Supplementary Materials

Yixuan Wang[1,2], Leonor Fermoselle[2], Tarik Kelestemur[2], Jiuguang Wang[2], Yunzhu Li[1]

I. Method

A. Unknown Space

Using depth image and camera parameters, we could create a voxel representation of the space. For each voxel, we can have labels, such as `unexplored`, `free`, `unknown`, and `outside`. The definition of each label is listed below:

- `unexplored`: If a voxel is never viewed by a sequence of camera observations, the voxel is labeled as `unexplored`.
- `free`: If a voxel is viewed by a sequence of camera observations and it is in the free space, the voxel is labeled as `free`.
- `unknown`: If a voxel is viewed by a sequence of camera observations and it is occluded, the voxel is labeled as `unknown`.
- `outside`: If a voxel is viewed by a sequence of camera observations and it is outside of the room, the voxel is labeled as `outside`.

Such a voxel representation is critical for VLM decision. For example, when there are a lot `unknown` voxels inside the cabinet or behind the box, VLM will utilize such information regarding the unknown space to decide where to interact.

B. Relation Detection

Given such voxel representation, we can define the following object relations:

- `of`: If a child object can be possibly a part of the parent object, such as a handle is possibly a part of a cabinet, and their point cloud centroids are close enough, the child object node is `of` the parent object node.
- `inside`: If a child object is found after opening or flipping a parent object, the child object is `inside` the parent object node.
- `on`: If a child object's bounding box is within the parent object's bounding box in the x-y plane, and the child object's lowest z value is close to the parent object's highest z value, the child object is `on` the parent object node.
- `under`: If a child object is found after sitting down or lifting, the child object is `under` the parent object node.
- `behind`: If a child object is found after pushing a parent object aside, the child object node is `behind` the parent object node.

C. Skill Construction

We define the manipulation skills, including `open`, `flip`, `lift`, `push`, `sit`, and `collect`. Specifically, the skills are defined below:

- `open`: To open a cabinet, we first detect the handle object node and the cabinet object node. Using principal axis analysis, we could know the axis of the handle and the normal axis of the cabinet. Given the handle position, we can define the spot end-effector position for grasping the handle. Since we do not know the articulation of the cabinet, we will first open the door using impedance control. Since the door might open via translational motion or rotational motion, we will adjust the target end-effector position and pose according to the feedback from the manipulation results.
  In addition, to make the skill more robust, we take in the grasping feedback. If the grasping fails, we will retry the grasping with different action parameters.
- `flip`: We assume that we only flip open boxes. The end-effector will first enter the open box in a top-down pose. Then it pushes the box side down to flip the box, as shown in our video.
- `lift`: We assume that we only lift clothes. The end-effector first approaches the cloth and grasps it using the top-down pose. The grasping position is the center of the cloth.
- `push`: In our experiments, we will push large objects aside. Specifically, it will walk to the front of objects first and then walk aside. To push objects, it will stretch its arm and walk aside with the stretching arm.
- `sit`: In our experiments, we will walk to the front of the object, and then sit down to check the space under it. Then sitting action can be executed using Boston Dynamics API.
- `collect`: To collect the object, we will first grasp objects using the off-shelf grasping planner or heuristics. Then we command the end-effector to go to the designated pose and place the object into the blanket.

With these manipulation skills, our robot explore the environment interactively and exploit the built graph.

https://bdaiinstitute.github.io/curiousbot/
[1]Columbia University [2]Boston Dynamics AI Institute