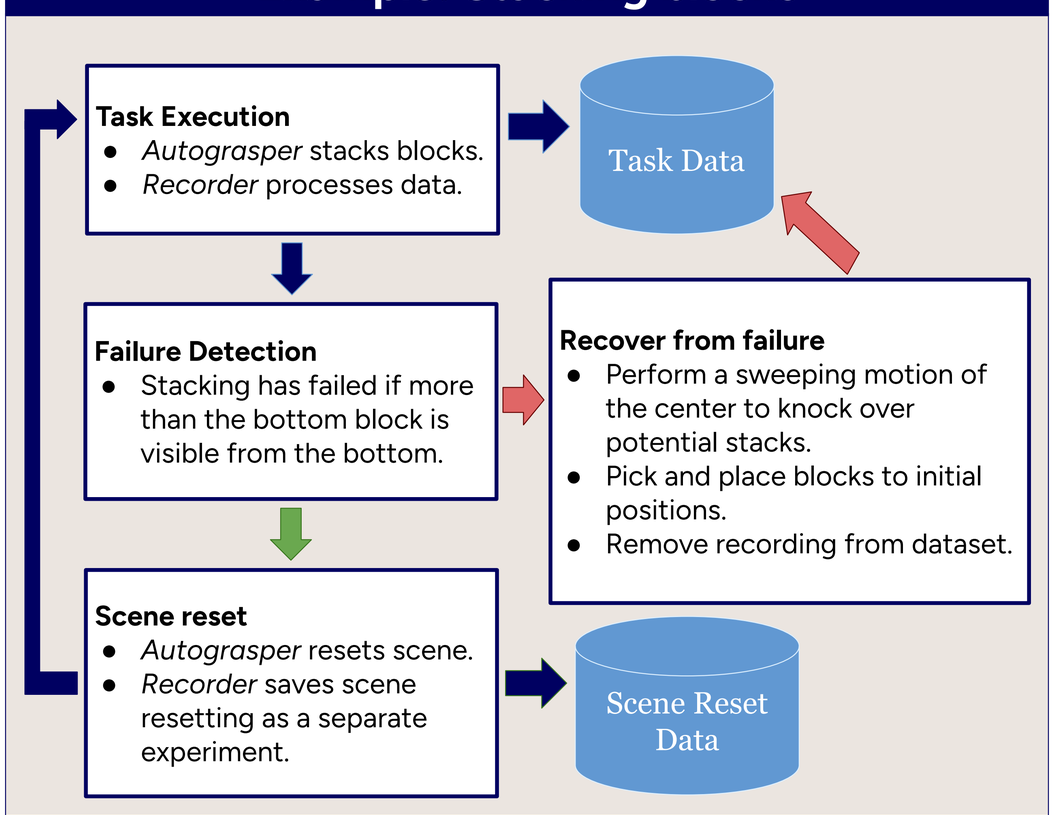
CloudGripper-Autograsper: A Cloud Robotics Toolkit for Automatic Data Collection

An Open-Source Toolkit for Autonomous Robotic Manipulation Data Collection and Seamless Simulation-to-Reality Transition

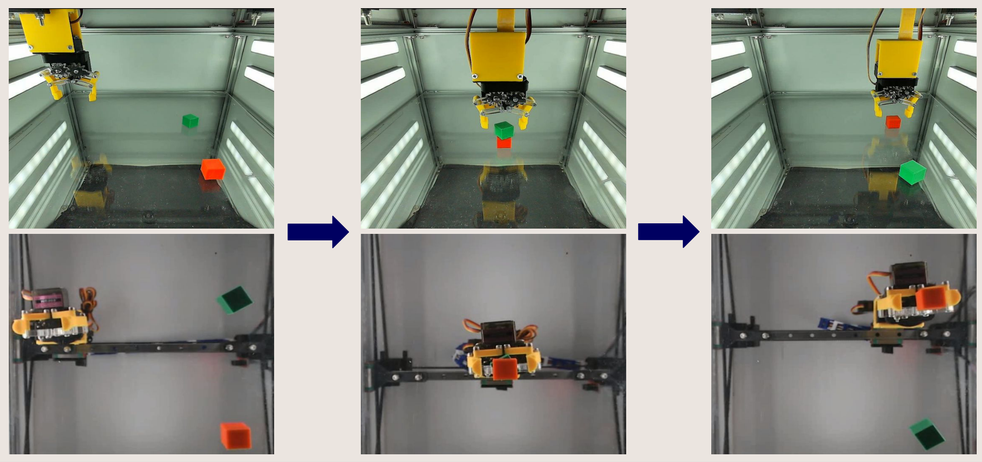
Motivation

- **Efficient data collection** remains a significant challenge in robotic manipulation, often requiring **costly human intervention** for tasks like scene **resetting after failures**.
- We present **CloudGripper-AutoGrasper**, an open-source toolkit designed address these issues. The toolkit is Built on the CloudGripper platform [1] —featuring 32 robotic arm cells— and is compatible with the CloudGripper simulation.

Example: Stacking blocks



Automating data collection



Task Definition + Failure Conditions + Recovery Sequence

Continuous Data Collection Without Human Intervention

- During task execution, the system **automatically detects failed states** by continuously monitors robotic joint data and camera feeds.
- It employs image processing techniques, such as contour detection using OpenCV, to detect object misplacements.
- Upon detecting a failure, the robot initiates an **error recovery process** that includes sweeping actions to make all objects visible and sequential pick-and-place operations to **reset the objects to designated positions**.

Highlights and results

Features:

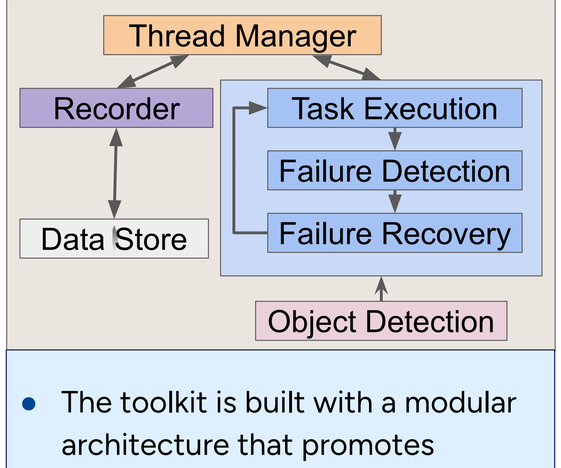
- Automatic scene resetting
- Compatibility with CloudGripper simulation
- Real time failure detection
- Error recovery

Results:

- **200 episodes collected without manual intervention** in under 8 hours.
- The source code for the CloudGripper-Autograsper toolkit is **publicly available** along with the collected dataset (QR code).

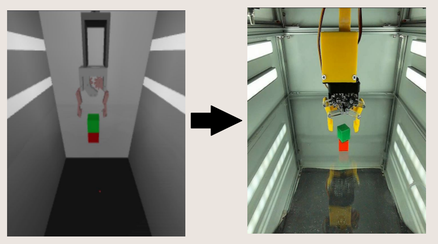
[1] Zahid, Muhammad, and Florian T. Pokorny. "CloudGripper: An Open Source Cloud Robotics Testbed for Robotic Manipulation Research, Benchmarking and Data Collection at Scale." 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2024.

Modular design



- The toolkit is built with a modular architecture that promotes **scalability** and **flexibility**.
- The independence of each module enables **changing some modules without the others**, for instance updating the failure conditions while keeping the task and failure recovery.

Sim2Real



```
robot = SimRobot()
robot = RealRobot()
```

- Seamless transition between simulated and real-world environments through compatible APIs.
- Users can switch between simulation and physical robots with a **single line of code.**

Axel Kaliff, Shutong Jin, Muhammad Zahid, Florian T. Pokorny

KTH Royal Institute of Technology