8 Appendix

8.1 P-data, an abstract representation for rope states

A common way to abstract the state space in knot tying is using the **P-data** representation [24]. The P-data representation translates a rope configuration to a matrix of discrete values that depends on the number of link intersections. The P-data algorithm stages are: (1) project the 3D rope onto a 2D on the horizontal plane. (2) Select rope direction by defining the head and tail of the rope. (3) Move from head to tail and count the number of intersections along the path, starting from 1 to N. Those intersections are also called crosses. Finally, (4) each intersection gets **over/under** value based on which segment is over the other in the height dimension and also gets a **sign** plus/minus. The sign defined as

$$sign = \frac{\overrightarrow{l}_{over} \times \overrightarrow{l}_{under}}{|\overrightarrow{l}_{over} \times \overrightarrow{l}_{under}|} \cdot \overrightarrow{e_z},$$

where e_z is the unit normal of the horizontal plane, and l_{over} and l_{under} are the two strands directional vectors. Examples of P-data of projected knots in Figure 2.

8.2 Collection process

We maintain a set of configurations we have already seen during data collection and their respective number of crosses, i.e. $D_Q = \{(q_i, \texttt{Cross}(\texttt{Top}(q_i))\}_i$. For every data collection iteration t, we load the simulation with a configuration sampled from D_Q, q_t, take a random curve c_t, and reach a new configuration q_{t+1} with a topological state s_{t+1}. If the number of crosses in $Cross(s_{t+1}) > Cross(s_t)$ the transition $(q_t, s_t, c_t, q_{t+1}, s_{t+1})$ is added to D, and the configuration q_{t+1} is added to D_Q.

8.3 TWISTED Algorithm

Our TWISTED algorithm in algorithm 1

Algorithm 1 TWISTED algorithm

Input q_{init} Initial configuration state and s_g topological goal state
Output Low-level plan if found

$init : T, \mathbf{P}$ ▷ see **data structures**
$s_{init} = Top(q_{init})$
populate $\mathbf{P}$ with plans from s_{init}
while Not timeout **do**
    $s_{selected} = SelectTopologicalState()$
    $P_{selected} = SelectPlan(s_{selected})$
    $q_{selected} = SelectConfiguration(s_{selected})$
    $PlanFound = FollowPlan(q_{selected}, P_{selected})$
    if $PlanFound$ **then**
        $ReturnPlan$
    else
        $RandomExpand()$
    end if
end while

8.4 Inverse model

Our inverse model architecture in figure 4

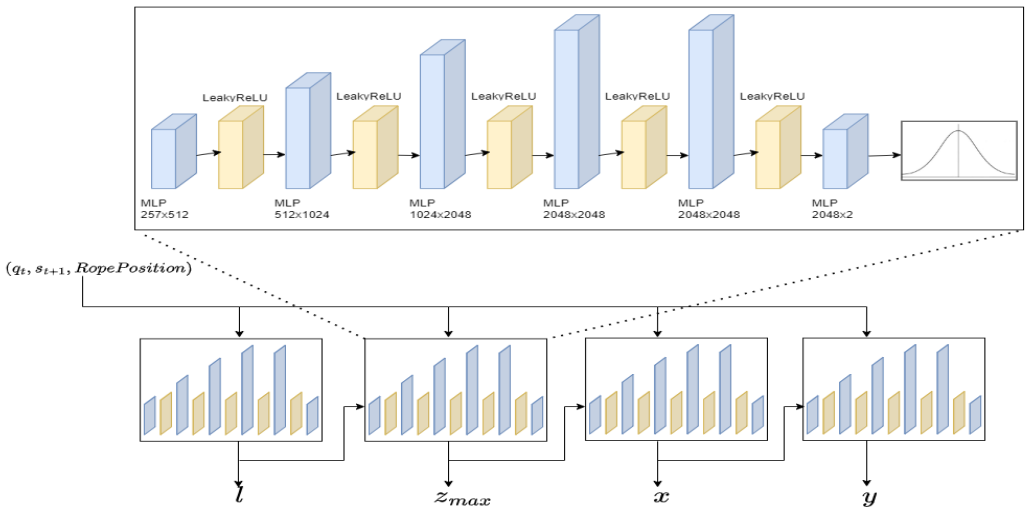

Figure 4: Inverse model - Auto-regressive Stochastic Network. The network predicts an action in an auto-regressive manner: first is predicts the link index $l \in [1, L]$, then the height of the curve z_{max}, finally it predicts the x and y coordinates of the curve. All predictions are stochastic (Multinominal for link index, and Gaussian otherwise). Besides the previous elements, the input of each element includes the current configuration q_t, the next topological state s_{t+1}, and the link positions of all the rope links. The weights of the sub-components are not shared.