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

Yucheng Kang, Zhiyi Ren

530.707 Robot Systems Programming Independent Class Project, Spring 2018
Johns Hopkins University | Whiting School of Engineering | Baltimore, MD

Motivation

UR5 is a widely-used, 6DOF collaborative industrial robot arm. With a RGBD camera and gripper mounted, UR5 can sense the objects and perform pick-and-place tasks. However, when UR5 faces a cluster of unknown obstacles and objects, it may find it difficult to map the environment by itself. 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.

After the map is constructed, UR5 can now move small wooden cubes to the target location by detecting the visual markers. Existing motion planning algorithms can sufficiently help UR5 avoid obstacles.

Hardware and Infrastructure

Robot	UR5, 6 DOF collaborative industrial robot arm.
Gripper	Afag EU-20 UR gripper.
Camera	Microsoft Kinect V1: arm and hand gesture detection. Intel RealSense R200: obstacle mapping, visual marker detection.
Others	ArUco markers [1], 3D-printed camera mount, wooden cubes, vacuum-formed polystyrene obstacles

Existing Software

The project involves integrating a few existing software packages to use the Kinect camera, R200 camera, Octomap, Moveit [2], ArUco markers, and UR5 arm. Major packages include

1. **universal_robot** (wiki.ros.org/universal_robot) drivers, descriptions, and utilities for UR5 in ROS.
2. **ur5_modern_driver** (github.com/ThomasTimm/ur_modern_driver) new driver for UR5 in ROS, improved usability and ros_control compatibility.
3. **aruco_ros** (wiki.ros.org/aruco_ros) real-time 3D pose estimation of ArUco markers in ROS.
   - Since the camera is moving, sometimes it does not see all the markers, but we want Moveit to know where the markers are throughout the mapping process. Thus we implemented the node to keep publishing the most recent available transform from UR5 base_link to the markers.
   - Sometimes the output from ArUco libraries is not accurate. We implemented a simple filter to select poses that has vertical Z-axis.
4. **aruco_hand_eye** (github.com/jhu-lcsr/aruco_hand_eye) R200 hand-eye calibration using aruco_ros and ViSP.
5. **octomap_mapping** (wiki.ros.org/octomap_mapping) 3D occupancy grid mapping using Octrees.
   - We found that updating octomap in Rviz using Moveit native plugin in real-time is very CPU/RAM-consuming. Rviz is very sluggish and even crashes if user attempts to use Rviz. To solve the issue, we let the octomap server alone constructs the map, and manually load the map into Moveit only once after the mapping process.
6. **octomap_ros** (wiki.ros.org/octomap_ros) conversion between ROS and Octomap native types.
7. **openni_camera** (wiki.ros.org/openni_camera) ROS driver for Kinect.
8. **openni_tracker** (wiki.ros.org/openni_tracker) OpenNI skeleton tracking with tf broadcaster.
9. **vision_opencv** (wiki.ros.org/vision_opencv) ROS interface for OpenCV library.
10. **moveit_ros** (wiki.ros.org/moveit_ros) Integration of Moveit with ROS.
11. **actionlib** (wiki.ros.org/actionlib) standardized ROS interface for preemptable tasks.

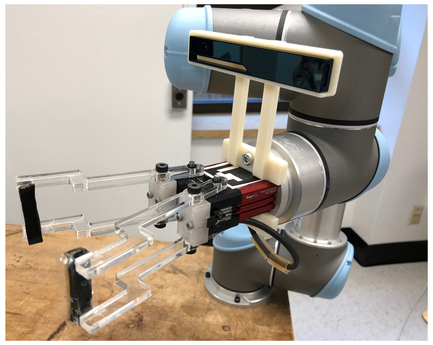
New Software

To implement arm and hand gesture detection, we implemented following software:

1. **kinect_teleop** Teleoperation with Kinect
   - filter the user's body-to-hand transform given by skeleton tracking
   - convert this transform to the target pose of UR5 end-effector
   - recognize the user's gestures and then feed the results to a simple FSM as inputs

To implement UR5 control and Moveit motion planning, we implemented following software:

1. **central** central controller of the system
   - start and end mapping/pick-and-place tasks based on user input and system feedback.
   - calculate UR5 end-effector target locations based on ArUco marker inputs.
   - use a customized action client-server interface ur5_planning for motion request.
2. **ur5_planning** customized interface with Moveit
   - add table, gripper, camera mount, and cube as obstacles into planning scene
   - Request RRT-Connect or cartesian path planning



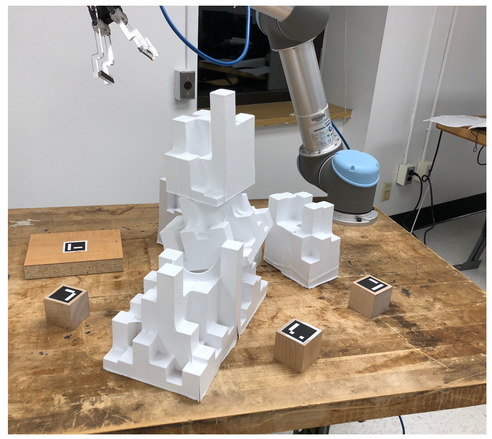


Figure 1—Hardware
UR5 with gripper and R200 camera, obstacles, ArUco markers, and objects.

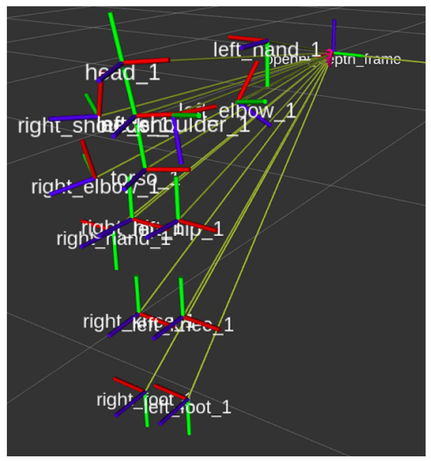
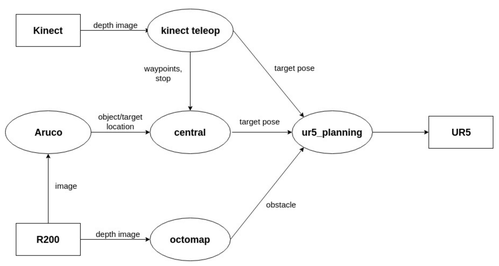


Figure 2—Kinect teleoperation
Gesture recognition and skeleton tracking.



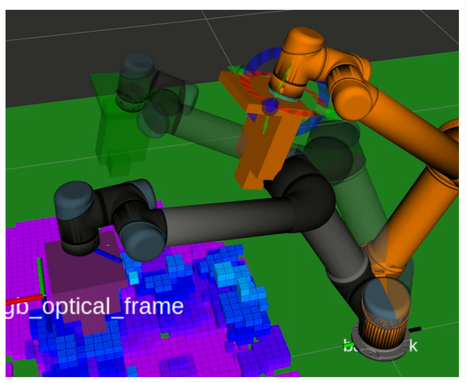


Figure 3—Overall Framework, motion planning with Moveit
Octomap with marker frames and motion planning with attached cube for collision checking.

Work Flow

1. User calibrates skeleton tracking in front of Kinect camera.
2. User makes a "START" gesture to tell UR5 to start following the arm and mapping.
3. User moves an arm around, and UR5 follows to construct an obstacle map of the environment. It also detects the ArUco tags at objects and target location.
4. User makes a "STOP" gesture to tell UR5 to stop following the arm and start pick-and-place.
5. Moveit loads the constructed map from octomap server.
6. Moveit plans and executes UR5 to pick-and-place objects. Use cartesian path when approaching the object.

Lessons Learned

- Moveit framework is highly complicated, and it provides both easy and advanced interface APIs. We started with the advanced one, but soon realized that many of the features are unnecessary. We then restarted with the easy one, and implemented advanced features if needed.
- Real-time updating Octomap with Moveit plugin updater in Rviz is very CPU/RAM-consuming.
- rqt_launchtree can be used to analyze large launch files. It helped us better understand and quickly look up the structure of the Moveit framework.
- Realsense R200 camera has a minimum range of ~0.5m for depth image, so it is important to keep it away from obstacles while constructing octomap.
- Hand segmentation can be achieved in both depth images and RGB images. However, in depth images, the user's hand must be close to Kinect due to the low resolution; in RGB images, the color threshold is somewhat sensitive to the lighting.

Future Work

- **Intermediate pose** Existing motion planning algorithms may cause UR5 to move to a few awkward poses before reaching final pose. We would like use hand gesture to input a few intermediate poses into Moveit. We would like to implement a customized probabilistic roadmap planner using these intermediate poses as samples.
- **Visual servoing** Existing related packages such as ViSP [3] can use RGB image to guide robot arm to the objects without the use of visual markers.
- **Predictive control [4]** Published research has implemented model predictive control of UR5 for online generation of optimal trajectories matching user arm position and orientation. This enables UR5 to follow the arm in higher frequency, low latency, and with a smoother trajectory.

References and Acknowledgements

We would like to acknowledge the instructor, Dr. Louis Whitcomb, and the TAs, Tyler Paine and Soham Patkar, for the valuable suggestions. We also would like to thank Dr. Noah Cowan for the use of the gripper.

1. ArUco: a minimal library for Augmented Reality applications based on OpenCV, www.uco.es/investiga/grupos/ava/node/26
2. Moveit! Motion Planning Framework, docs.ros.org/kinetic/api/moveit_tutorials.
3. ViSP: Visual Servoing Platform, visp.inria.fr.
4. Real-Time Predictive Control of an UR5 Robotic Arm Through Human Upper Limb Motion Tracking, Omarali et al, 2017, HRI'17.