Sampling-based MPC for Contact-rich Skills

Yuezhe Zhang
Cognitive Robotics, 3mE, TU Delft
Y.ZHANG-130@student.tudelft.nl

TU Delft

1. Abstract

We propose a method for **sampling-based MPC** that uses **IsaacGym** as **dynamic models** and a platform to do **parallel sampling on a GPU**. We show the application for versatile contact-rich skills, including **navigation, collision avoidance, push, and pull**, without the need to build the contact models.

2. Motivation

In order for the robot to robustly operate in dynamic and uncertain environments, it must react to new situations with intelligent decision-making and fast execution. **Model Predictive Control (MPC)** addresses this problem via constrained optimization in a receding horizon way and has been widely used on real robotic systems [1]. However, most MPC methods have the following issues:

- **Inflexible convexification** of the constraints and cost functions for **high-dimensional systems**
- Hard to model **discontinuous contact skills**

3. Algorithm

Sampling-based MPCs, such as MPPI [2] and STORM [3], offer promising alternatives. These algorithms make no restrictions on the convexity, non-linearity, discontinuity of the dynamics and the costs. A general sampling-based MPC algorithm is summarized in Algorithm 1. The key idea of the algorithm is to **sample control sequences** from a given distribution and **forward simulate them in parallel** (e.g. on a GPU) using the system's dynamics to generate trajectories. Each simulated trajectory is then **evaluated against a cost function**. The cost of each trajectory is then converted to **update the distribution** of control sequences.

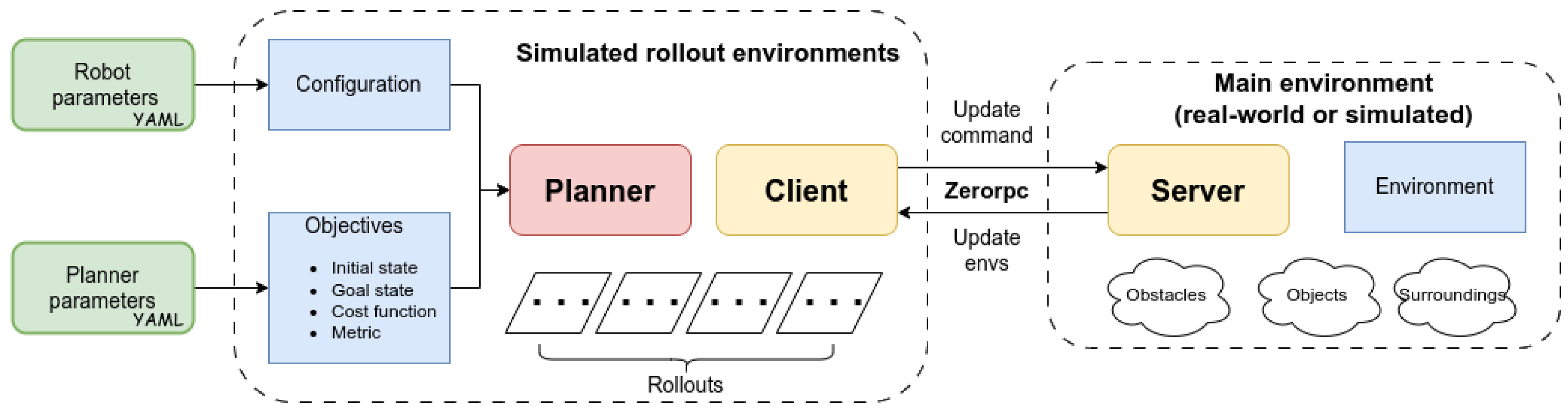
Algorithm 1 Sampling-based MPC

Given:
F, g: Dynamics and clamping function
K: Set of sampled environment ids
T: Number of timesteps
U: Initial control sequence
ψ_t: Parameters of policy at time t
c: Cost functions
while task not completed **do**
  $x_t \leftarrow$ GetState()
  /* *Begin parallel sampling* */
  for i in K **do**
    $\mathbf{v} \leftarrow$ SampleControls(U, T)
    $\mathbf{c} \leftarrow$ ComputeRolloutCosts(v, F, c, ψ_t)
    $U \leftarrow$ UpdateDistribution($\mathbf{v}$, $\mathbf{c}$)
  end for
  /* *End parallel sampling* */
  u_t = ComputeNextCommand(U)
  ExecuteCommand(u_t)
  ShiftCommand($\mathbf{u}$)
end while

4. Software Implementation

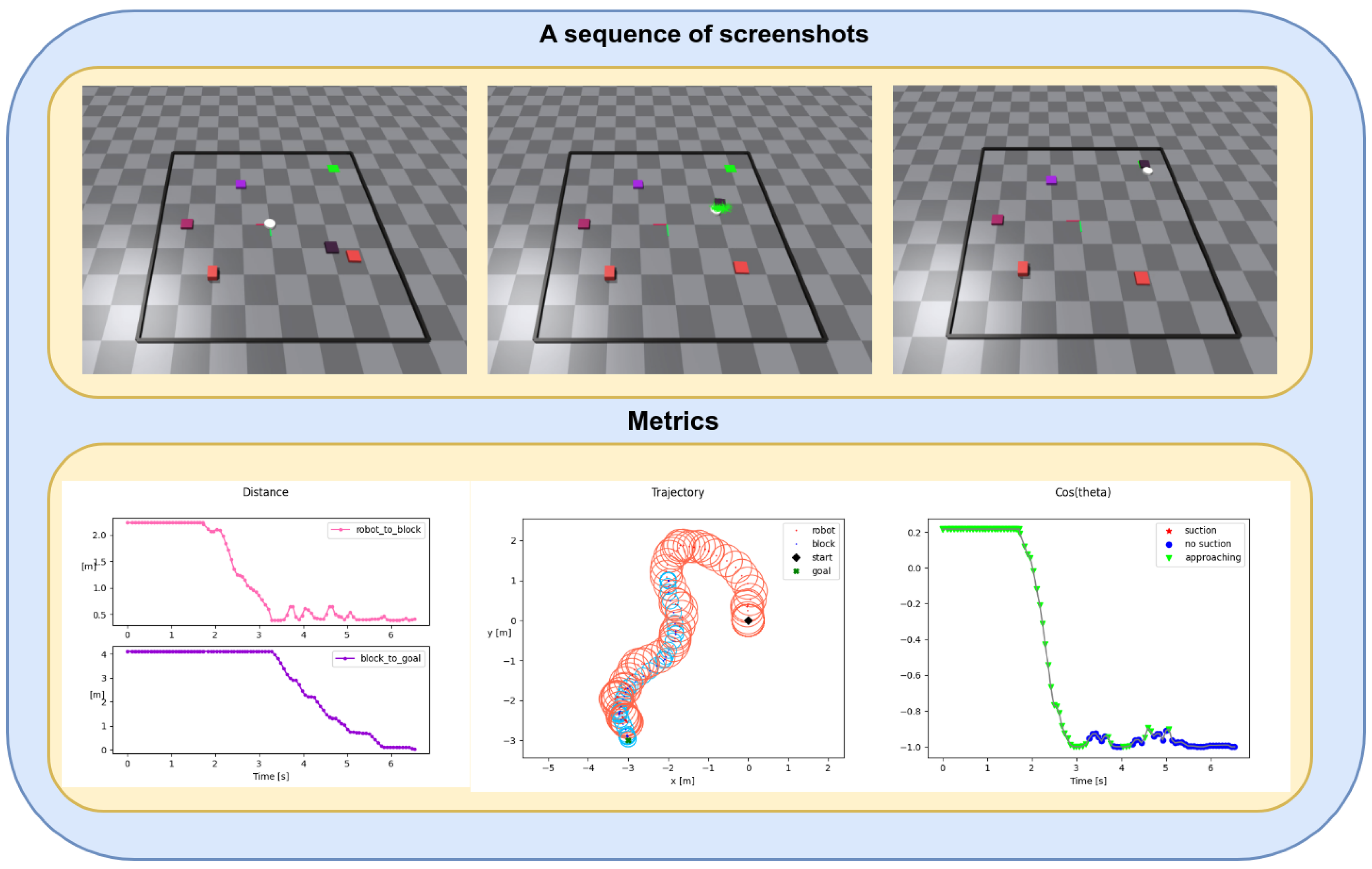
The code structure of this work is split up into two files as can be seen below. The **server file** mainly reflects the setting in the real-world environment, while the **client file** mainly simulates the planning sequences in the rollout environments. You only need to set up the following files:

- **Configuration files** for robot and planner parameters.
- **An objective function** to describe the task.
- **A world file** that includes obstacles, objects and surroundings.

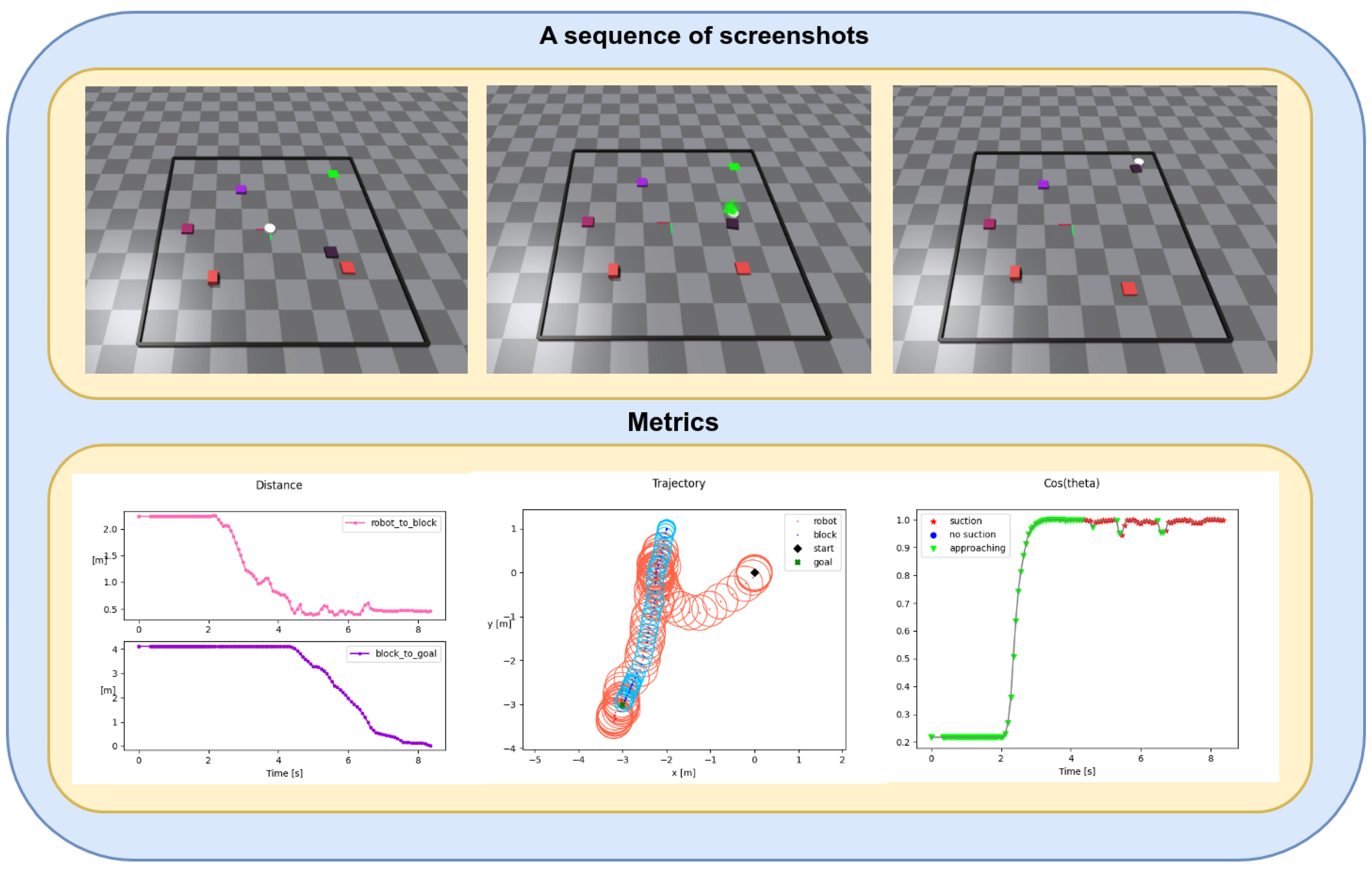


5. Experiments

Push



Pull



SCAN TO SEE THE VIDEOS

7. References

[1] Moses Bangura and Robert Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.

[2] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[3] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759. PMLR, 2022.