

A Dynamics model for each manipulation primitive

A.1 Hit

The state is represented by the object position \mathbf{x} , and the action is the impact \mathbf{I} . The model parameters include the object mass m and the friction coefficient μ . The motion equation is

$$\mathbf{x}^{\text{des}} = \mathbf{x}_0 + \frac{\mathbf{I}}{m}t - \frac{1}{2}\mu g t^2, \quad (1)$$

Given the initial state \mathbf{x}_0 and the target \mathbf{x}^{des} , the advantage function is

$$A(\mathbf{x}, \mathbf{I}) = -(\|\mathbf{x} - \mathbf{x}^{\text{des}}\|^2 + 0.01\|\mathbf{I}\|^2). \quad (2)$$

We applied TT to approximate the advantage function $A(\mathbf{x}, \mathbf{I})$, and the correct impact is computed by maximizing (2).

A.2 Push

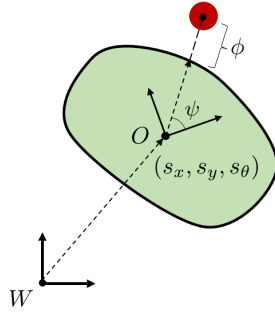


Figure 1: Illustration of pushing dynamics.

The state is characterized by $[s_x, s_y, s_\theta, \psi, \phi]$, and the action is denoted as $[f_x, f_y, \dot{\psi}, \dot{\phi}]$. Here, $[s_x, s_y, s_\theta] \in SE(2)$ denotes the position and orientation of the object in the world frame. ψ is the relative angle of the contact point in the object frame. ϕ represents the distance between the contact point and the object surface. $\mathbf{f} = [f_x, f_y]^\top$ are the forces exerted on the object, while $\mathbf{v}_p = [\dot{\psi}, \dot{\phi}]^\top$ represents the angular and translational velocities of the robot's end-effector. The physical parameters include the object mass m , radius r , and the friction coefficient μ between the object and table.

The applied force on this object can be mapped to its resulting velocity through a convex limit surface convex approximation [1], resulting in a sub-level set

$$H(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{L}\mathbf{w}, \quad (3)$$

where $\mathbf{L} = \text{diag}[f_{\max}^{-1}, f_{\max}^{-1}, m_{\max}^{-1}]$, with f_{\max} as the maximum friction force between object and table, and m_{\max} as the maximum torsional friction.

The robot dynamics is defined based on the Quasi-Static approximation and the limit surface, resulting in a similar expression as [2], namely

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{R}\mathbf{t} \\ \mathbf{v}_p \end{bmatrix} = \begin{bmatrix} \mathbf{R}\mathbf{L}\mathbf{w} \\ \mathbf{v}_p \end{bmatrix}, \quad (4)$$

with

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

$$\mathbf{w} = \begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \mathbf{J}^\top \mathbf{f} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -p_y & p_x \end{bmatrix} \mathbf{f}, \quad (6)$$

where \mathbf{R} is the rotation matrix and \mathbf{w} denotes the applied pusher wrench. \mathbf{J} is the Jacobian matrix of the contact point in the body frame. The contact position $[p_x, p_y]^\top$ in the object frame can be computed by

$$p_x = (r(\psi) + \phi) \cos(\psi), \quad p_y = (r(\psi) + \phi) \sin(\psi), \quad (7)$$

for any shape that can be parameterized radially with the radial distance described as $r(\psi)$.

We parameterize the object shape using a concatenation of Bézier curves, with the weight matrix C defined as

$$C = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & \cdots \\ 0 & 1 & \cdots & 0 & 0 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 1 & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & 1 \\ 1 & 0 & \cdots & \cdots & \cdots & 0 & 0 \end{bmatrix}, \quad (8)$$

where the pattern $\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$ is repeated for each junction of two consecutive Bézier curves. For two concatenated cubic Bézier curves, each composed of 4 Bernstein basis functions, we can see locally that this operator yields a constraint of the form

$$\begin{bmatrix} w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}, \quad (9)$$

which ensures that $w_4 = w_5$. These constraints guarantee that the last control point and the first control point of the next segment are the same, therefore enforcing C_0 continuity of the reconstructed shape. Fig. 2 shows an example of reconstructing the shape of a mustard bottle from the YCB dataset.

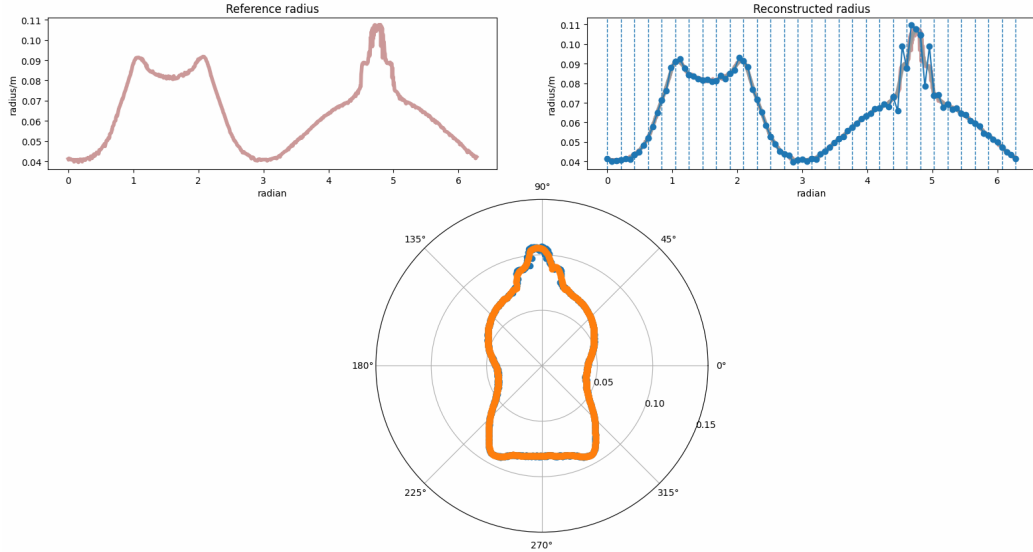


Figure 2: Shape parametrization of a mustard bottle using basis functions.

43 A.3 Reorientation

44 The state is the orientation angle θ , and the control input is the normal force f_n between the gripper
45 and the object. We build the dynamics model of the reorientation primitive based on [3] as

$$\begin{aligned} I\ddot{\theta} &= \tau_g + 2\tau_f, \\ \dot{\theta} &= \dot{\theta}_0 - \ddot{\theta}\Delta t, \end{aligned} \quad (10)$$

46 where $\tau_f = \mu_t f_n^{1+\gamma}$ is the torsional sliding friction between robot gripper and the object. In this
47 work, we set $\gamma = 0$. μ_t is the torsional friction coefficient, which is related to the materials and
48 normal force distribution. $\tau_g = mgl\sin(\theta)$ is the gravity torque. We therefore include μ_t , object
49 mass m and length l as the model parameters. The task is to rotate the object from a vertically
50 downward to a vertically upward position. To achieve this, the object is given an initial angular
51 velocity $\dot{\theta}_0$ by swinging the robot arm.

52 B Background of Tensor Train

53 B.1 Tensors as Discrete Analogue of a Function

54 A multivariate function $P(x_1, \dots, x_d)$ defined over a rectangular domain made up of the Cartesian
55 product of intervals (or discrete sets) $I_1 \times \dots \times I_d$ can be discretized by evaluating it at points in the
56 set $\mathcal{X} = \{(x_1^{i_1}, \dots, x_d^{i_d}) : x_k^{i_k} \in I_k, i_k \in \{1, \dots, n_k\}\}$. This gives us a tensor \mathcal{P} , a discrete version
57 of P , where $\mathcal{P}_{(i_1, \dots, i_d)} = P(x_1^{i_1}, \dots, x_d^{i_d}), \forall (i_1, \dots, i_d) \in \mathcal{I}_{\mathcal{X}}$, and $\mathcal{I}_{\mathcal{X}} = \{(i_1, \dots, i_d) : i_k \in$
58 $\{1, \dots, n_k\}, k \in \{1, \dots, d\}\}$. The value of P at any point in the domain can then be approximated
59 by interpolating between the elements of the tensor \mathcal{P} .

60 B.2 Tensor Networks and Tensor Train Decomposition

61 Naively approximating a high-dimensional function using a tensor is intractable due to the combi-
62 natorial and storage complexities of the tensor ($\mathcal{O}(n^d)$). Tensor networks mitigate the storage issue
63 by decomposing the tensor into factors with fewer elements, akin to using Singular Value Decom-
64 position (SVD) to represent a large matrix. In this paper, we explore the use of Tensor Train, a type
65 of Tensor Network that represents a high-dimensional tensor using several third-order tensors called
66 *cores*.

67 We can access the element (i_1, \dots, i_d) of the tensor in this format simply given by multiplying
68 matrix slices from the cores:

$$\mathcal{P}_{(i_1, \dots, i_d)} = \mathcal{P}_{:,i_1,:}^1 \mathcal{P}_{:,i_2,:}^2 \cdots \mathcal{P}_{:,i_d,:}^d, \quad (11)$$

69 where $\mathcal{P}_{:,i_k,:}^k \in \mathbb{R}^{r_{k-1} \times r_k}$ represents the i_k -th frontal slice (a matrix) of the third-order tensor \mathcal{P}^k .
70 For any given tensor, there always exists a TT decomposition [4]. This low-rank structure further
71 facilitates sampling and optimization for robot planning and control.

72 There are several ways to acquire a TT model, including TT-SVD [4] and TT-Cross [5, 6]. TT-SVD
73 extends the SVD decomposition from matrix level to a high-dimensional tensor level. However, it
74 needs to store the full tensor first, which is impractical to very high-dimensional functions. TT-Cross
75 solves this issue by selectively evaluating function P on a subset of elements, avoiding the need to
76 store the entire tensor.

77 B.3 Function approximation using Tensor Train

78 Given the discrete analogue tensor \mathcal{P} of a function P , we obtain the continuous approximation
79 by spline-based interpolation of the TT cores corresponding to the continuous variables only. For
80 example, we can use linear interpolation for the cores (i.e., between the matrix slices of the core)
81 and define a matrix-valued function corresponding to each core $k \in \{1, \dots, d\}$,

$$P^k(x_k) = \frac{x_k - x_k^{i_k}}{x_k^{i_{k+1}} - x_k^{i_k}} \mathcal{P}_{:,i_{k+1},:}^k + \frac{x_k^{i_{k+1}} - x_k}{x_k^{i_{k+1}} - x_k^{i_k}} \mathcal{P}_{:,i_k,:}^k, \quad (12)$$

where $x_k^{i_k} \leq x_k \leq x_k^{i_k+1}$ and $\mathbf{P}^k : I_k \subset \mathbb{R} \rightarrow \mathbb{R}^{r_{k-1} \times r_k}$ with $r_0 = r_d = 1$. This induces a continuous approximation of P given by

$$P(x_1, \dots, x_d) \approx \mathbf{P}^1(x_1) \cdots \mathbf{P}^d(x_d). \quad (13)$$

This allows us to selectively do the interpolation only for the cores corresponding to continuous variables, and hence we can represent functions in TT format whose variables could be a mix of continuous and discrete elements.

B.4 Global Optimization using Tensor Train

An arbitrary function can be transformed into a nonnegative function in TT format, which can be interpreted as a probability density function. The efficient sampling techniques for density functions in TT format allow to pick samples of only high-density regions which in turn correspond to the optima. In practice, the chosen number of prioritized samples $N \geq 1$ and the sample(s) with the highest density (or least cost) is used to represent the optima. This leads to the near-global solutions, which can be further refined using local optimization techniques such as Newton-type optimization for continuous variables. Such process is gradient-free, and it can handle a mix of continuous and discrete variables.

References

- [1] J. Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 372–377. IEEE, 2016.
- [2] F. R. Hogan and A. Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *International Journal of Robotics Research (IJRR)*, 39(7):755–773, 2020.
- [3] F. Viña Barrientos, Y. Karayiannidis, C. Smith, and D. Kragic. Adaptive control for pivoting with visual and tactile feedback. In *IEEE International Conference on Robotics and Automation, Stockholm, Sweden 16-21 May 2016*. Institute of Electrical and Electronics Engineers (IEEE), 2016.
- [4] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
- [5] I. Oseledets and E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.
- [6] D. V. Savostyanov and I. V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. *The 2011 International Workshop on Multidimensional (nD) Systems*, pages 1–8, 2011.