Real Robot and Sim -to-Real

Building and Working in Environments for Embodied AI (part V)

CVPR 2022 Tutorial

清華大學
Tsinghua University

SIMON FRASER
UNIVERSITY

Overview

- We are going to talk about:
  - Sim-to-real gaps for robots
  - How to mitigate the gaps
- This section is mainly for people who would like to deploy embodied AI in the real world.

Outline

- Suggested pipeline to deploying policy in the real world
- Real and simulated 3D sensors

Outline

- Suggested pipeline to deploying policy in the real world
- Real and simulated 3D sensors

Problem Formulation

Assume a visual policy $\pi(a|s)$ from the simulator

How do we deploy this policy π to the real robot?

Challenges

Sources of issues may come from:

- Robot communication
- Actuator accuracy
- Sensor noise
- World model mismatch
- Controller implementation
- …

Communication & Actuator Accuracy

- Communication Latency
  - Control latency
  - Sensor latency
- Deformation of components
  - E.g., robot finger can be soft

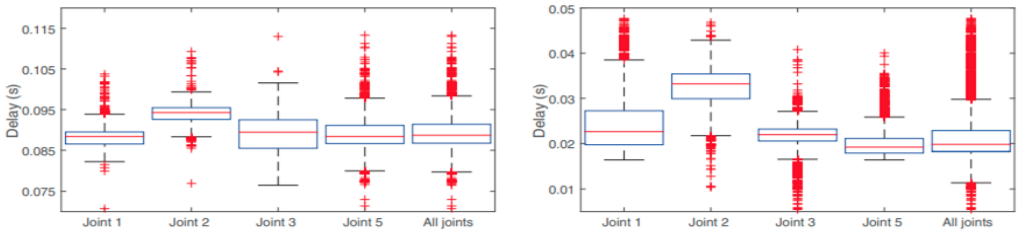


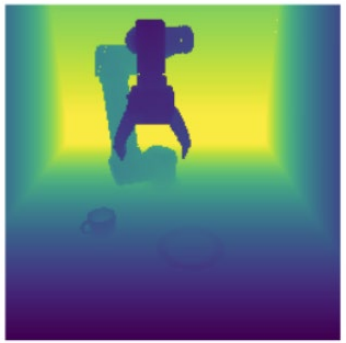
Figure 8. Boxplot of individual joint's actuation delay. Note the different y axis interval. Left: Kuka. Right: Universal Robot.

Andersen, Thomas Timm, et al. "Measuring and modelling delays in robot manipulators for temporally precise control using machine learning. "Conference on Machine Learning and Applications (ICMLA) 2015.

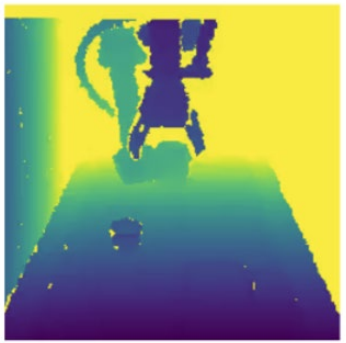
Sensor Noise

- Proprioception sensor noise
  - Joint position
  - Joint velocity
  - Joint torque
- Environmental sensor noise
  - Visual domain gap

Collecting demonstrations



Running on real robot



World Model Mismatch

- Jerk limit of real actuator (da/dt)
- Error in modeling environment
  - Geometry difference
  - Physical property difference
  - …

Controller Implementation

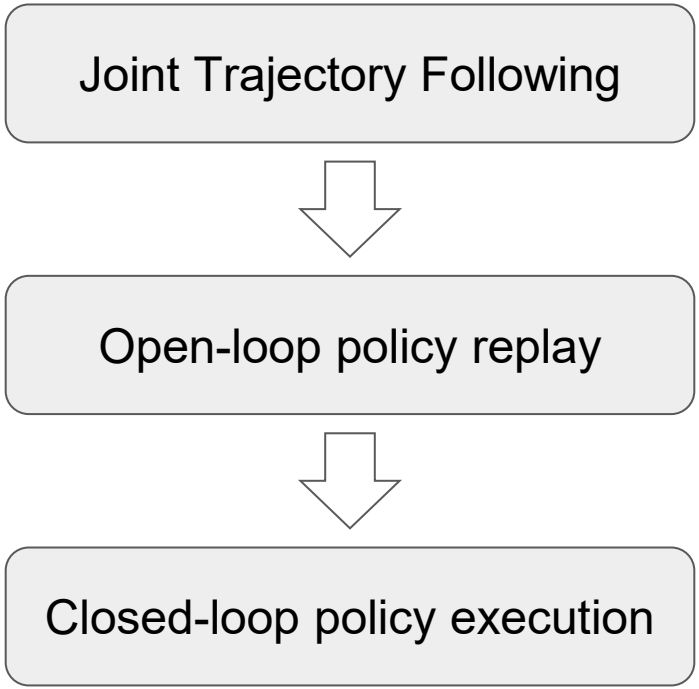
Simulation API	Available on Real?	Notes
PD joint/Cartesian position control	Yes	
PD joint/Cartesian velocity control	Yes	
Set PD parameters	Probably	
Get joint position	Yes	
Get joint velocity	Yes	Noise caused by differentiation
Get joint torque	Maybe	Large Sim2Real gap

Controller Implementation

Simulation API	Available on Real?	Notes
Forward/Inverse kinematics	Yes	
Forward/Inverse dynamics	Maybe	
Passive force	Maybe	
Torque control	Maybe	Run at high frequency (>100Hz)
Get inertia/Coriolis matrix	Maybe	
Get (calibrate) friction	Maybe	

We suggest that you take a 3-step approach to address challenges progressively!

3-Step Deployment Pipeline



3-Step Deployment Pipeline

Joint Trajectory Following

Goal: Check the **communication** and **accuracy** of the real robot

Practice

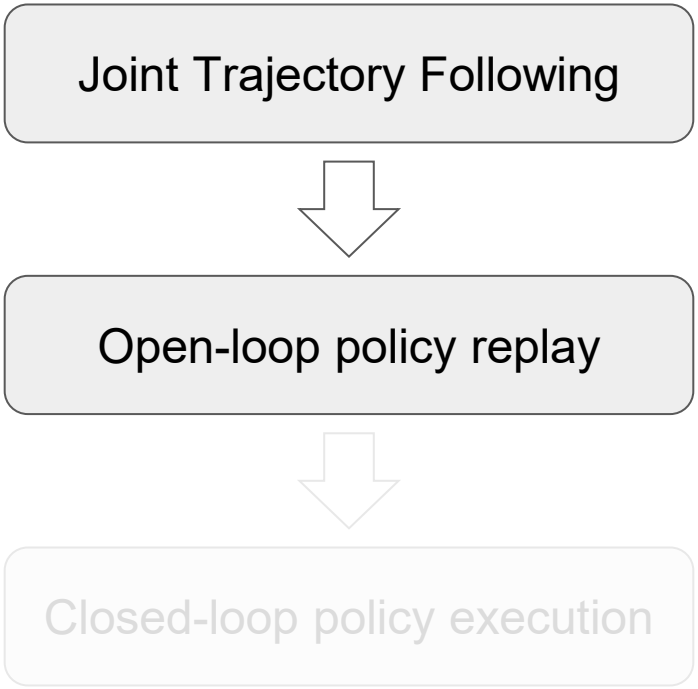
Assume we have a policy π obtained from the simulator

- The **simulator rollout** yields $(s_1, a_1, s_2, a_2, \ldots, s_n)$.
- Extract robot joint positions from the trajectory $\{s_i\}$.
- Conduct the robot joint position trajectory following on the real robot.
- Compare the real robot motion with that in simulator visually, they should be the same.

Explanation and What to Expect

- Basically all real robots support joint trajectory following.
- Joint position controllers can avoid accumulation of execution errors.
- Sufficient for tasks in static scenes (e.g., simple grasping)
- If there is a big difference between sim and real, the most likely cause is implementation bugs.

3-Step Deployment Pipeline



Goal: Check the **communication** and **accuracy** of the real robot

Goal: Check the **controller** of the policy action space on the real robot

Practice

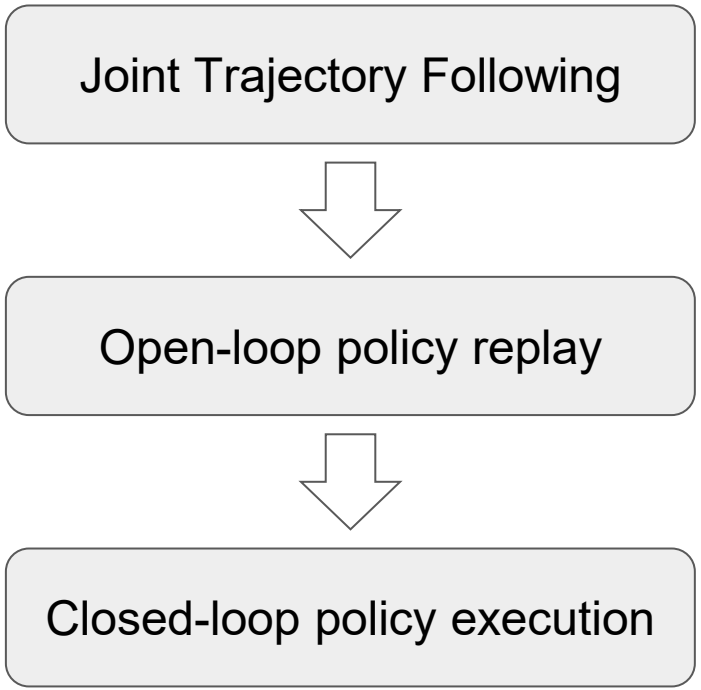
Assume we have a policy π obtained from the simulator

- The **simulator rollout** yields $(s_1, a_1, s_2, a_2, \ldots, s_n)$
- Extract robot actions $\{a_i\}$
- Execute the action sequence on the real robot.

Explanation and What to Expect

- The policy action space may involve modules whose performance are sensitive to implementation details, like
  - Inverse kinematics
  - Velocity controllers
- The robot may fail due to
  - Issues in controller implementations (mismatch, wrong params)
  - Error accumulation along task horizon

3-Step Deployment Pipeline



Goal: Check the **communication** and **accuracy** of the real robot

Goal: Check the **controller** of the policy action space on the real robot

Goal: Check the influence of the **visual gap**

Practice

Assume we have a policy π obtained from the simulator. This time we DO NOT use simulation rollout. We rollout online.

1. Get the observation from real-world, e.g.RGB image from camera, depth map from 3D sensor, robot proprioception.
2. Output the action from observation with the policy.
3. Execute the action.
4. Repeat 123.

Explanation and What to Expect

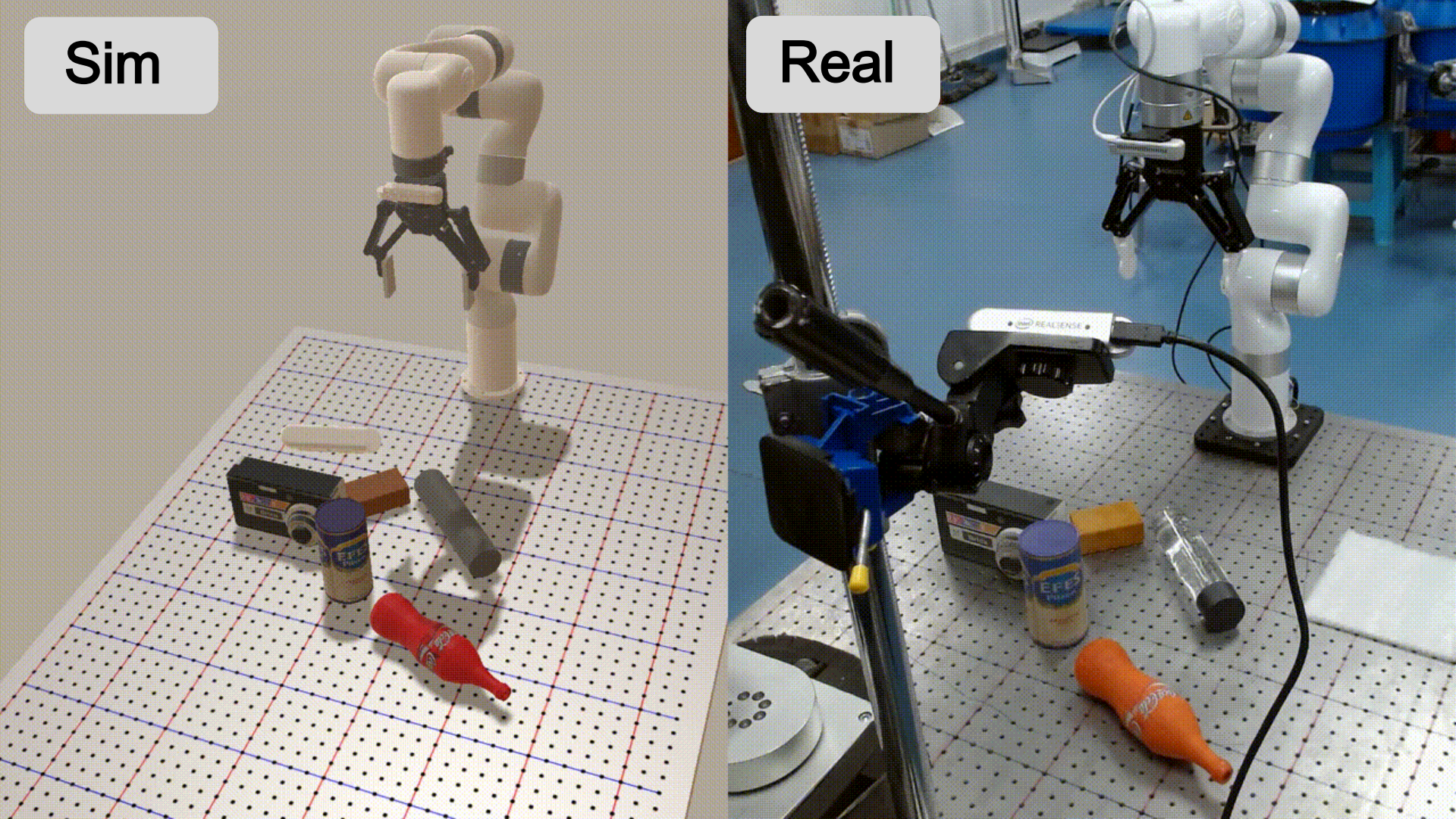
- We use feedback (e.g., visual sensors) from the real world to correct errors in policy execution (thus closed-loop)
- Therefore, we are badly affected by sensor simulation gap (e.g., visual gap)
- For non-static scenes, latency gap may be a big issue
- When everything is done correctly, your real robot
  - can deal with dynamic scenes
  - can be more robust than open-loop systems

Additional Tip for Sim-to-Real

- Most of the time the problem is not sim-to-real gap, but only a bug.

Finally, the Emergency Stop!

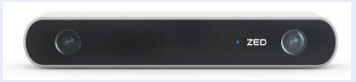
Safety first, you and your robot!


Sim
Real

Outline

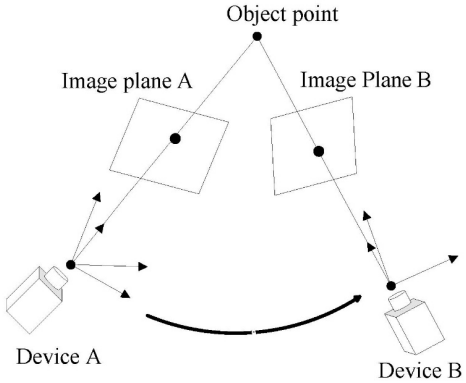
- Suggested pipeline to deploying policy in the real world
- Real and simulated 3D sensors

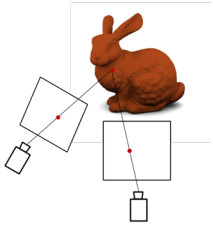
3D Optical Sensor

<table>
<tr><td rowspan="3">Depth by Triangulation</td><td>Passive Stereovision</td><td>BumbleBee2, ZED
   </td></tr>
<tr><td>Active Stereovision</td><td>Intel RealSense D415, D435
   </td></tr>
<tr><td>Structured Light</td><td>Kinect V1, Orbbec Astra
   </td></tr>
<tr><td colspan="2">Time-of-Flight</td><td>Helios, Kinect V2
   </td></tr>
</table>

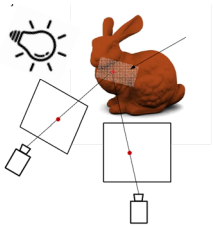
Principles

Depth by triangulation

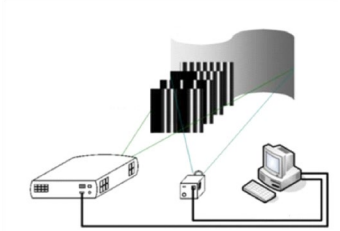




Passive Stereovision



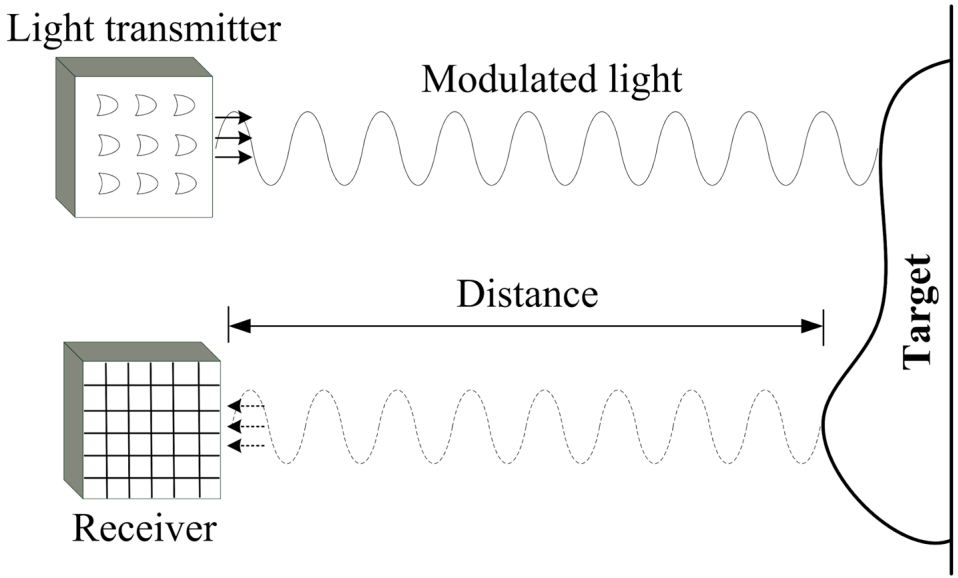
Active Stereovision



Structured Light

Principles

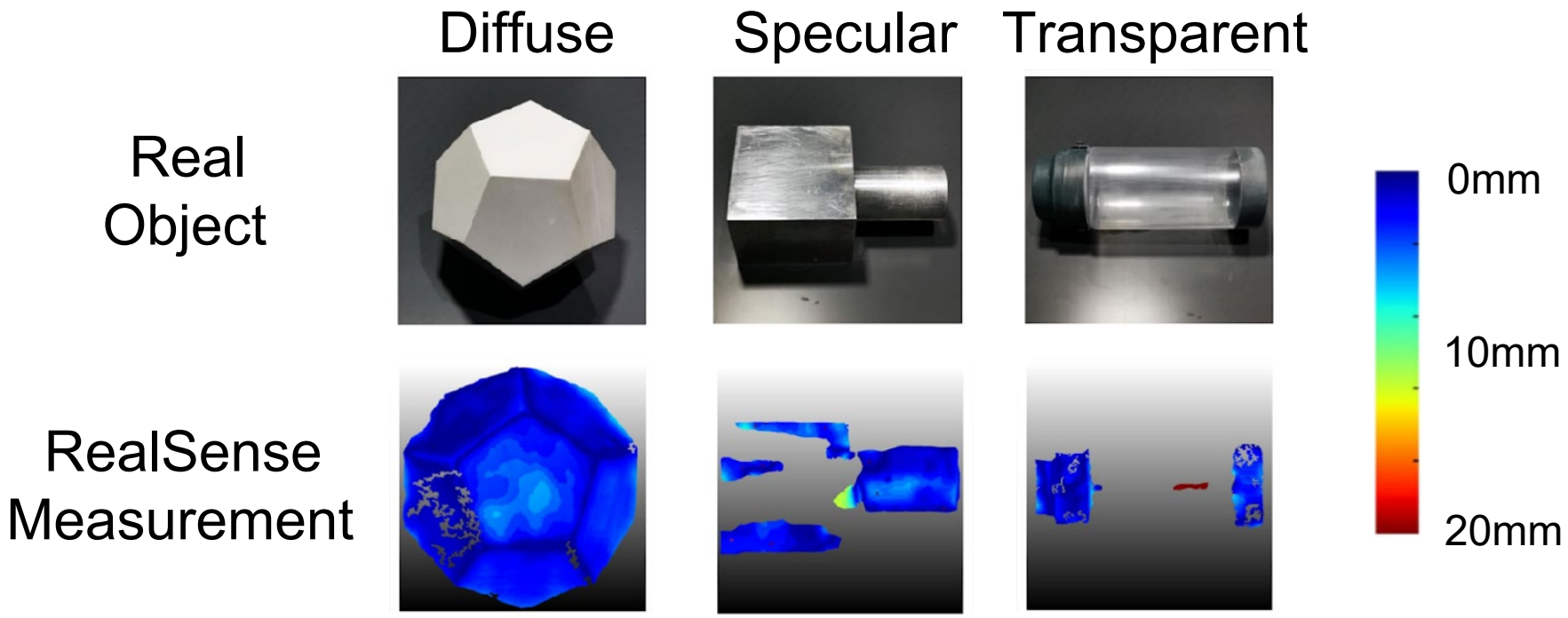
Depth by time-of-flight



Case Study:
Active Stereovision Sensor

- Advantages
  - Support dynamic scenes
  - Support textureless areas
  - Robust to environmental illumination
  - Light weight

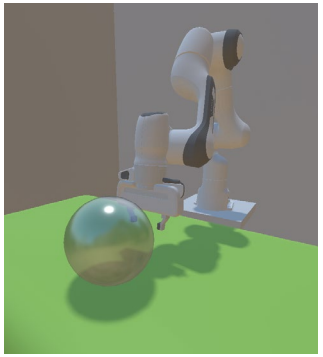
Optically Challenging Objects

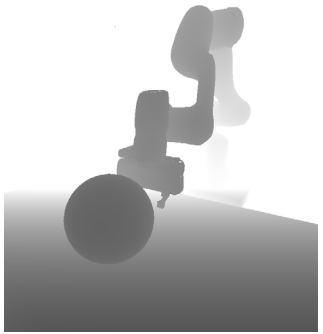


Captured by ourselves

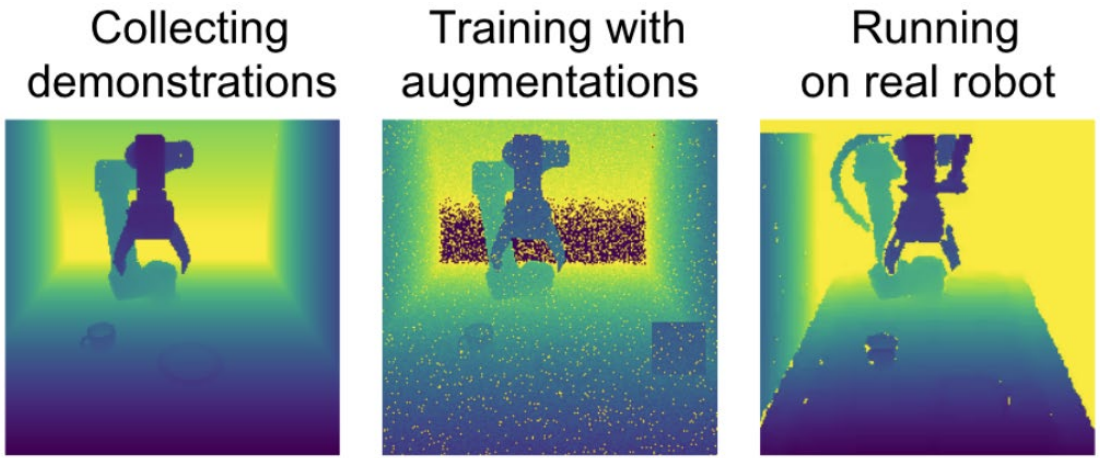
Depth Sensor Simulation

- Most environments use “ground truth” depth directly from the depth buffer.
  - **Pro:**
    - Simple
    - Small overhead
  - **Con:**
    - Does not model occlusion.
    - Does not model artifacts from specular and transparent objects.





Depth Sensor Simulation Methods



Noise-based depth map augmentation

Pro:

Optimized augmentation strategy by searching

Con:

Cannot simulate the complex real error pattern by combining simple augmentations

Pashevich, Alexander, et al. "Learning to augment synthetic images for sim2real policy transfer." 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.

Depth Sensor Simulation Methods

GAN-based approach

Pro:

Can model real error distribution in theory

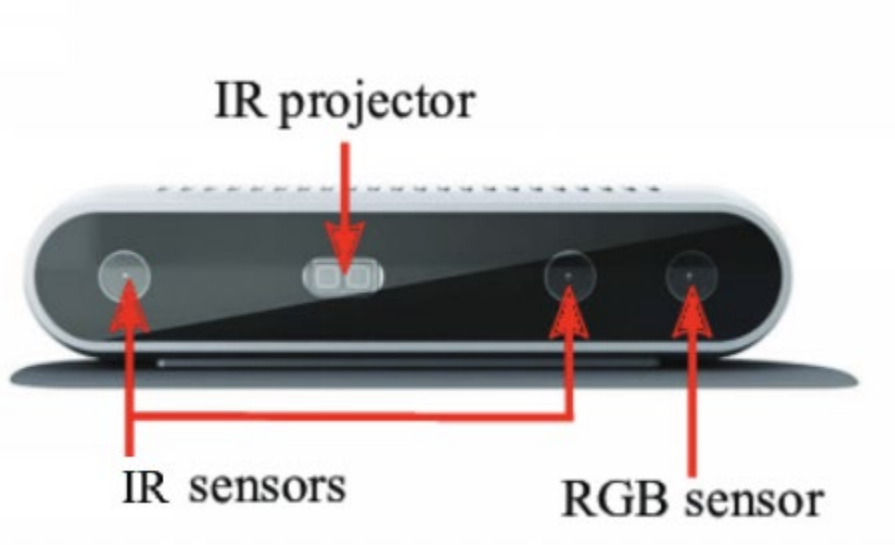
Con:

Not physically plausible

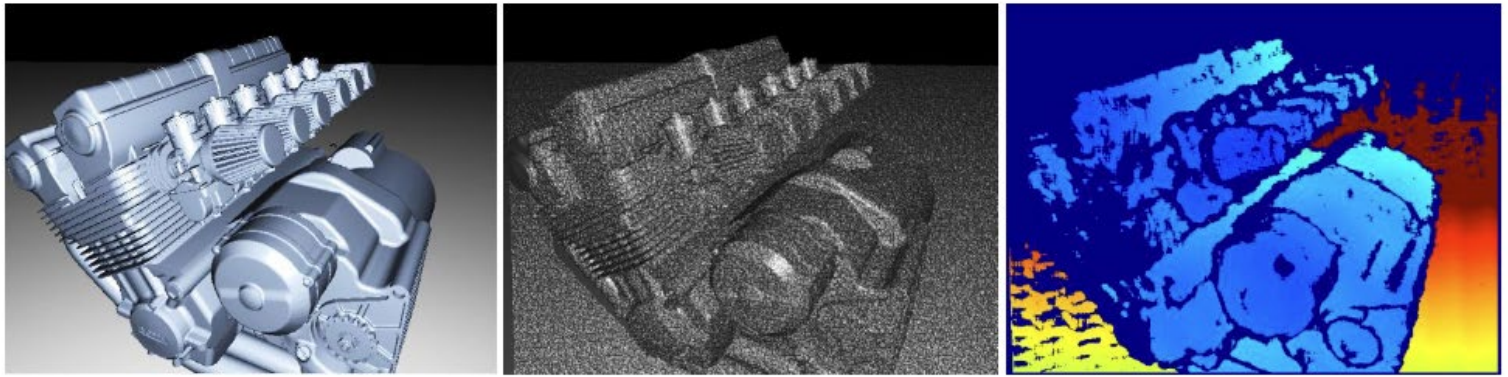
Bousmalis, Konstantinos, et al. "Unsupervised pixel-level domain adaptation with generative adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Physics -Grounded Depth Sensor Simulation

Active Stereovision Sensor



Simulated Stereo Matching from Rasterized Images



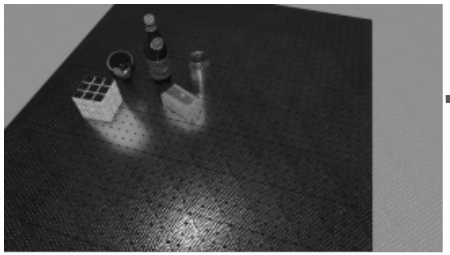
RGB IR Depth

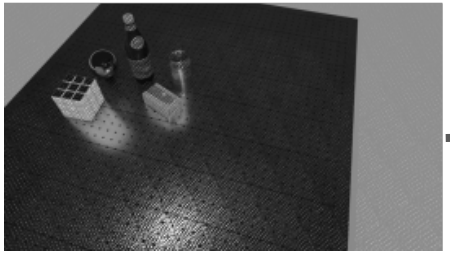
Pro:
Can simulate the error caused by **stereo matching**
Con:
Cannot simulate the light transport of **transparent** or **translucent** objects

Planche, Benjamin, et al. "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition." 2017 International Conference on 3D Vision (3DV). IEEE, 2017.

Physics -Grounded Depth Sensor Simulation

Stereo IR images by
Ray-tracing

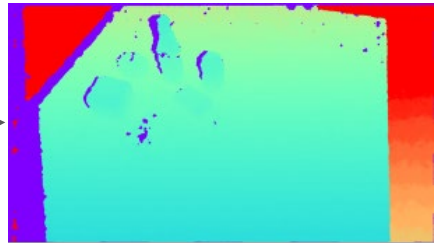




Sensor Noise Model

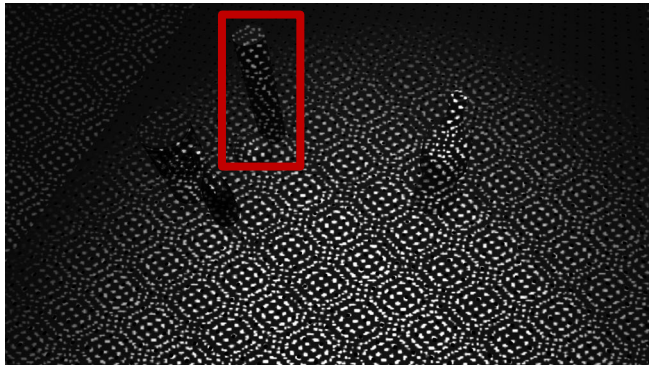
Sensor Noise Model

Stereo Matching

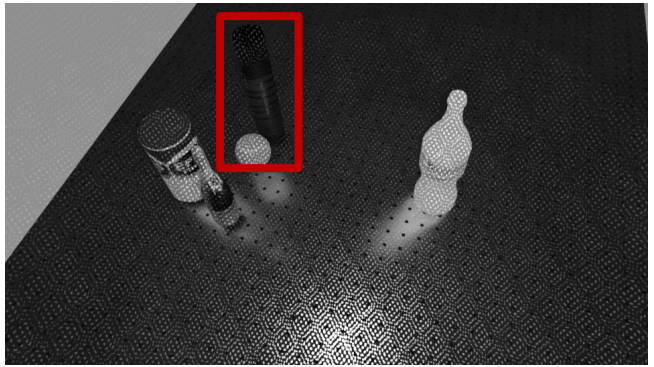


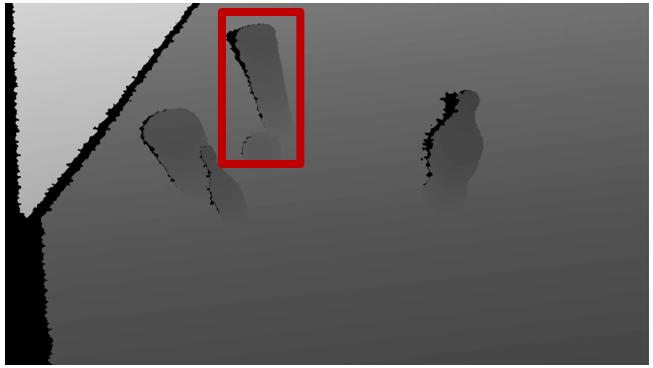
Simulated depth

Rasterization vs. Ray -tracing

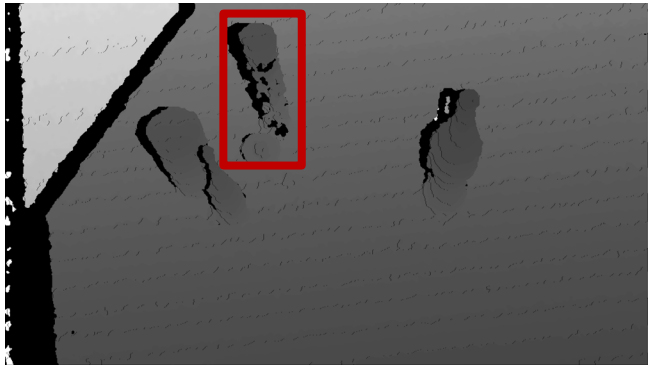


IR





Depth



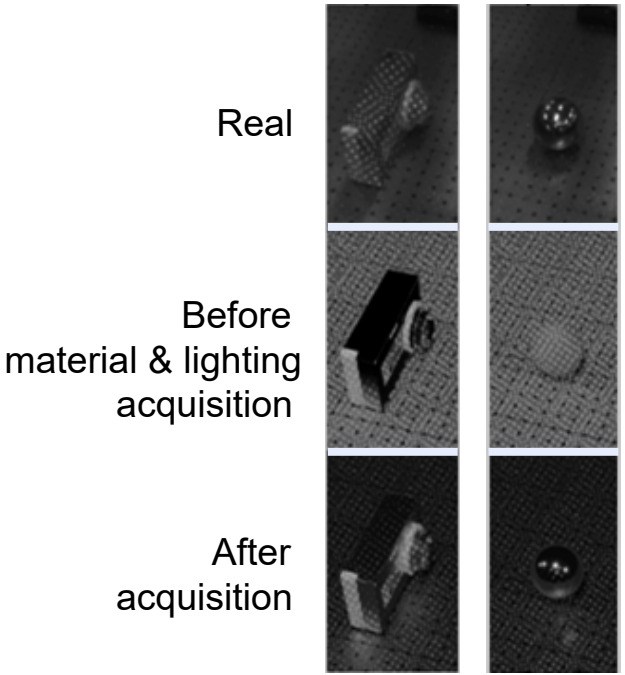
Is a Correct Renderer All We Need?

- With a physically accurate ray tracer, and a realistic noise model, is the gap closed?

Is a Correct Renderer All We Need?

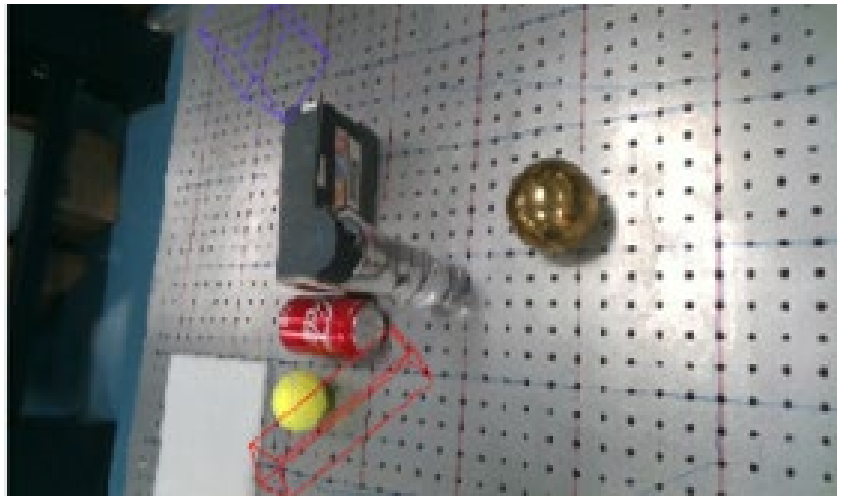
- With a physically accurate ray tracer, and a realistic noise model, is the gap closed?

- What else do we need?
  - Realistic lighting
  - Realistic material



Application: Sim-to-Real 6D Pose Estimation

Train on **simulation only** , test on real



Existing simulation

Physics-grounded simulation

Paper: https://arxiv.org/abs/2201.11924
Document of Depth Sensor Simulation in SAPIEN:
https://sapien.ucsd.edu/docs/2.0/tutorial/rendering/raytracing_renderer.html

Real Robot and Sim -to-Real

Building and Working in Environments for Embodied AI (part V)

CVPR 2022 Tutorial

清華大學
Tsinghua University

SIMON FRASER
UNIVERSITY