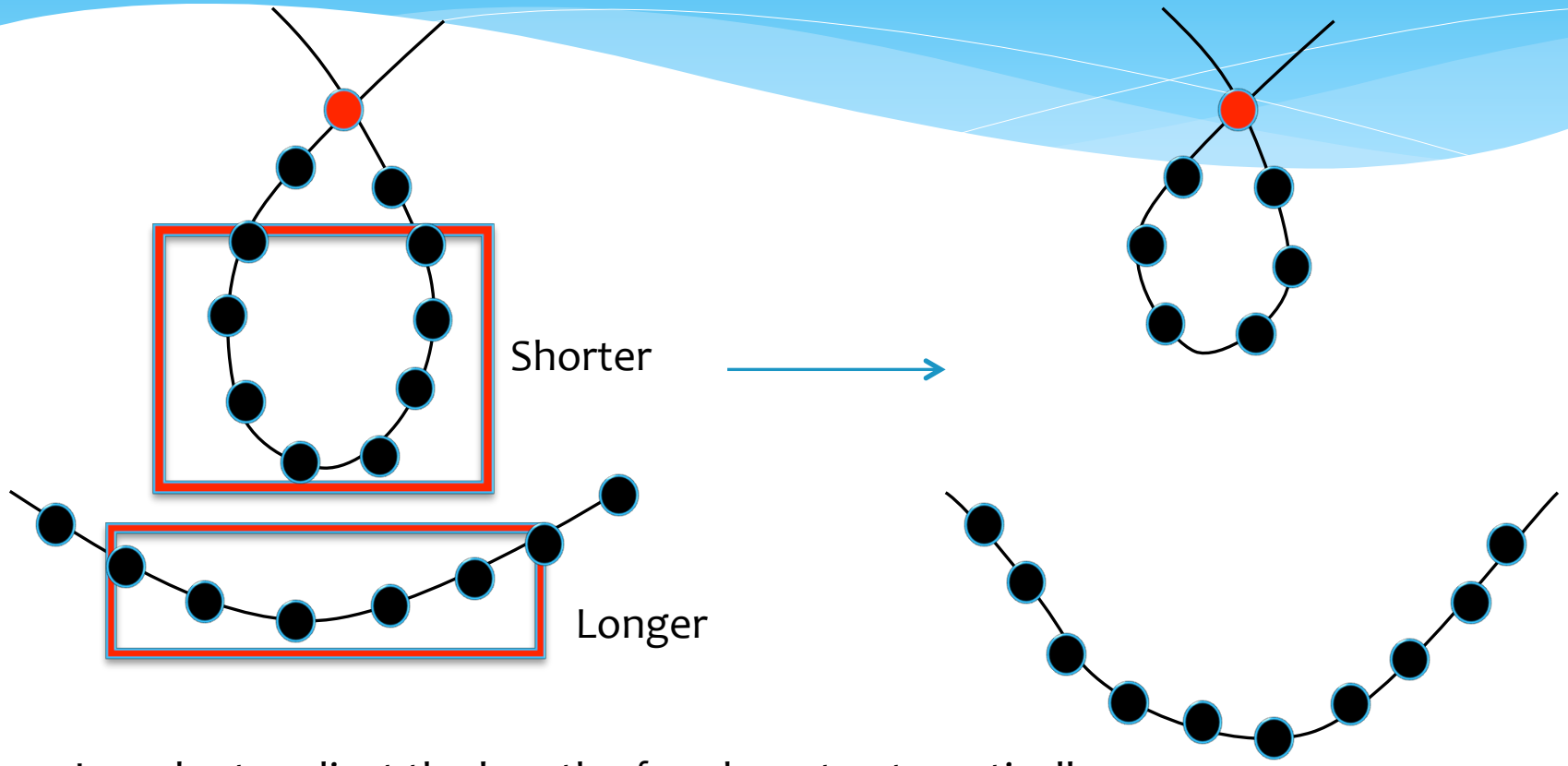
Future plan(2)

* We need to adjust the number of points automatically for each points of the knot projection.





Future plan(3)

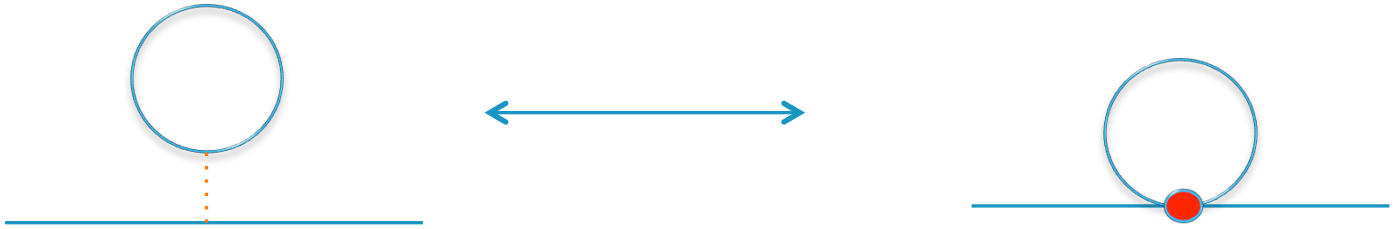


In order to adjust the length of each part automatically,
we need to check ‘appropriateness’.
(We have no plan.)

Future plan(4)

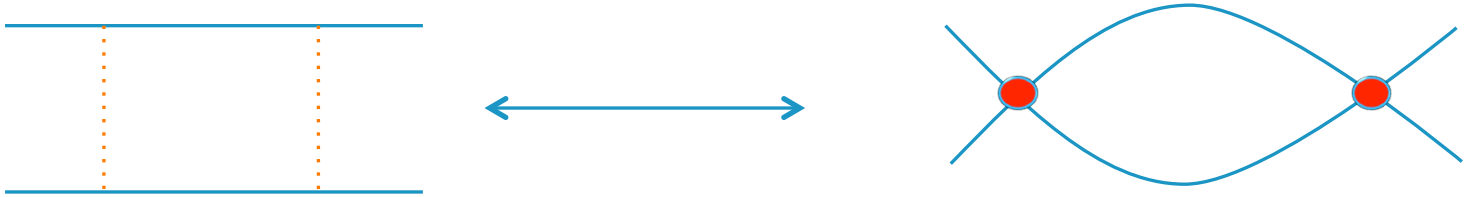
We want BeadsKnot to allowed us to do Reidemeister Move.

(1)



Connect by a joint/
smoothing

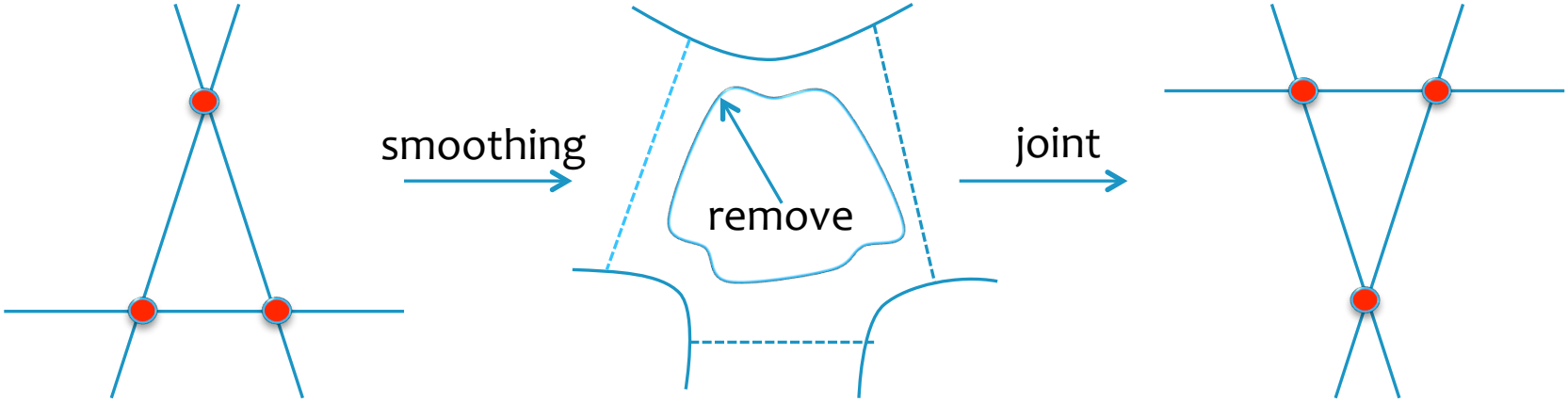
(2)



Future plan(4)

We want BeadsKnot to allowed us to do Reidemeister Move.

(3)



We want to make these operations automatic…