UR5 Obstacle-Avoidance Pick-and-Place with Arm Assistance

EN.530.707 Robot System Programming

Yucheng Kang, Zhiyi Ren

April 3rd, 2018

Motivation

When UR5 faces a cluster of obstacles and objects, it may find it difficult to map the environment and move objects with a RGBD camera alone. Unlike mobile robot that can avoid obstacles and map the environment as it moves, UR5 motion is much more restricted. We would like to use human arm movement to guide UR5 move around, avoid the obstacles, and construct the map first before moving objects.

During teleoperation, human hand may not be accurate enough to pick up the object. Visual servoing can provide much more precise operation when the object is close and in sight.

Existing motion planning algorithms may cause UR5 to move to a few awkward poses before reaching final pose. We would like use human arm movement to indicate a few intermediate poses to smooth the UR5 trajectory.

Description

The main task is to let UR5 pick objects up, avoid obstacles, and place them at designated locations. The details of procedures are the following:

1. The user moves an arm in front of Kinect camera to teleoperate UR5. UR5 moveS around to construct an obstacle map of the environment using a R200 camera at the end-effector. It also detects the AR tag at the object target location.
2. The user moves the arm to specify some intermediate poses of UR5 for motion planning.
3. UR5 moves towards the object according to arm movement.
4. When gripper is close to the object, visual servoing takes over and picks up the object precisely.
5. UR5 moves the object to target location using motion planning with intermediate poses.

Software

There are a few new package/nodes to be implemented:

- Central controller
- Convert arm tracking poses to UR5 joint states.
- Hand-eye calibration (camera to end-effector)
- Motion planning with intermediate poses. For this package, we will use some core codes from OMPL [1].

These are the major existing packages that will be used:

- For UR5 control: universal_robot (tested), moveit (tested)
- For UR5 motion planning: moveit (tested), pcl, octomap [2]
- For skeleton tracking: openni_tracker [3], skeleton_markers [4]
- For visual servoing: visp_tracker [5]
- For target position localization and calibration: ar_track _alvar [6]

We need to test these packages unless indicated tested.

Hardware

The required hardware is the following:

- UR5 with desktop (provided)
- Gripper (provided)
- Microsoft Kinect camera (provided)
- Intel RealSense R200 camera (provided)
- Wooden cubes (provided)
- Foam obstacles (additional)
- Camera mount at end-effector (additional, 3D-printed)

We both have experiences using UR5 and the gripper for pick-and-place operation. We also have used R200 camera for hand-eye calibration in 601.636. We have not used Kinect before. We may use R200 for arm tracking instead after we test Kinect and R200.

Safety Plan

UR5 operation has some certain risks. It may swing around, especially during motion planning task. For teleoperation, the user should move the arm slowly so that UR5 moves at similar speed. For motion planning, soft obstacles will be used to mitigate damages to the gripper and UR5. Users should also stay away from UR5 during operation, and be ready to force abort the motion if necessary.

Project Timeline

Week 1: UR5 + Kinect

- Implement UR5 control with moveit.
- Implement arm tracking with Kinect.
- Write the node that converts arm poses to UR5 poses.

Week 2: UR5 + R200

- Design and install camera mount.
- Implement AR tag tracking and calibration with R200.
- Implement visual servoing with R200.

Week 3: UR5 Motion Planning

- Implement customized planner with intermediate pose input and core functionalities from OMPL.
- Integrate planner with moveit and octomap.

Week 4

- Integrate arm tracking, visual servoing, and motion planning

Week 5

- Testing and debugging.
- Record videos

References

[1] The Open Motion Planning Library
https://ompl.kavrakilab.org/

[2] OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees
https://octomap.github.io/

[3] OpenNI tracker package in ROS
http://wiki.ros.org/openni_tracker/

[4] Skeleton Markers package in ROS
http://wiki.ros.org/skeleton_markers/

[5] ViSP moving edge tracking wrapper package in ROS
http://wiki.ros.org/visp_tracker/

[6] Alvar wrapper package in ROS
http://wiki.ros.org/ar_track_alvar/