CS686 MPA 1st Paper Presentation

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning (IROS 2022)

2023.10.25.

1st year M.S. Student @KAST, SGVR Lab

CHANMI LEE

KAIST

Contents

Overview

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

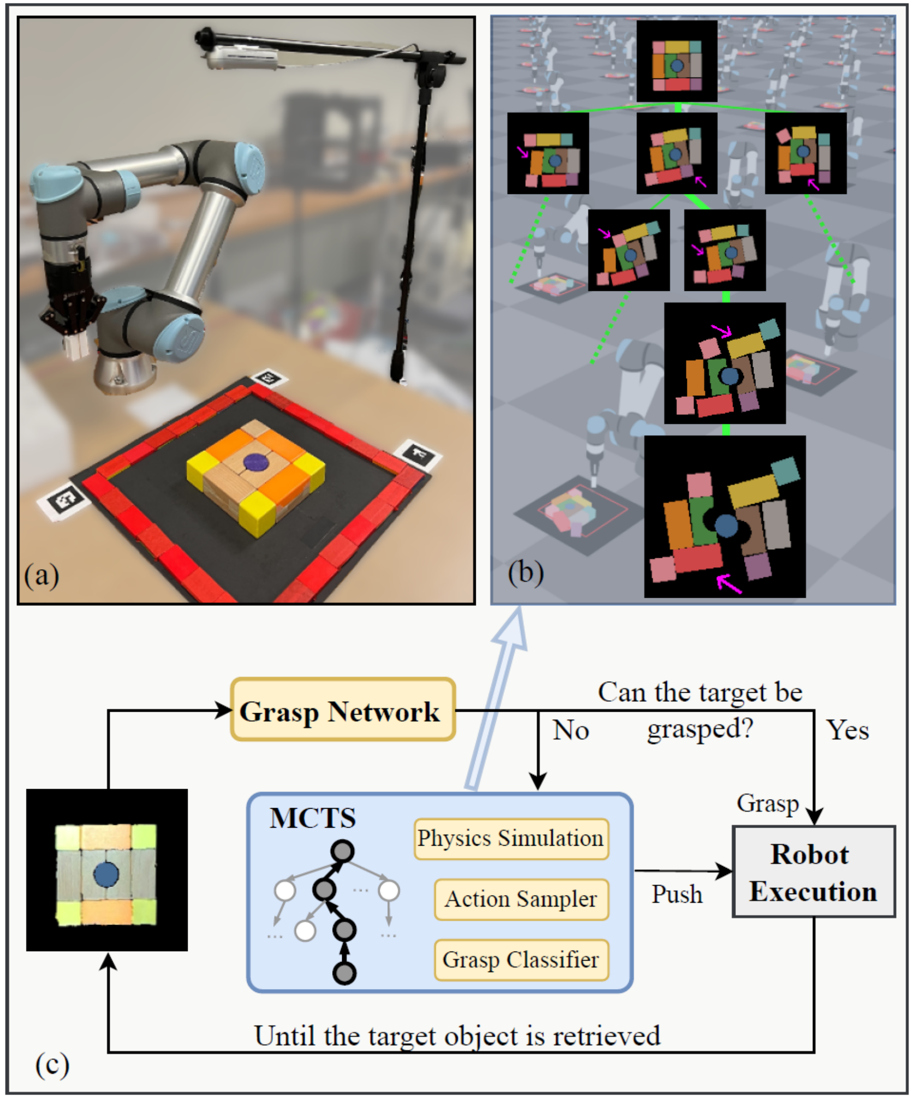
Purpose

- For Long-Horizon, Episodic Robotic Planning Task
  - Accelerating
  - Improve solution quality

Main Idea

- Use Parallel Monte Carlo Tree Search
- Use Batched Simulation

➔ PMBS

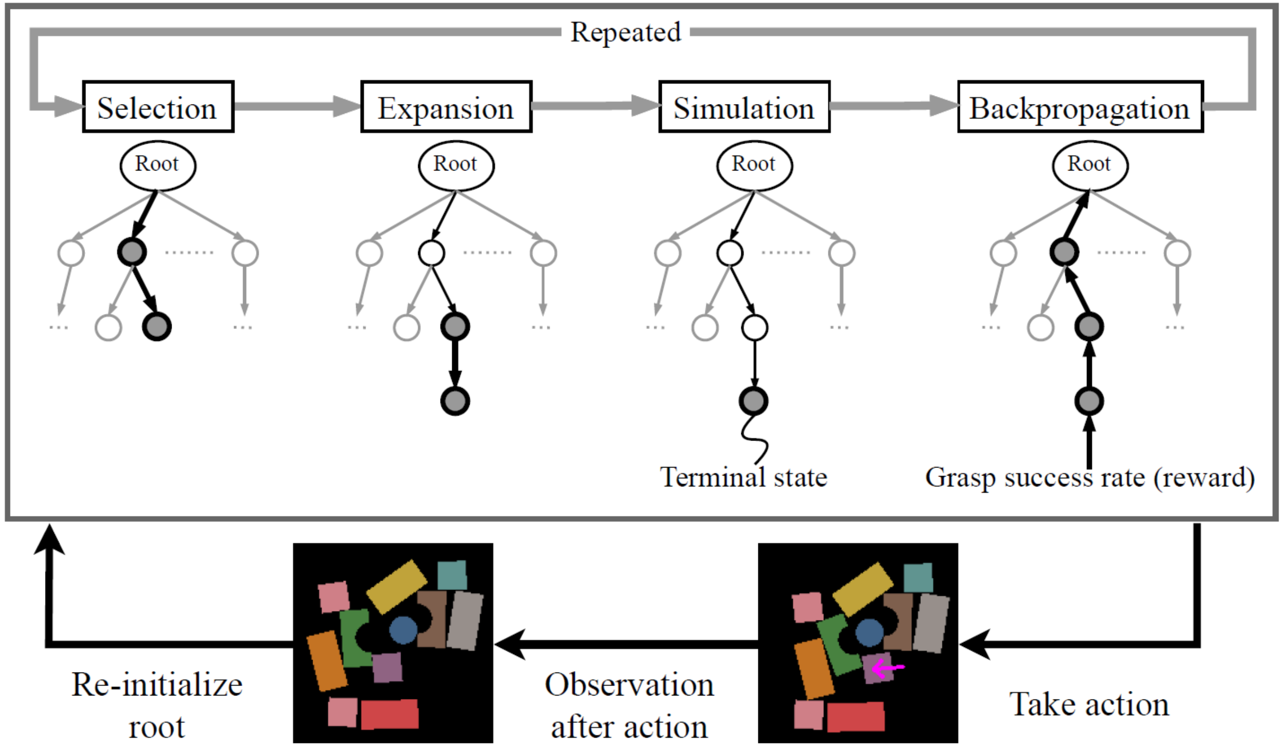


Background

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Monte Carlo Tree Search (MCTS)

- MCTS
  - Heuristic search algorithm for decision processes
  - Build a search tree
  - Balancing exploration and exploitation
  - Iteratively performing 4 stages operations
    - Selection
    - Expansion
    - Simulation
    - Back propagation

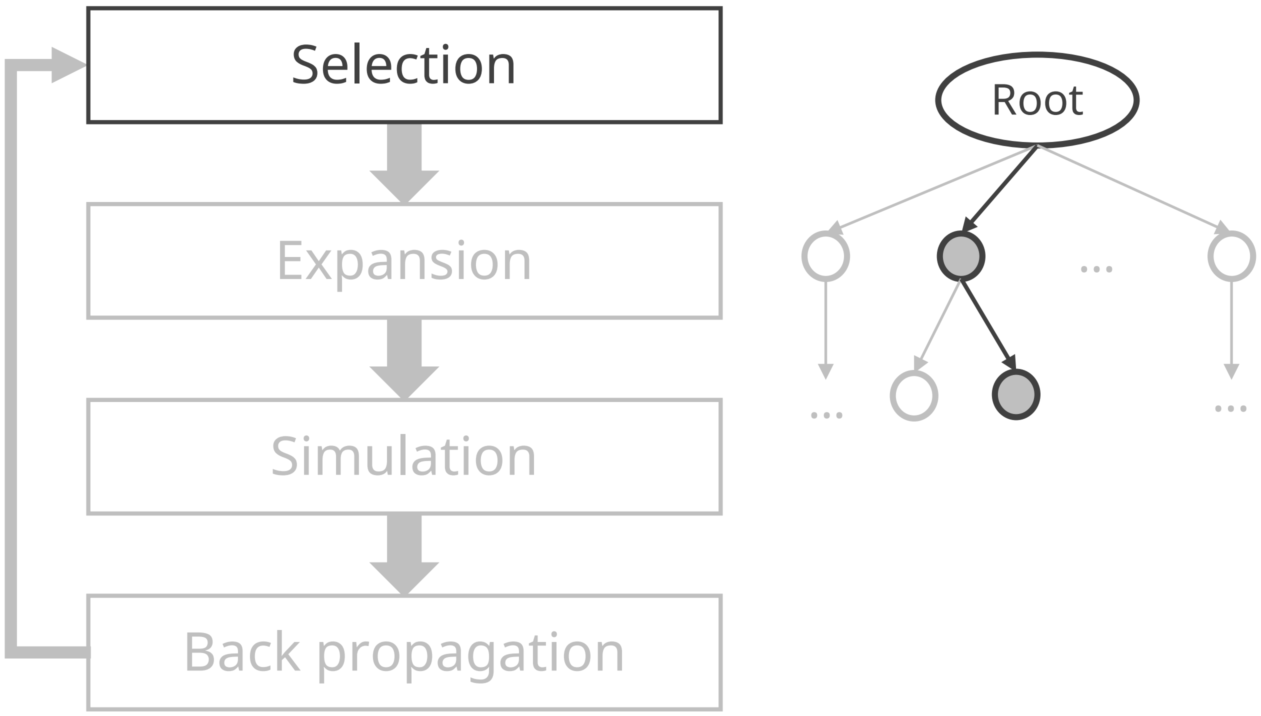


Background

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Monte Carlo Tree Search (MCTS)

- Iterate 4 stages



- Selection
  - Select best node to grow tree
  - Until terminal node or non-children node
  - Criterion : Upper Confidence Bound (UCB)

$$\underset{n' \in children\ of\ n}{\text{argmax}} \frac{Q(n')}{N(n')} + c\sqrt{\frac{2\ln(N(n))}{N(n')}}$$

$Q(n)$: sum of rewards for node n

$N(n)$: number of times n was selected so far

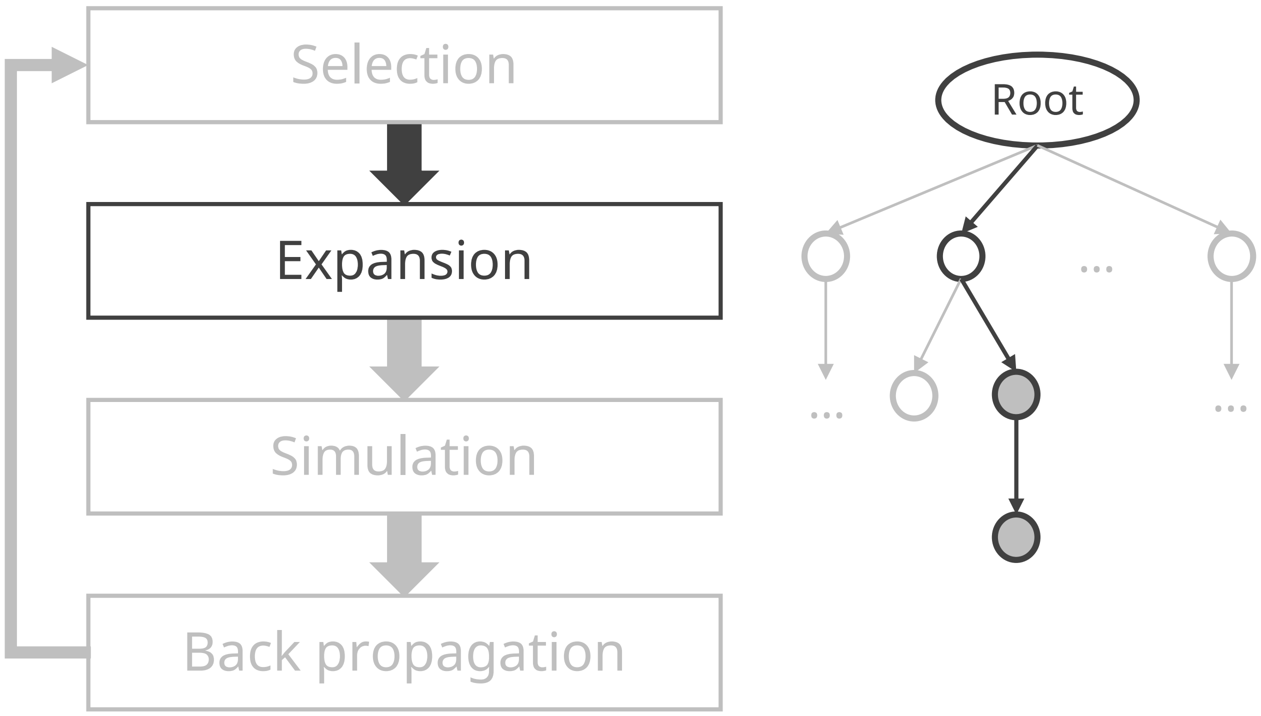
n' : child node

Background

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Monte Carlo Tree Search (MCTS)

- Iterate 4 stages



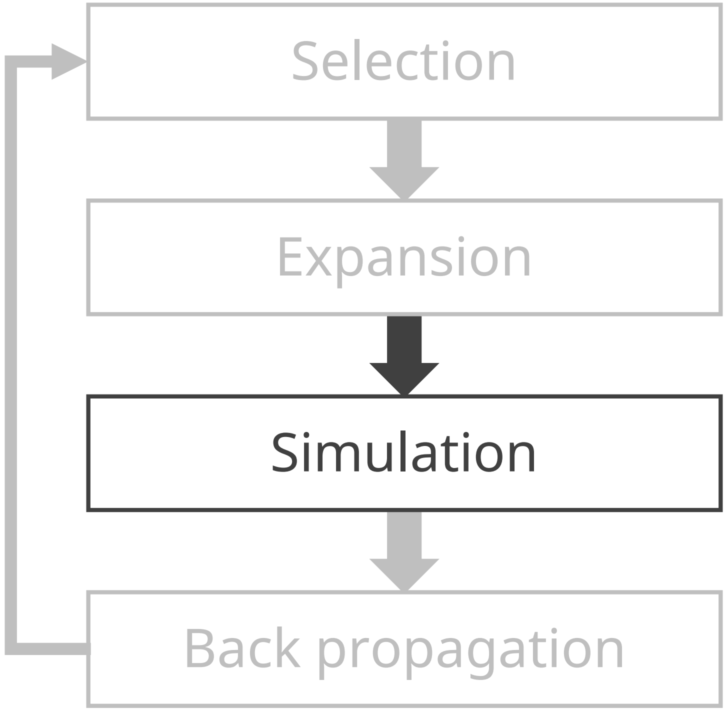
- Expansion
  - Add new child node n’ to the tree
  - (If it is not a terminal node)

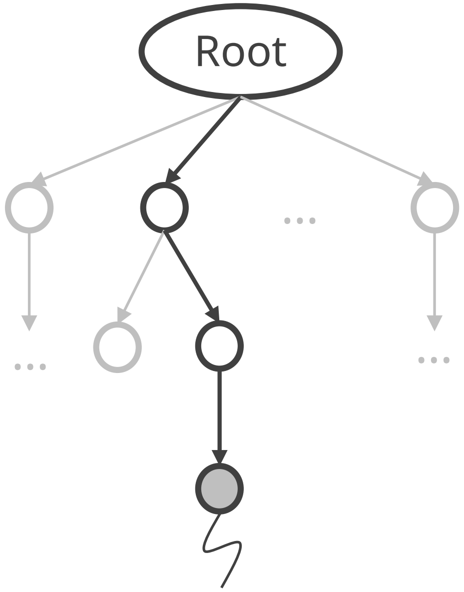
Background

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Monte Carlo Tree Search (MCTS)

- Iterate 4 stages





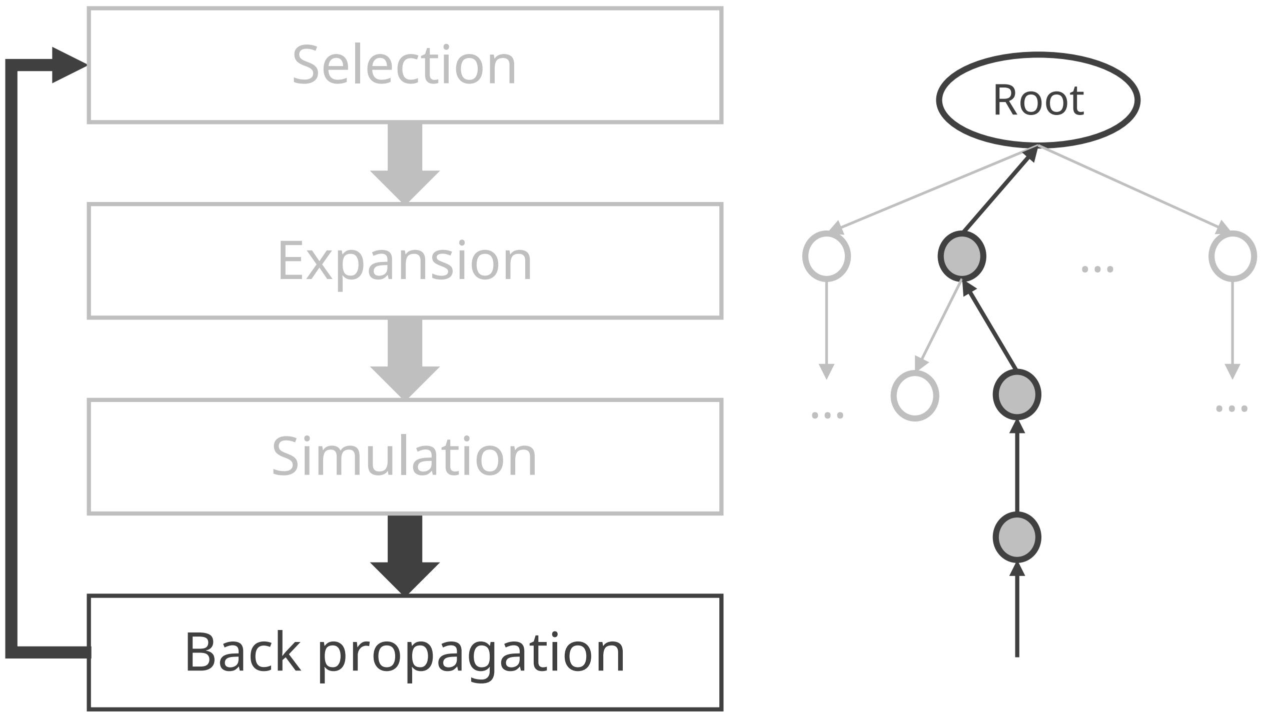
- Simulation
  - Carried out at new child node n'
  - Repeat select-expand-simulate until reached a terminal state
  - Terminal state → yield reward

Background

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Monte Carlo Tree Search (MCTS)

- Iterate 4 stages



- Back propagation
  - Obtained terminal reward
  - Propagate back from n' to root node
  - For all nodes along path,

    ➔ update sum of reward $Q(n)$

    ➔ increment the number of visit $\mathrm{N}(n)$

Background

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

GPU-Based Physics simulators

- Leverage the power of GPU
  → Compute physics simulations
- Can perform many calculations simultaneously

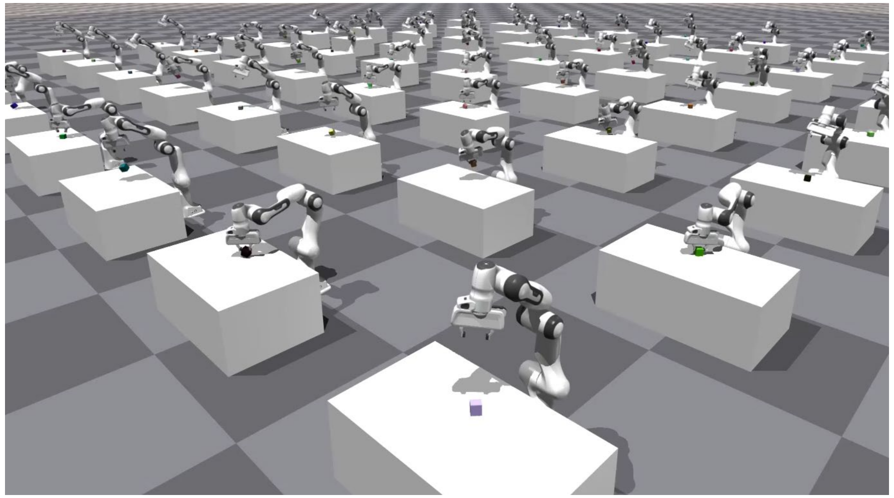
ISAAC

[1]

[2]

Batched Rigid-body Simulation

- Run multiple rigid-body simulations simultaneously in batched manner
- Speed up with better quality solution



[3]

Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Parallel MCTS with Batched Simulation

- Preliminary
  - Object-retrieval-from-clutter task ~ MDP $< S, A, T, O, P >$
    - State space (S)
      - $\mathrm{S}_t = (robot_t,\ obj_t^1,\ obj_t^2,\ \ldots\ ,\ obj_t^n)$
      - $robot_t$: joint angles vector for time t
      - obj_t^i : pose & geometry vector for time t
    - Action space (A)
      - $\mathrm{A} = \{A^p, A^g\}$
      - $a^p = (x_s, y_s, x_g, y_g) \in A^p$: push action
      - $a^p = (x,\ y,\ z,\ \theta) \in A^g$: grasp action
    - Transition function (T)
      - $T(a_t) : s_t \rightarrow s_{t+1}$
    - Reward function (R)
      - $r = \{0, 1\}$
      - $r = 1$: target grasp, $r = 0$: don't grasp
    - Observation space (O)
      - $o_t \in O^p$: top-down RGB-D images

Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

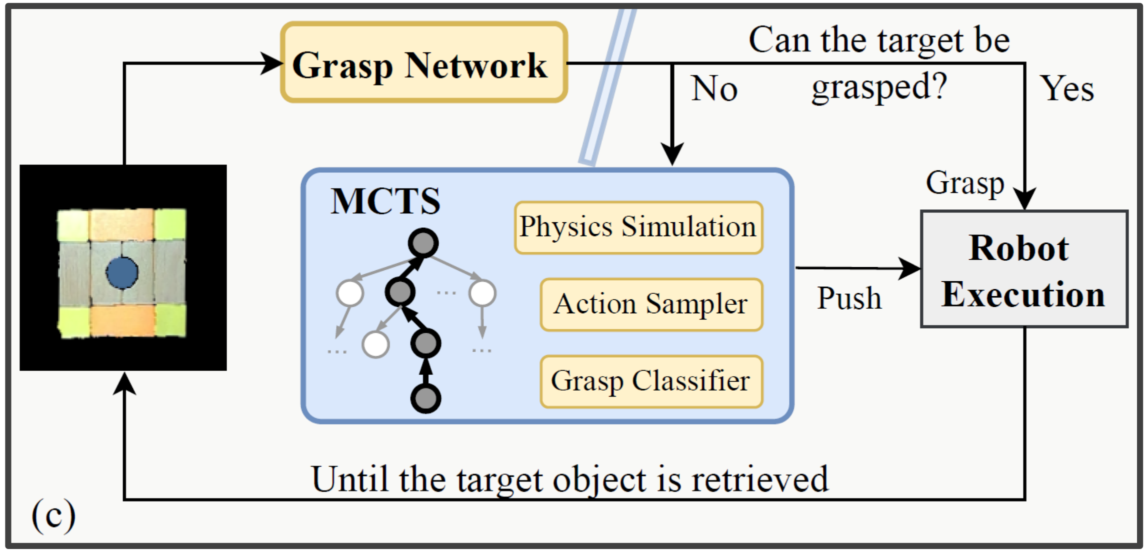
Parallel MCTS with Batched Simulation

- **Algorithm 1:** Parallel MCTS with Batched Simulation

Function `Main` (s_t, o_t)
  while there is a target object in workspace **do**
    if the target object can be grasped (query GN) **then**
      Execute grasp the target object
    else Execute `Parallel-MCTS` (s_t) `// Push`

Function `Parallel-MCTS` (s)
  Create root node n_0 with state s
  es_level ← 1 `// Early stop level`
  graspable_nodes ← ∅
  while (within time budget) and (depths of all graspable_nodes are greater than es_level) **do**
    $[(n^1, a^1), \dots, (n^{N_e}, a^{N_e})] \leftarrow$ `Selection`(n_0)
    Reset all $\hat{N}(n)$ to 0
    $[n'^1, \dots, n'^{N_e}] \leftarrow$ `Expansion`$([(n^1, a^1), \dots, (n^{N_e}, a^{N_e})])$
    for n' in $[n'^1, \dots, n'^{N_e}]$ **do**
      if $GC(n'(o)) > R_c^*$ **then**
        graspable_nodes ← graspable_nodes $\cup \{n'\}$
    if all nodes at es_level − 1 are fully expanded or terminal **then**
      es_level ← es_level + 1
    $[r^1, \dots, r^{N_e}] \leftarrow$ `Simulation`$([n'_1, \dots, n'_{N_e}])$
    `Backpropagation` $([(n'^1, r^1), \dots, (n'^{N_e}, r^{N_e})])$
  return the a^p that leads to best child node of root, ranked by Eq. 2

- Main Function
  - Grasp object or do parallel-MCTS to push objects



  - Grasp Network : provide grasp action
  - Action Sampler : sampling actions
  - Grasp Classifier : grasp probability → grasp ability

Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding u ... zo

Parallel MCTS with Batched Simulation

- **Algorithm 1:** Parallel MCTS with Batched Simulation

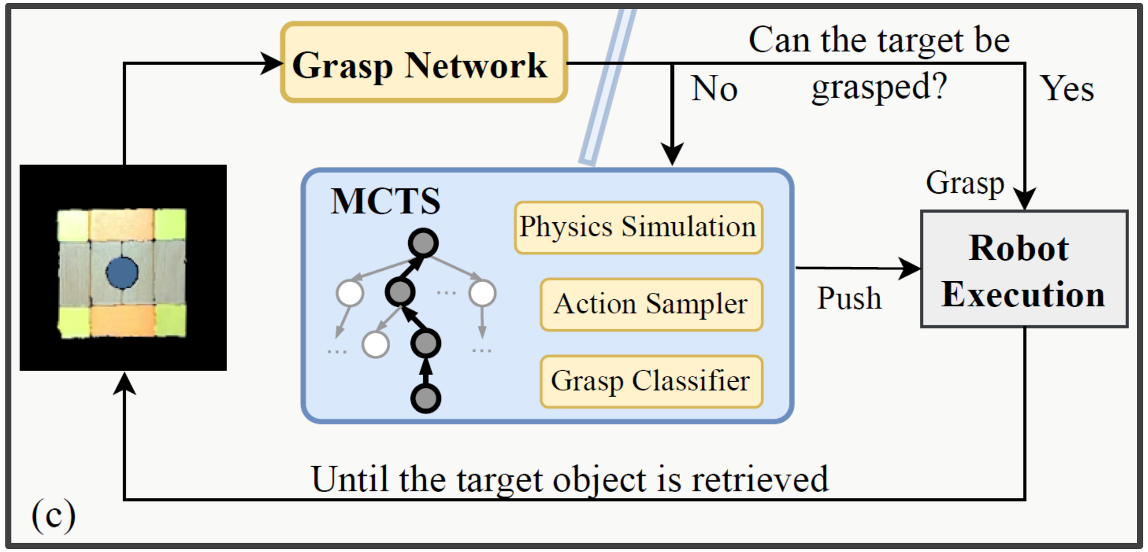
  Function `Main` (s_t, o_t)
    while there is a target object in workspace **do**
      if the target object can be grasped (query GN) **then**
        Execute grasp the target object
      else Execute `Parallel-MCTS` (s_t) `// Push`

  Function `Parallel-MCTS` (s)
    Create root node n_0 with state s
    es_level $\leftarrow 1$ `// Early stop level`
    graspable_nodes $\leftarrow \varnothing$
    while (within time budget) and (depths of all graspable_nodes are greater than es_level) **do**
      $[(n^1, a^1), \ldots, (n^{N_e}, a^{N_e})] \leftarrow$ `Selection` (n_0)
      Reset all $\hat{N}(n)$ to 0
      $[n'^1, \ldots, n'^{N_e}] \leftarrow$ `Expansion` $([(n^1, a^1), \ldots, (n^{N_e}, a^{N_e})])$
      for n' in $[n'^1, \ldots, n'^{N_e}]$ **do**
        if $GC(n'(o)) > R^*_c$ **then**
          graspable_nodes $\leftarrow$ graspable_nodes $\cup \{n'\}$
      if all nodes at es_level $-$ 1 are fully expanded or terminal **then**
        es_level $\leftarrow$ es_level $+ 1$
      $[r^1, \ldots, r^{N_e}] \leftarrow$ `Simulation` $([n'_1, \ldots, n'_{N_e}])$
      `Backpropagation` $([(n'^1, r^1), \ldots, (n'^{N_e}, r^{N_e})])$
      return the a^p that leads to best child node of root, ranked by Eq. 2

Fig. 3: Sampled push actions.

Grasp probability: 0.57   Grasp probability: 0.99

- Main Function
  - Grasp object or do parallel-MCTS to push objects



  - Grasp Network : provide grasp action
  - Action Sampler : sampling actions
  - Grasp Classifier : grasp probability → grasp ability

Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

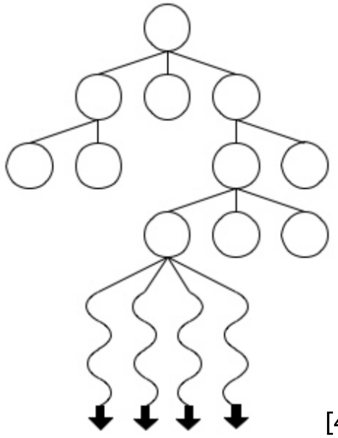
Parallel MCTS with Batched Simulation

- **Algorithm 1:** Parallel MCTS with Batched Simulation

Function Main (s_t, o_t)
  while there is a target object in workspace **do**
    if the target object can be grasped (query GN) **then**
      Execute grasp the target object
    else Execute $\boxed{\texttt{Parallel-MCTS}(s_t)}$ // Push

Function Parallel-MCTS (s)
  Create root node n_0 with state s
  es_level $\leftarrow 1$ // Early stop level
  graspable_nodes $\leftarrow \varnothing$
  while (within time budget) and (depths of all graspable_nodes are greater than es_level) **do**
    $[(n^1, a^1), \ldots, (n^{N_e}, a^{N_e})] \leftarrow \boxed{\texttt{Selection}(n_0)}$
    Reset all $\hat{N}(n)$ to 0
    $\boxed{[n'^1, \ldots, n'^{N_e}] \leftarrow \texttt{Expansion}([(n^1, a^1), \ldots, (n^{N_e}, a^{N_e})])}$
    for n' in $[n'^1, \ldots, n'^{N_e}]$ **do**
      if $GC(n'(o)) > R_c^*$ **then**
        graspable_nodes $\leftarrow$ graspable_nodes $\cup \{n'\}$
    if all nodes at es_level $- 1$ are fully expanded or terminal **then**
      es_level $\leftarrow$ es_level $+ 1$
    $[r^1, \ldots, r^{N_e}] \leftarrow \boxed{\texttt{Simulation}([n'_1, \ldots, n'_{N_e}])}$
    $\boxed{\texttt{Backpropagation}([(n'^1, r^1), \ldots, (n'^{N_e}, r^{N_e})])}$
  return the a^p that leads to best child node of root, ranked by Eq. 2

Leaf Parallelization



[4]

- Parallel-MCTS
  - Exploit parallelism within MCTS
    ➔ Improve performance
  - Use leaf parallelism
    - Select one leaf node n'
    - Multiple processors run simulation from n'
  - Get push action with 4 steps
    - Selection / Batch Parallel Expansion / Batch Parallel Simulation / Back propagation

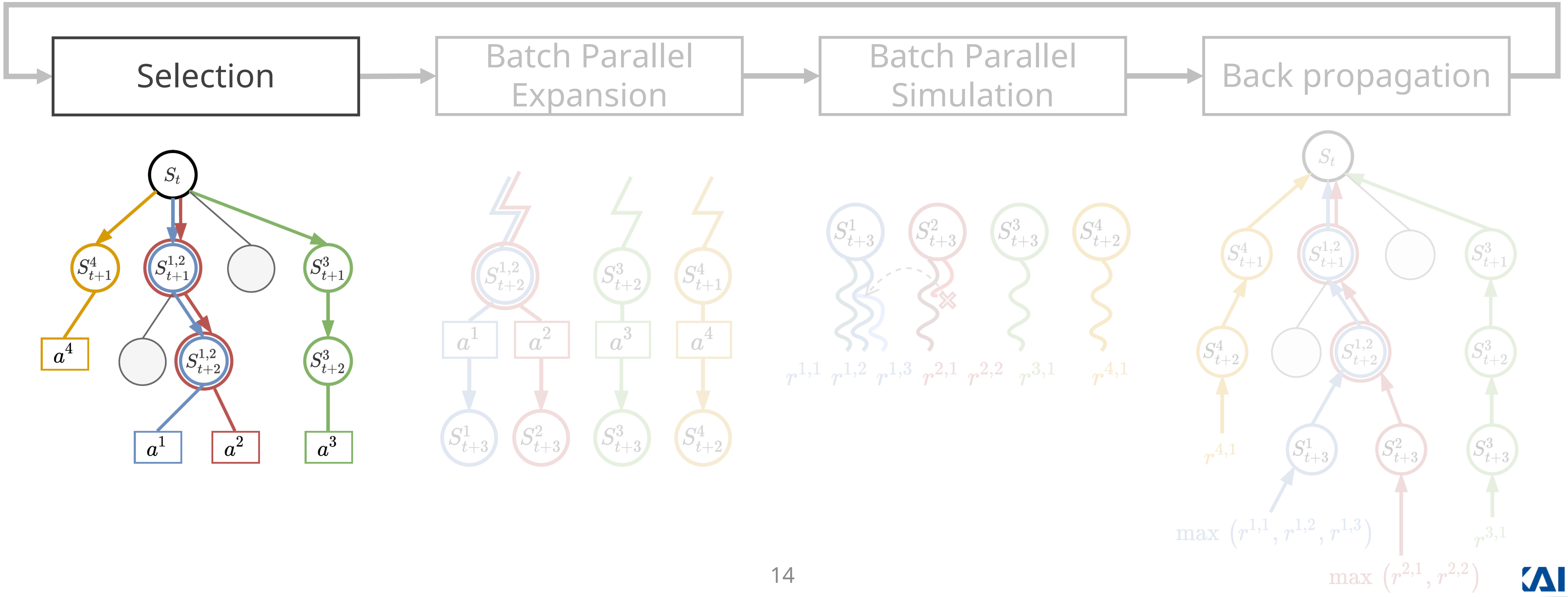
KAIST

Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Parallel MCTS with Batched Simulation

- Parallel-MCTS



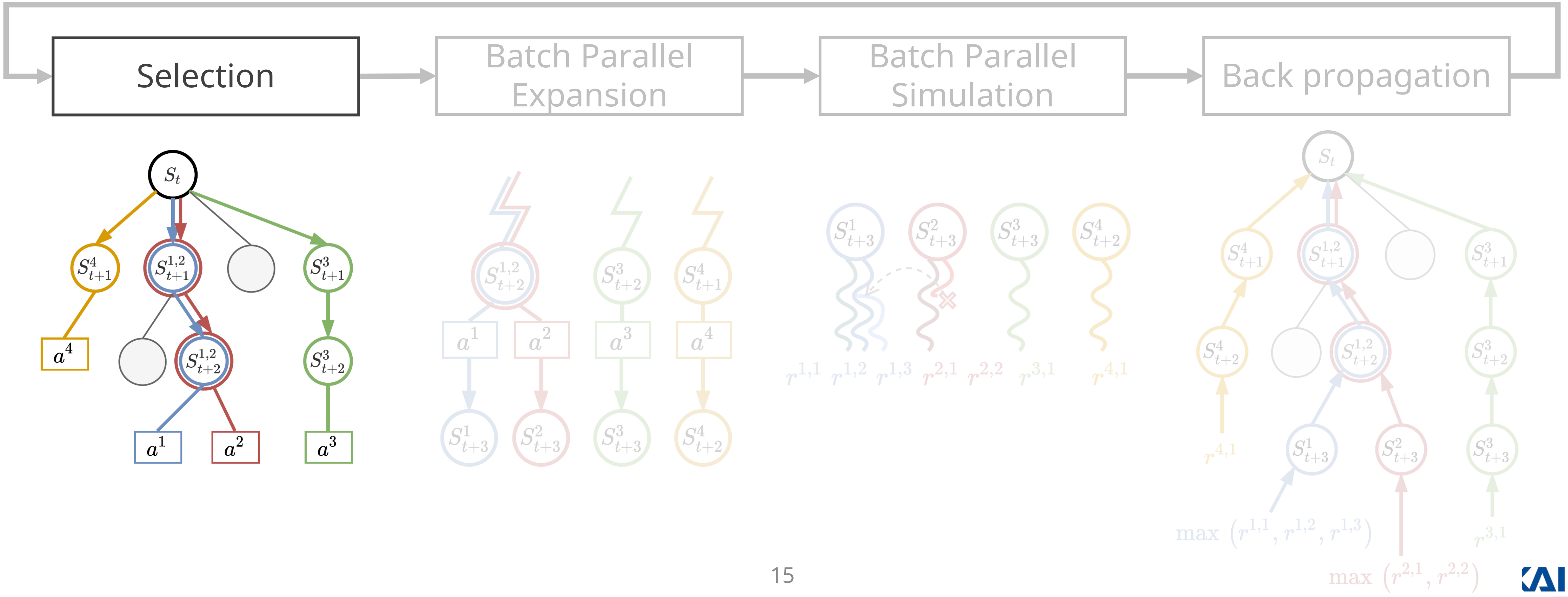
Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Parallel MCTS with Batched Simulation

$$\underset{n' \in children\ of\ n}{\operatorname{argmax}} \frac{Q(n')}{N(n') + \widehat{N}(n')} + c\sqrt{\frac{2\ln(N(n) + \widehat{N}(n))}{N(n') + \widehat{N}(n')}}$$

- Parallel-MCTS

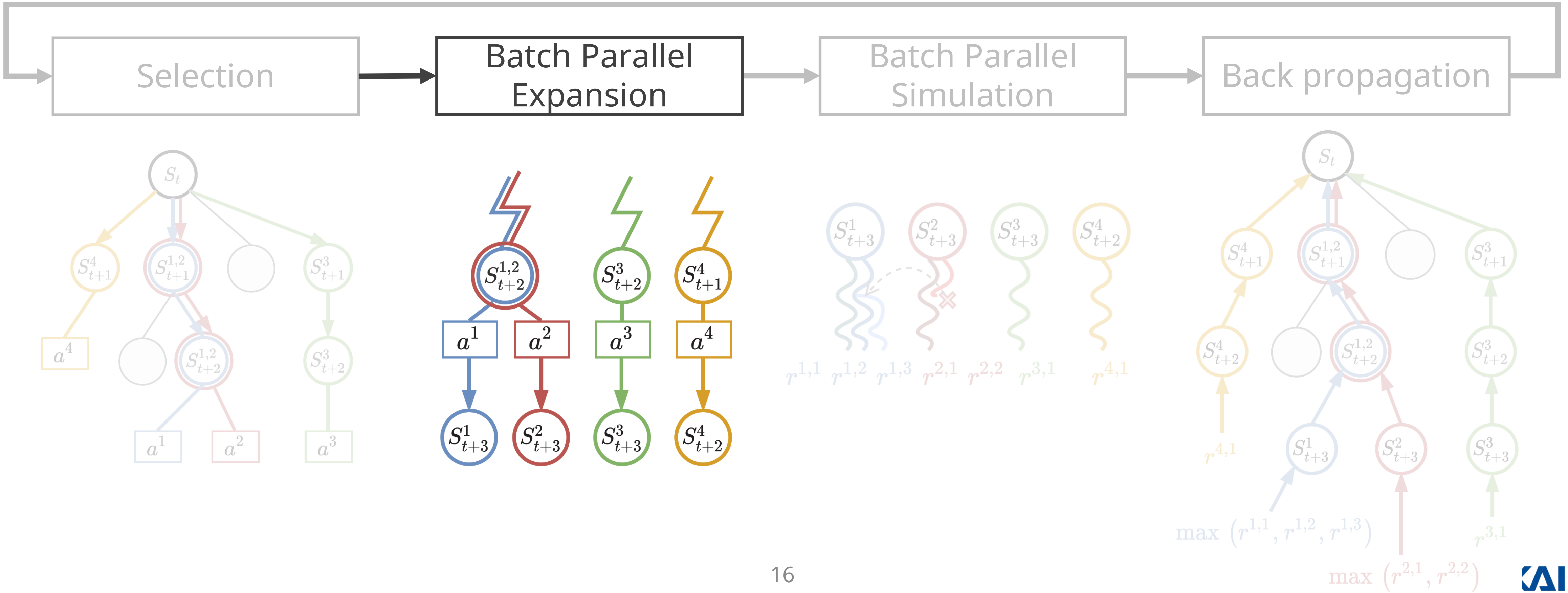


Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Parallel MCTS with Batched Simulation

- Parallel-MCTS

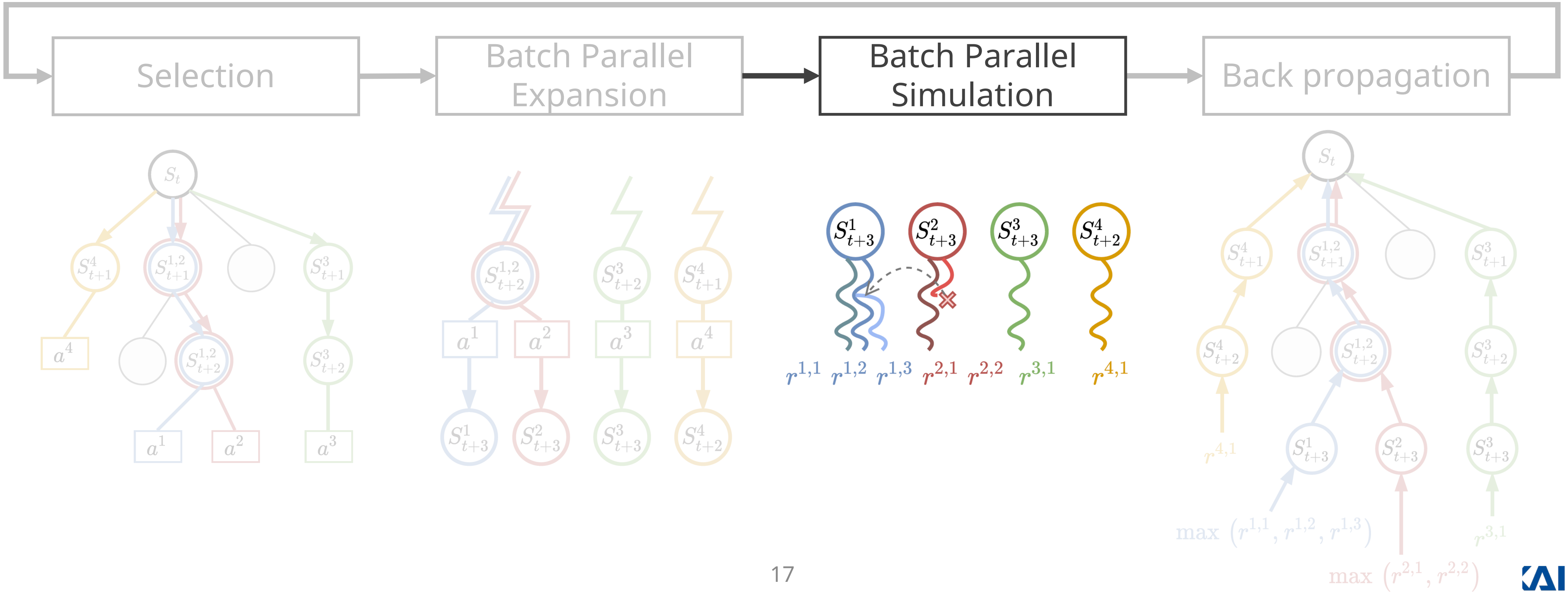


Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Parallel MCTS with Batched Simulation

- Parallel-MCTS

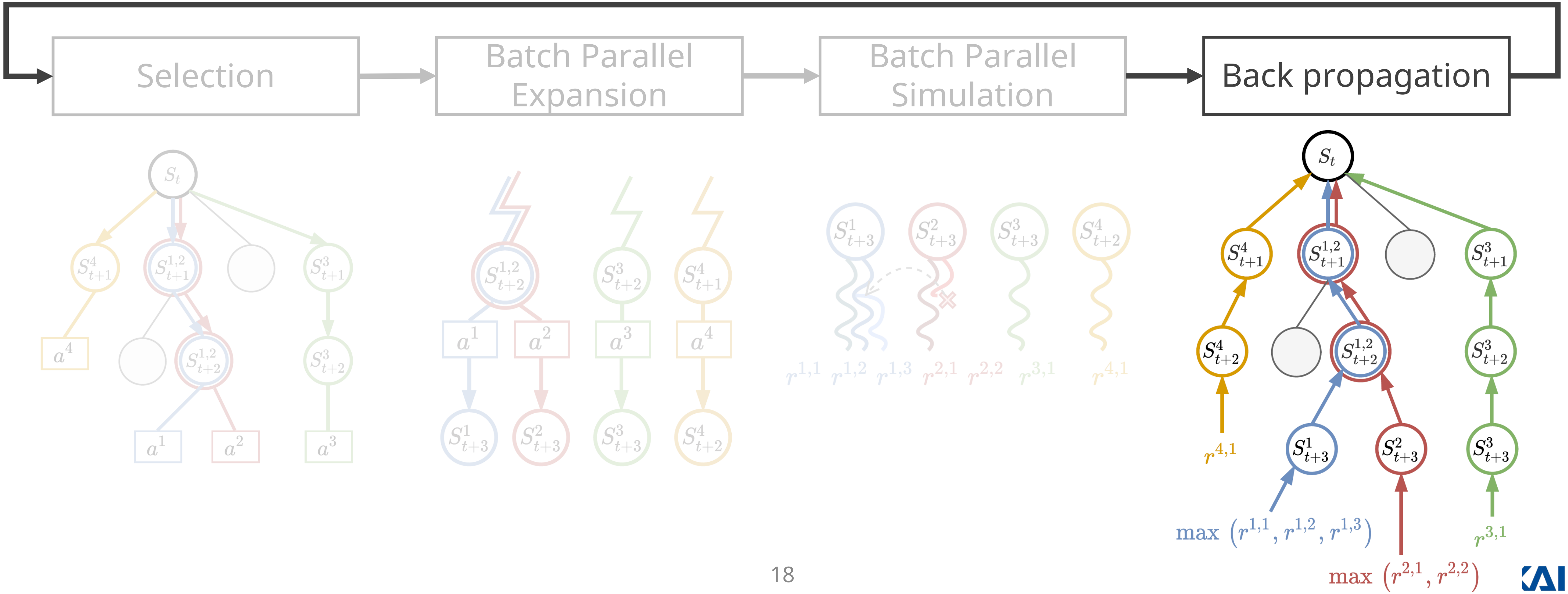


Methodology

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Parallel MCTS with Batched Simulation

- Parallel-MCTS



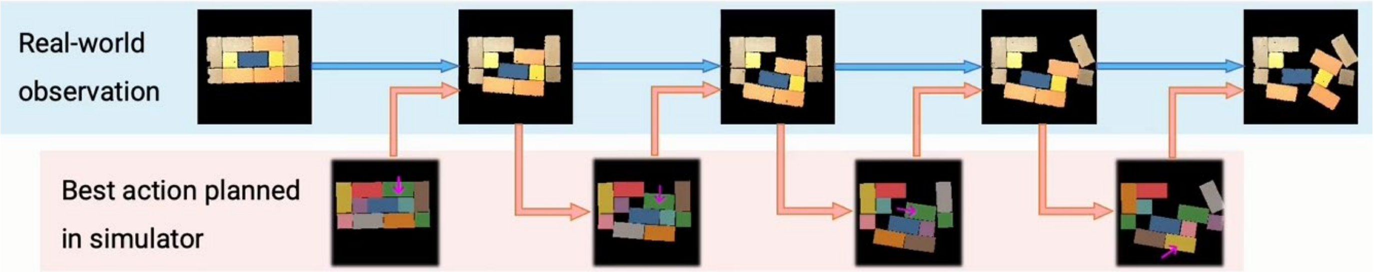
Experimental Results

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Experiments

- Environment Setting
  - Simulator : Isaac Gym
  - Robot : UR5e with 2G-Gripper
  - Camera : Intel RealSense D455
  - Evaluator
    - Nvidia RTX 2080Ti GPU
    - Intel i7-9700K CPU
    - 32GB memory

GPU-based simulator
(Issac Gym)



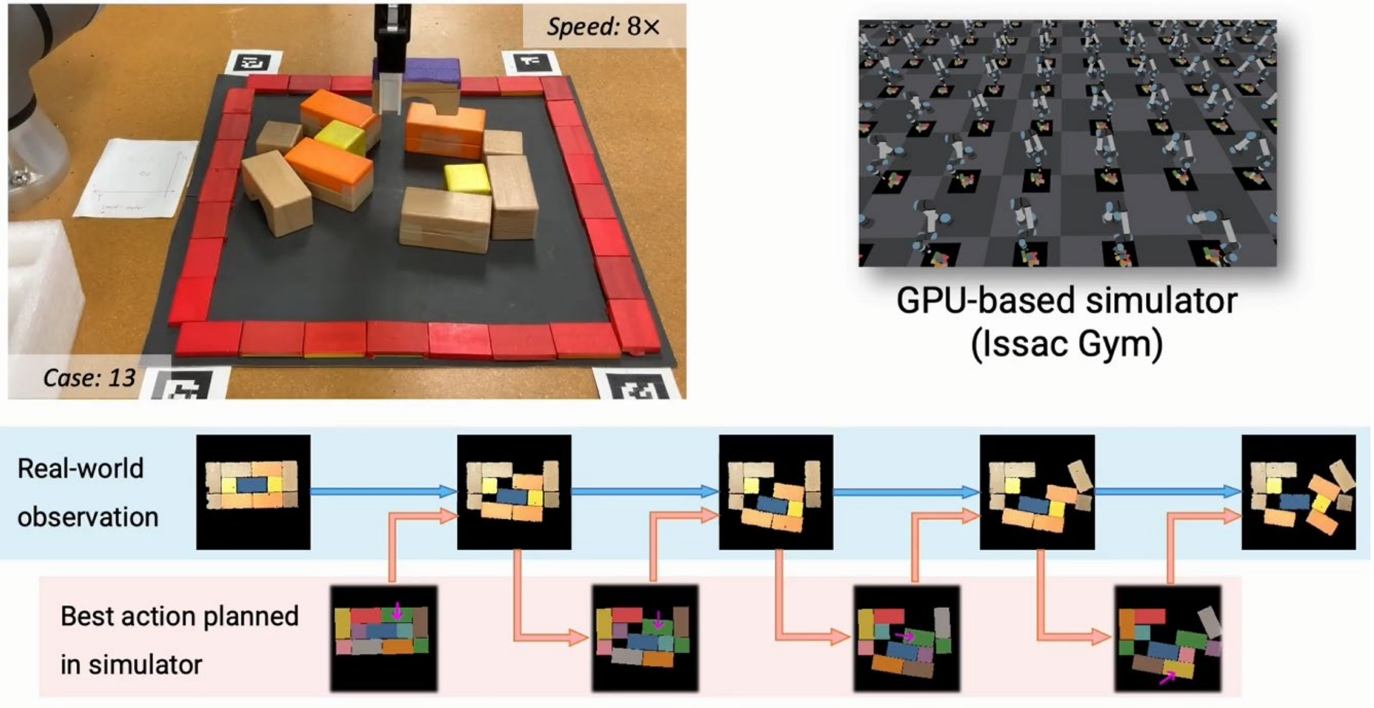
Massive parallel planning in simulation and executing the best action on the real robot

Experimental Results

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Experiments

- Environment Setting
  - Hyperparameter
    - Discount factor : $\gamma = 0.8$
    - Max depth : $d_T = 0.8$
    - Simulation depth : $d_s = 0.8$
    - Threshold of GC : $R_c^* = 0.8$
    - UCB exploration term : $c = 0.3$
    - Time limit (budget) : $T_{max} = 60\ sec$
    - Number of robot (envs) : 1000



Massive parallel planning in simulation and executing the best action on the real robot

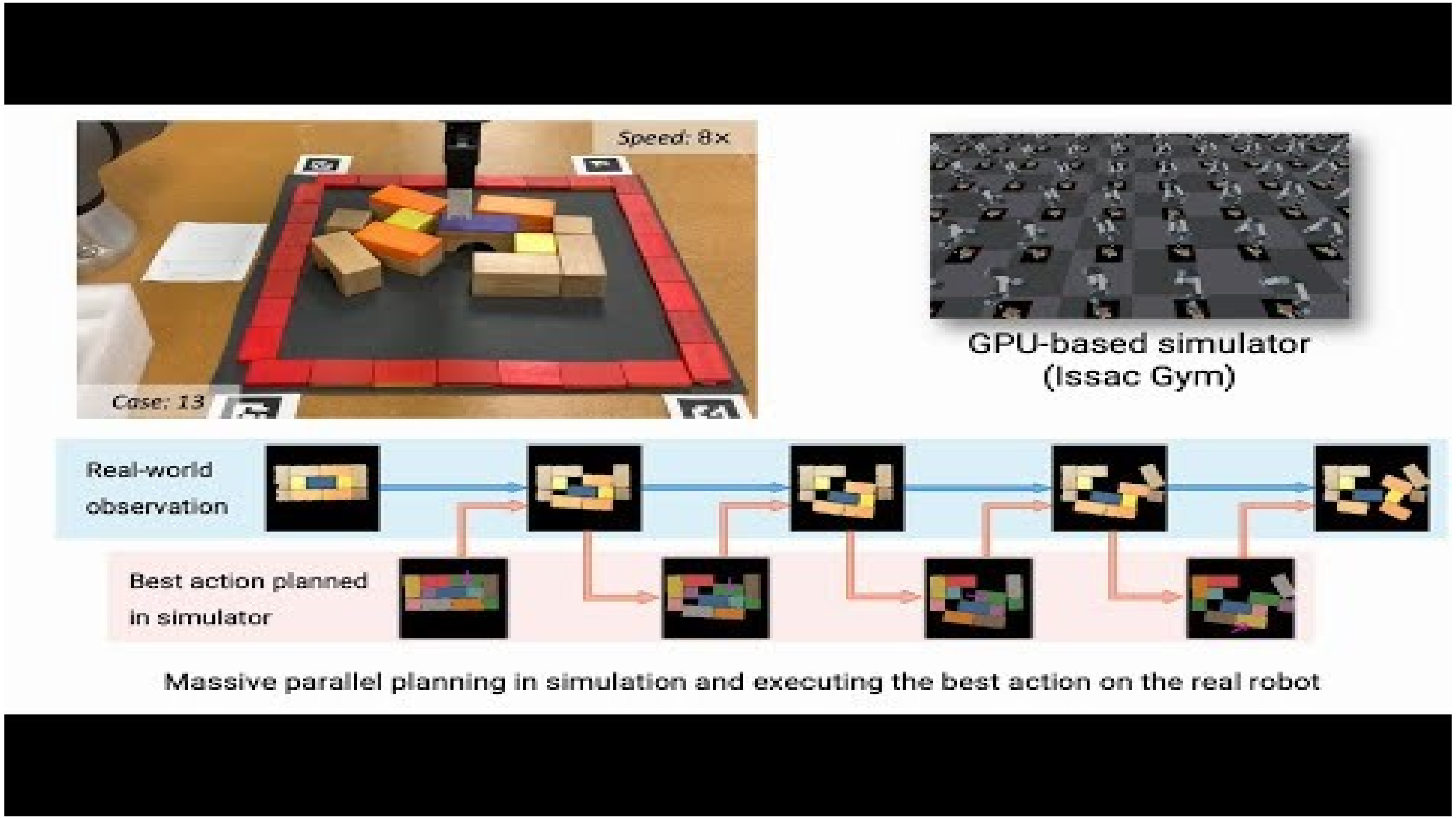
Experimental Results

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Results

Link : https://www.youtube.com/watch?v=-Br2IBjArgY

- (6’01” ~)



Experimental Results

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Results

- PMBS vs. MCTS (with time-budge limited up to 60 sec)

	Num. of Actions	Time	Completion	Grasp Success
PMBS-60	**5.72**	**73**s	100%	100%
MCTS-60	10.45	529s	83.3%	87.0%
PMBS-60 (sim)	5.03	81s	100%	97.2%
MCTS-60 (sim)	10.77	587s	76.7%	96.6%

  - Number of Actions : PMBS < MCTS
  - Time : PMBS < MCTS
  - Completion : PMBS > MCTS
  - Grasp Success : PMBS > MCTS

Summary

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Conclusion & Discussion

- Propose PMBS (Parallel Monte Carlo tree search with GPU-enabled batched simulations)
- Strengths
  - Accelerating long-horizon, episodic robotic planning task
  - Over 30x speed up compared to serial MTCS
  - Better solution quality compared to serial MTCS
  - Achieve near real-time planning performance in solving complex, long-horizon episodic tasks
- Weaknesses
  - Not enough to be real time evaluation
  - Evaluated for only 20 cases (environments) ➔ not sure for random environment

Reference

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

[1] https://www.google.com/search?q=isaac+sim&sca_esv=576128351&tbm=isch&sxsrf=AM9HkKmNdzW7GkG4AnR6NBhCGpjpmuNQTw:1698160616992&source=lnms&sa=X&ved=2ahUKEwjGoOOi_Y6CAxXNhVYBHRTQCOgQ_AUoAXoECAMQAw&biw=1080&bih=1771&dpr=1#imgrc=nRQtBGSnUK-FGM

[2] https://pypi.org/project/brax/

[3] https://developer.nvidia.com/isaac-gym

[4] https://dke.maastrichtuniversity.nl/m.winands/documents/multithreadedMCTS2.pdf

Thank You

Quiz

Parallel Monte Carlo Tree Search with Batched Rigid-body Simulations for Speeding up Long-Horizon Episodic Robot Planning

Q1) What is not related to PMBS? (PMBS : Parallel Monte Carlo Tree Search)

A. Selection

B. Batched Parallel Simulation

C. Batched Parallel Expansion

D. Quick Sort

Q2) In this paper, which kind of parallel MCTS was used?

A. Leaf Parallelization

B. Root Parallelization

C. Tree Parallelization with global mutexes

D. Tree parallelization with local mutexes