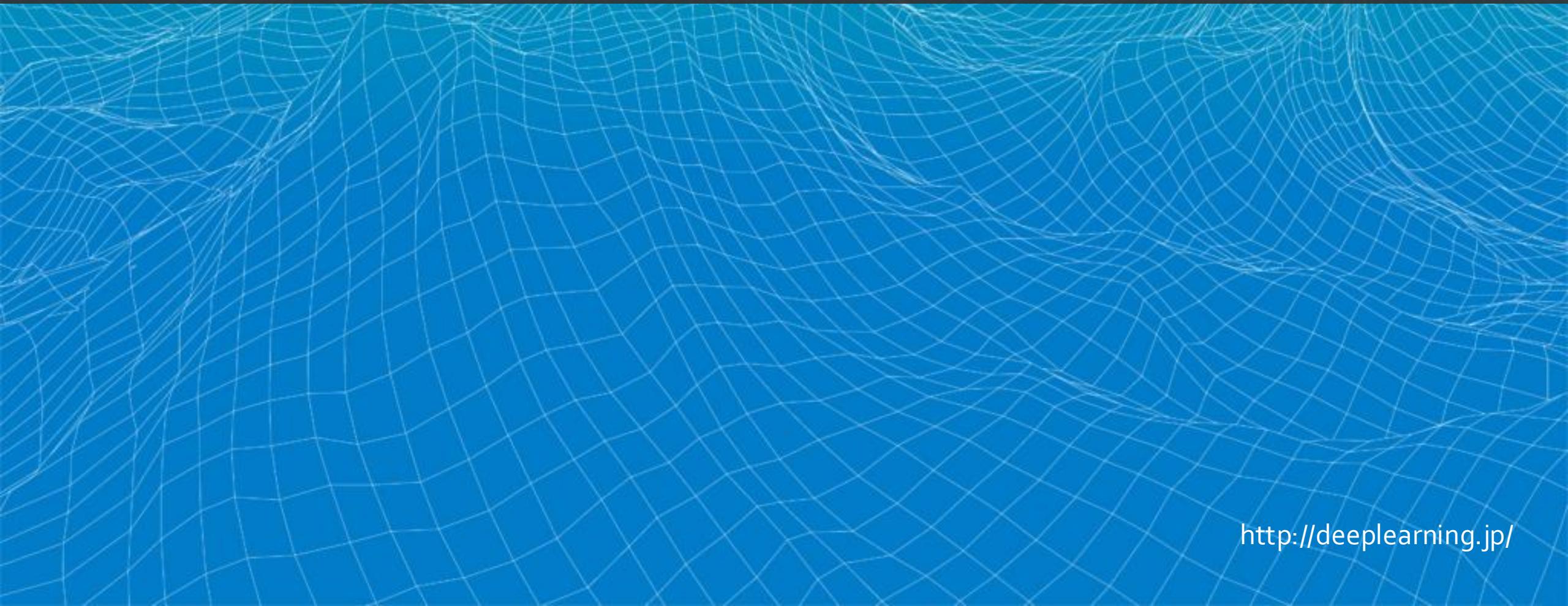
DEEP LEARNING JP
[DL Papers]

“MANISKILL3: GPU PARALLELIZED ROBOTICS SIMULATION AND RENDERING FOR GENERALIZABLE EMBODIED AI ”

Presenter: Satoi Yamaguchi



http://deeplearning.jp/

自己紹介

- 山口慧（Satoi Yamaguchi）

- 早稲田修士一年
- 松尾・岩澤研LLMATCH研究員

- 研究内容
  - LLMを使ったロボット制御

論文情報

論文誌	“MANISKILL3: GPU PARALLELIZED ROBOTICS SIMULATION AND RENDERING FOR GENERALIZABLE EMBODIED AI ” ICLR2025採択
著者	Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, Hao Su
概要	GPUを用いた高効率な並列シミュレーション・レンダリングにより，従来のロボットシミュレータよりメモリ効率と実行速度を大きく改善できるManiSkill３を公開した
Link	https://arxiv.org/abs/2410.00425 https://www.maniskill.ai/

Agenda

以下の通りで紹介します

1. Introduction
2. Related Work
3. ManiSkill3‘s Features
4. Limitation
5. Conclusion

Agenda

以下の通りで紹介します

1. Introduction
2. Related Work
3. ManiSkill3‘s Features
4. Limitation
5. Conclusion

Introduction

- **Robot Simulatorの活用・課題**
  - Robot制御をRLなどで行う研究では，Robot Simulatorにて得たデータをもとに学習を行われている
    - 実際のロボットからデータをとるのはコストがかかり，スケールしにくい
    - 実機とシミュレータのギャップが性能に大きく影響するため，なるべくギャップを小さくする必要がある

Introduction

- **既存Robot Simulatorの課題**
  - ロボットタスクの中でも特にマニピュレーションタスクは実世界への応用が限定的で，RGB画像に代わるような高度な状態予測が求められる[Handa et al.2023]
  - 既存のGPUシミュレータ（Issac やMujoco’s MJXなど）には，各並列環境ごとに異なる状況をシミュレーションする機能や並列レンダリングを行う機能がない

⇨GPUを用いたより効率的なシミュレーション環境「ManiSkill3」を提案

Agenda

以下の通りで紹介します

1. Introduction
2. Related Work
3. ManiSkill3‘s Features
4. Limitation
5. Conclusion

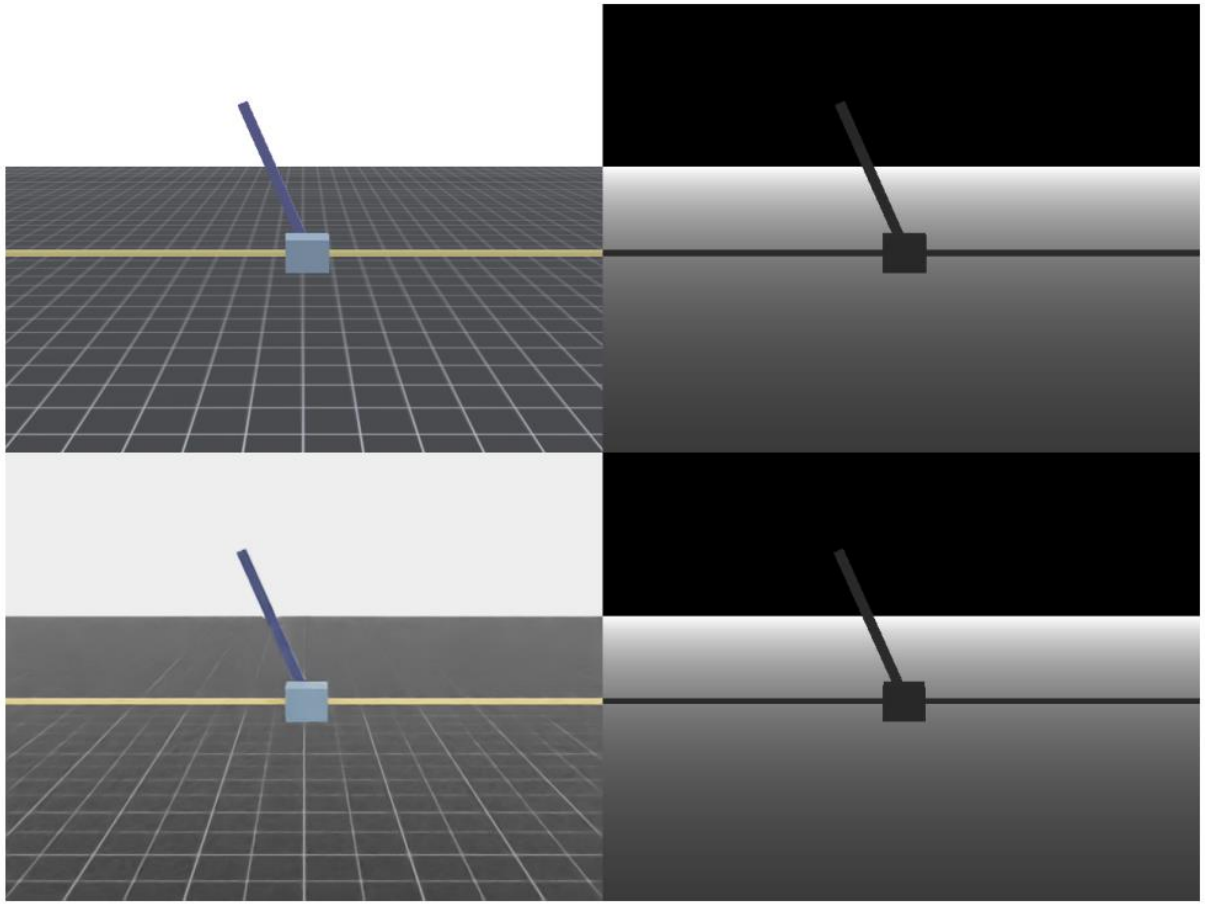
Related Workとの比較

Feature	ManiSkill 3	Isaac Lab	RoboCasa	RLBench	OmniGibson	Habitat	AI2THOR
GPU Parallelized Simulation	✓	✓	✗	✗	✗	✗	✗
GPU Parallelized Rendering	✓	✓	✗	✗	✗	✗	✗
Heterogeneous GPU Parallelized Simulation	✓	✗	✗	✗	✗	✗	✗
Large Scale Demonstrations	✓	✗	✓	✓	✗	✗	✗
Realistic Object Physics	✓	✓	✓	✓	✓	✗	✗
Photorealistic Rendering	✓	✓	✓	✗	✓	✗	✗
Room-Scale Scenes	✓	✓	✓	✗	✓	✓	✓
Visual RL Baselines	✓	✓	✗	✗	✗	✓	✓
Trajectory replay/conversion	✓	✗	✗	✗	✗	✗	✗
Task Categories	12	5	2	1	2	1	1
Interactive GUI	✓	✓	✗	✗	✗	✓	✗

Table 1: Comparison of major features across different open-source robotics frameworks/tools

ManiSkill3 vs Issac Lab

- Cartpoletaskにてそれぞれの FPS と GPU メモリを比較
  - 640x480の RGB Camera と Depth Camera で比較



ManiSkill3 vs Issac Lab

- FPSとGPUメモリの観点でManiSkillが優勢
  - FPSが一定になり始めるとGPU効率が向上することが分かった

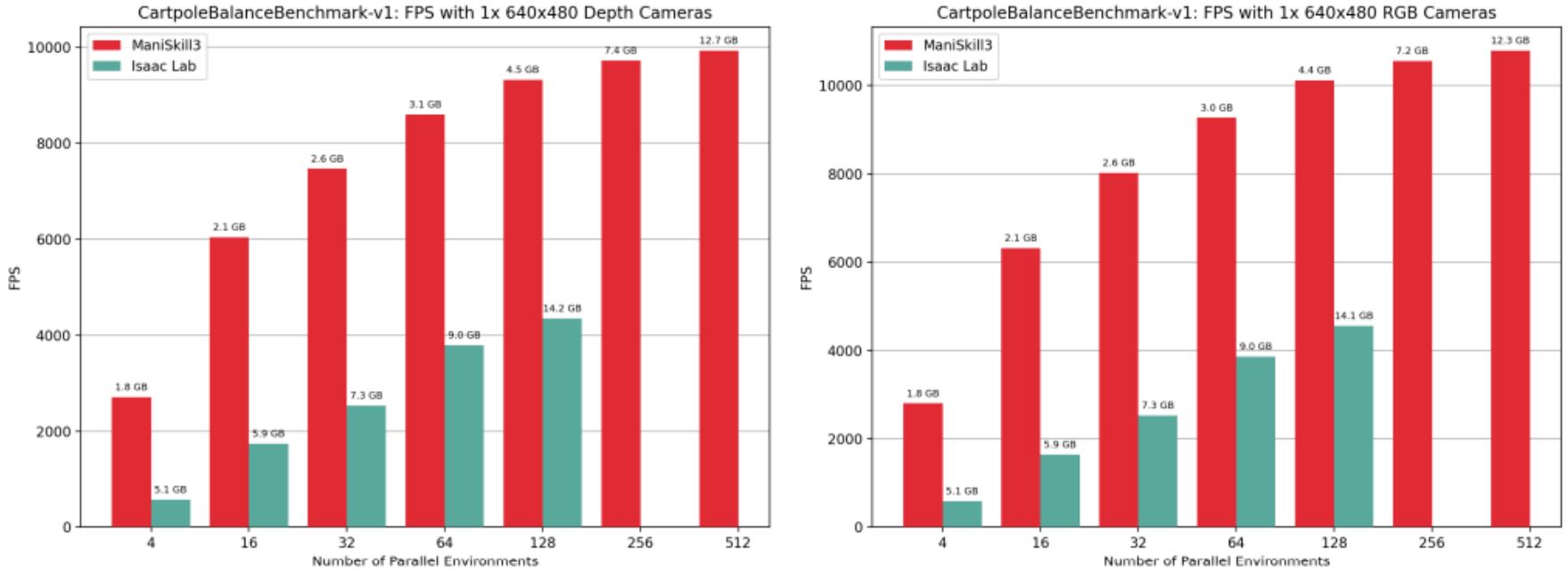


Figure 3: GPU Simulation+Rendering of RGB/Depth speeds of the Cartpole environment with a 640x480 resolution camera in ManiSkill3 and Isaac Lab. Annotated numbers indicate GPU memory

ManiSkill3 vs Issac Lab

- レンダリング後の環境を比較
  - ManiSkill3はIssacなどの環境とレンダリング後の写り方をそろえようとしているため，大きな差は見受けられない

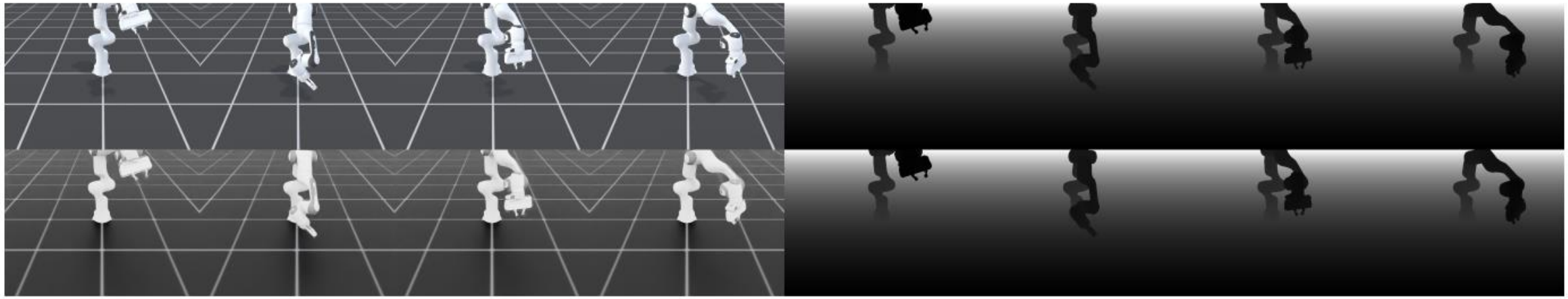


Figure 20: Comparison of ManiSkill3 (Top row) and Isaac Lab (Bottom row) parallel rendering 640x480 RGB+Depth image outputs of the Franka Panda arm.

Agenda

以下の通りで紹介します

1. Introduction
2. Related Work
3. ManiSkill3‘s Features
4. Limitation
5. Conclusion

ManiSkill3の主要な特徴

- 以下機能が既存sim環境より優れている

1. GPUシミュレーションとレンダリングにてSoTAである
2. 12の環境と20以上のロボットがあり，すべて並列計算可能
3. 汎化のための様々なシミュレーションが同時にできる
4. 統一されたAPIを用いて簡単にシミュレーションできる
5. スケール可能なデータ生成のためのパイプラインを提供

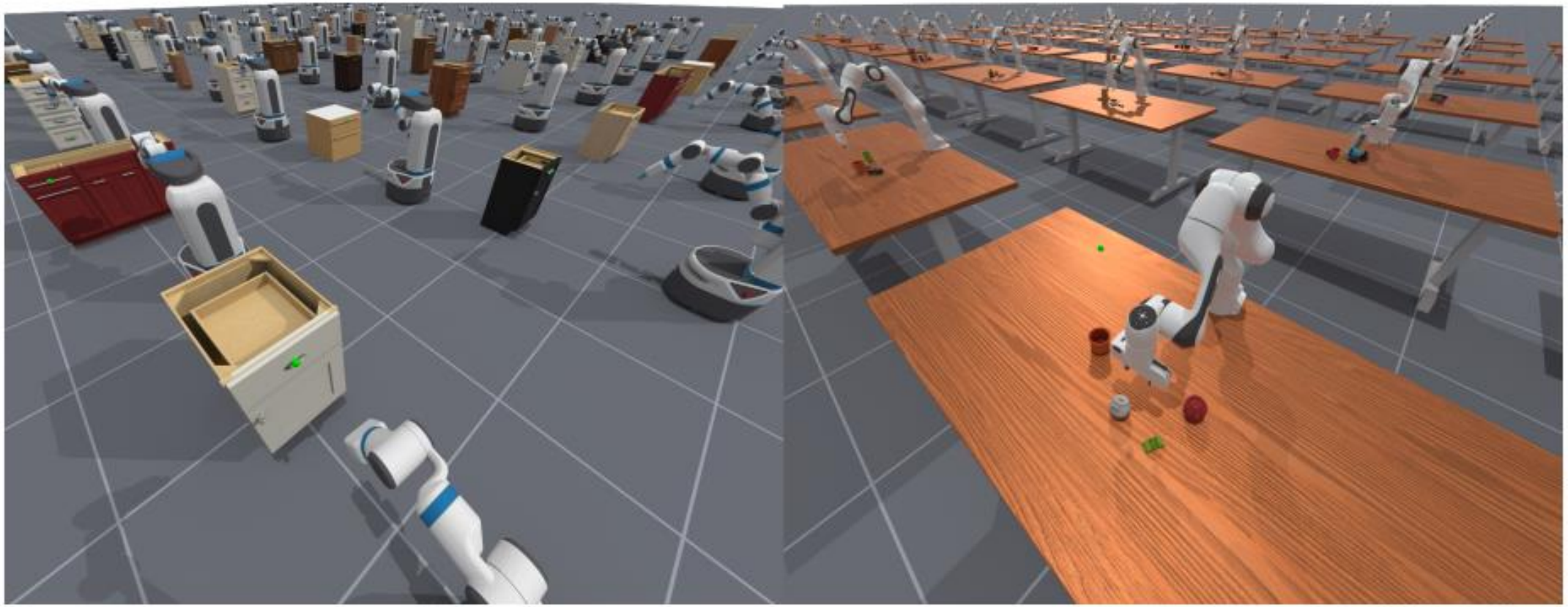
多様なシミュレーション用ロボット・タスク

- ManiSkill3内に様々なロボットや環境が用意されている
  - ドキュメントを参照すると，それぞれの特徴や使い方などが掲載されている

Grade	Description
A+	Values are the product of proper system identification
A	Values are realistic, but have not been properly identified
B	Stable, but some values are unrealistic
C	Conditionally stable, can be significantly improved

並列計算可能な多様なシミュレーション

- 並列で多様な環境のシミュレーションを行う
  - 例：open cabinet drawerタスク（左）
    PickClutterYCB task（右）



APIについて

ManiSkill

```
drawer = cabinet.find_link_by_name("drawer")
qpos = drawer.joint.qpos
drawer_pos = drawer.pose.p
handle_mesh_local_pos = drawer.generate_mesh(
  filter=lambda _, shape: shape.name == "handle",
  mesh_name="handle"
)[0].bounding_box.center_mass
target_gripper_pos = drawer_pos +
handle_mesh_local_pos
success = qpos > 0.5
```

IsaacLab

```
drawer_link_idx = cabinet.find_bodies("drawer")[0]
[0]
joint_idx =
cabinet.find_joints("cabinet_handle_joint")[0]
# note IsaacLab also does not have a usable API for
# sampling meshes so they hardcode mesh positions
handle_mesh_local_pos = torch.tensor([0.3, 0.01,
0.0])
handle_mesh_local_pos =
drawer_local_grasp_pos.repeat((self.num_envs, 1))
qpos = cabinet.data.joint_pos[:, joint_idx]
drawer_pos = cabinet.data.body_pos_w[:,
drawer_link_idx]
target_gripper_pos = drawer_pos +
handle_mesh_local_pos
success = qpos > 0.5
```

IsaacGymEnvs

```
# calls to self.gym, self.cabinet_dof_indexes, etc.
are computed/setup
# elsewhere and excluded as they take up a lot of
space and are hard to read
rigid_body_tensor =
self.gym.acquire_rigid_body_state_tensor(self.sim)
rigid_body_states =
gymtorch.wrap_tensor(rigid_body_tensor).view(self.nu
m_envs, -1, 13)
dof_state_tensor =
self.gym.acquire_dof_state_tensor(self.sim)
drawer_handle =
self.gym.find_actor_rigid_body_handle(
    self.env_ptr, cabinet_actor, "drawer")
# note IsaacGymEnvs does not have a usable API for
# sampling meshes so they hardcode mesh positions
handle_mesh_local_pose = gymapi.Transform()
handle_mesh_local_pose.p =
gymapi.Vec3(*get_axis_params(0.01, grasp_pose_axis,
0.3))
handle_mesh_local_pos = to_torch(
    [drawer_local_grasp_pose.p.x,
drawer_local_grasp_pose.p.y,
     drawer_local_grasp_pose.p.z],
     device=self.device
).repeat((self.num_envs, 1))
cabinet_dof_state = dof_state_tensor.view(num_envs,
-1, 2)[:, self.cabinet_dof_indexes]
cabinet_dof_pos = cabinet_dof_state[..., 0]
qpos = cabinet_dof_pos[:, 3] # hardcoded index for
the drawer joint
drawer_pos = rigid_body_states[:, drawer_handle][:,
0:3]
target_gripper_pos = drawer_pos +
handle_mesh_local_pos
success = qpos > 0.5
```

- オブジェクト指向に基づきAPIが設計されている
- 他フレームワークでは必要だったテンソルindを廃止している

→コードを短く，
より扱いやすい環境を提供

ManiSkill3によるデジタルツイン

- sim2realのテストも進んでおり，デジタルツインを用いて95%程度のタスク成功率を達成した

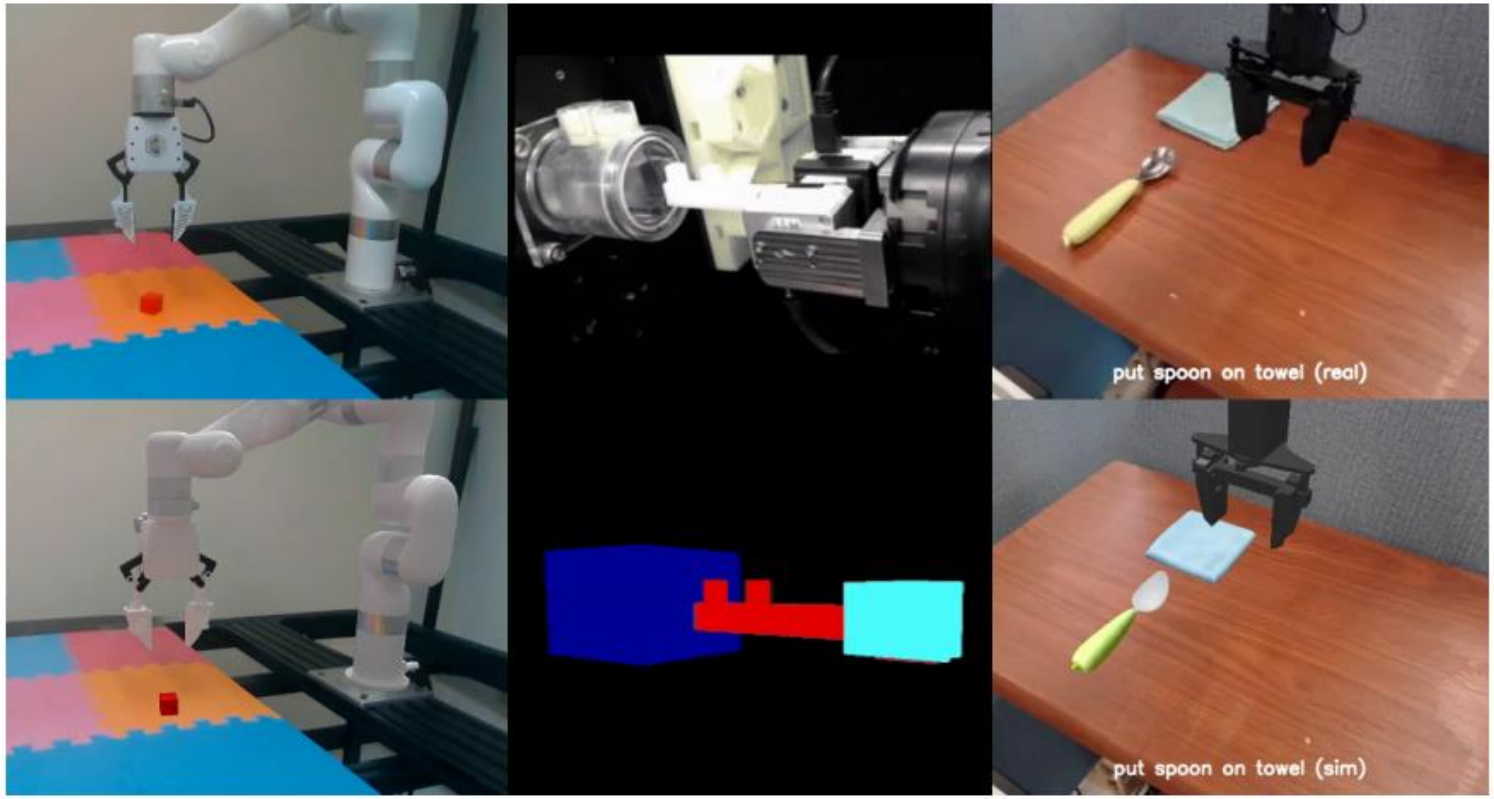
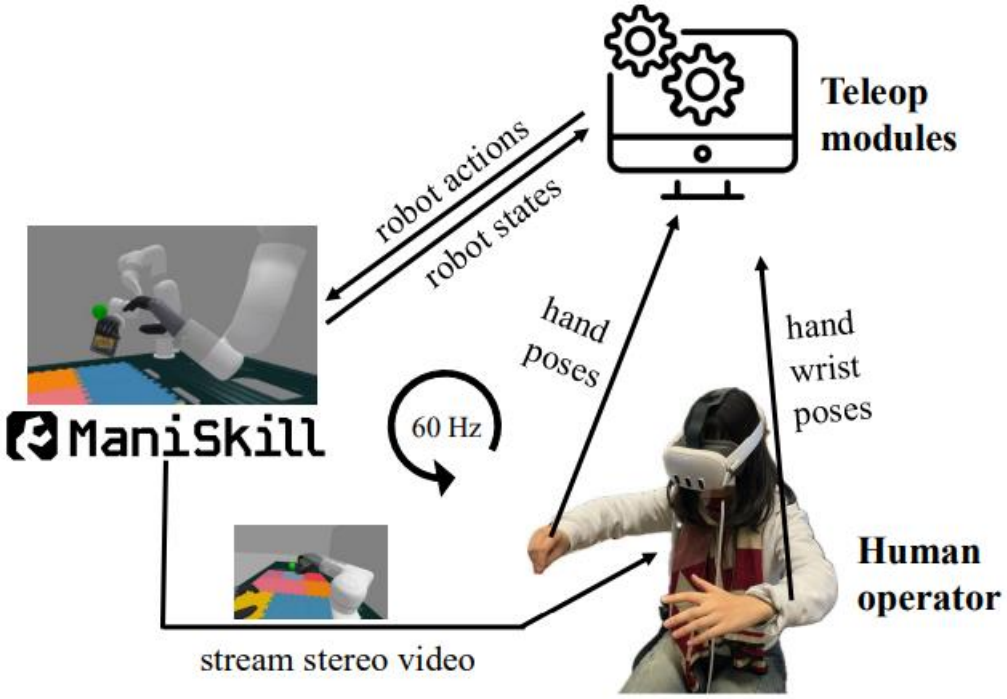


Fig. MiniSkill3によるデジタルツイン（下）

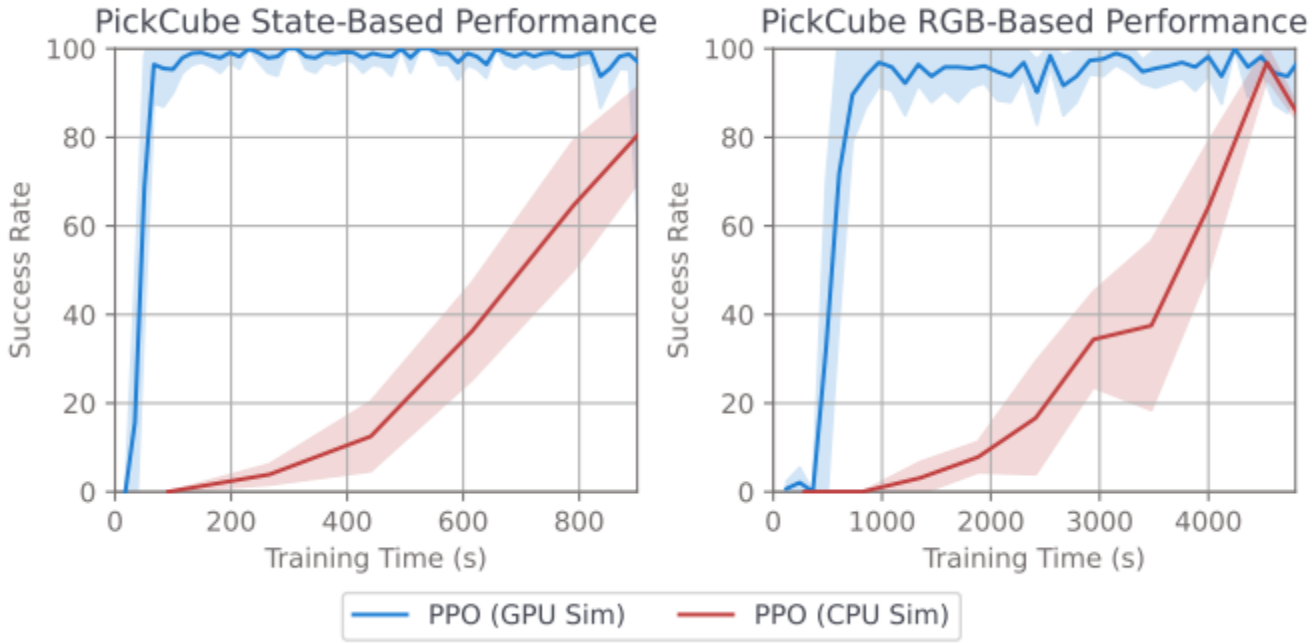
ManiSkill3によるテレオペレーション



- VRゴーグルを用いて，ManiSkill3内のロボットを遠隔操作できる
  - **Arm Control Module**
    - 関節角度に制約をつけた
    - 逆運動学に基づいて姿勢を決定する
  - **Hand Control Module**
    - 人間の指をロボットハンドの関節位置に変換している
  - **Controler Module**
    - 物をつかむ際の動きを直感的に再現した

PPOの訓練パフォーマンス紹介

- ManiSkill3を用いてPPOにより学習した
  - 過去最速だったManiSkill2の8-15倍ほど速く収束するようになった
  - RL側でPPOより早く収束するようなアルゴリズムが出てくれば,
    より早く解を得られるようになる



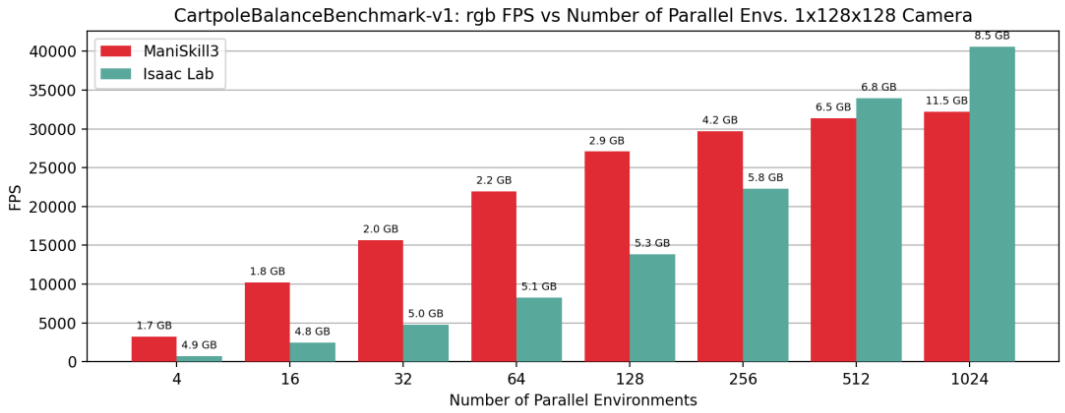
Agenda

以下の通りで紹介します

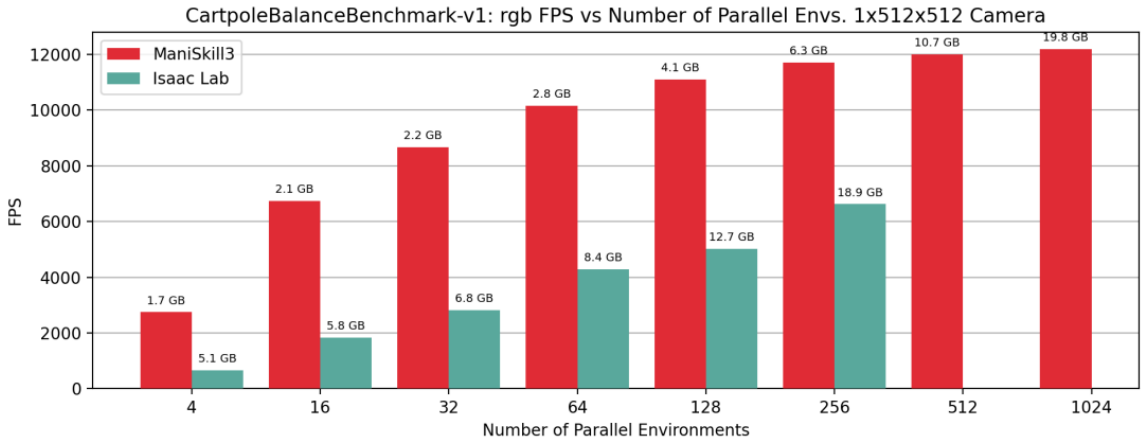
1. Introduction
2. Related Work
3. ManiSkill3‘s Features
4. **Limitation**
5. Conclusion

Limitation

- カメラ解像度が低い場合はIssac LabがMiniSkill3の最大1.2倍ほど速くなる
  - 反対にカメラ解像度が高い場合はMiniSkill3が2倍ほど速い



カメラ解像度：128 x 128



カメラ解像度：512 x 512

Limitation

- **並列計算の限界**
  - 複雑なジオメトリを伴うような環境だと並列計算可能な環境数が減少する
  - 剛体ベースの環境は可能だが，軟体のある環境はそれ自体でGPUをかなり逼迫するため，バッチ化されていない
- **Sim2realの課題**
  - レンダリングを高速化したため，RGBベースのsim2realが可能になったが，より堅牢なRLポリシーとなるような報酬設計が必要である
  - 現在のデモは静的なカメラ使用に限定されている

Conclusion

- 並列シミュレーション・レンダリングを効率化するため，ManiSkill3を提案した
  - 様々な環境のシミュレーションを並列で行える
  - 12の環境，20以上のロボットが用意されている
  - デジタルツイン可能なことを確認しており，タスク完了率は約95%
  - VRゴーグルを使ったテレオペレーションが行える

- 既存シミュレータと比べて，GPUメモリ効率は2-3倍，計算速度は10-1000倍程度改善した