Technical Report

IRI-TR-14-04

Rigid and deformable pick and place algorithms

Felip Martí
Guillem Alenyà

August, 2014

Institut de Robòtica i Informàtica Industrial

Abstract

This technical report explains the packages used in the WAM robot to pick and place cloth or tableware objects. The goal was to check if the WAM arm robot could perform several movements to fold and unfold deformables and manipulate some tableware objects. Eight different nodes have been implemented following a generic and a modular design in order to allow scalability and adaptability.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain

Tel (fax): +34 93 401 5750 (5751)
`http://www.iri.upc.edu`

Corresponding author:

F Marti
tel: +34 93 405 4490
`fmart@iri.upc.edu`
`http://www.iri.upc.edu/staff/fmart`

1 Introduction

1.1 Objectives

The goal of this report is to check if the WAM arm robot can perform several movements to manipulate deformables, to verify if the robot is physically able to fold and unfold a piece of cloth.

Moreover, we want also check if the robotic arm can physically manipulate some tableware objects like plates or cups, to test if the WAM is able to stuck up them, and then move the stack to another position.

1.2 Used technologies

The implementation has been done using the following technologies:

- Kinect
- WAM Arm
- ROS framework

(a) Kinect (b) WAM Arm (c) ROS framework

Figure 1: Used technologies

2 Packages description

Eight different ROS packages has been implemented in order to obtain a generic and a modular design. Two of the packages are generic algorithms to perform the pick and place action. Three of the packages are specific for the perception and manipulation of deformables, and the other three for tableware objects.

2.1 iri_wam_generic_pickorplace

This is the most generic node implemented. `iri_wam_generic_pickorplace` is a server actionlib that just performs the pick or the place action.

Dependencies

This node has the following IRI dependencies:

- **Inverse Kinematics:** iri_wam_common_msgs/wamInverseKinematics.srv
- **WAM Joints Movement:** iri_common_drivers_msgs/QueryJointsMovement.srv
- **Open Gripper:** iri_common_drivers_msgs/tool_open.action
- **Close Gripper:** iri_common_drivers_msgs/tool_close.action

Action Specification

To perform a generic pick or place action the following Goal has to be defined:

- **ini_point** Vector of 3 floats (X, Y, Z) with the desired PreGrasp or PreUngrasp position. The reference frame is the base of the robot.
- **grasp_point** Vector of 3 floats with the desired Grasp or Ungrasp position.
- **end_point** Vector of 3 floats with the desired PostGrasp or PostUngrasp position.
- **ini_EF_rpy** Vector of 3 floats with the desired end effector rotations. (Roll, Pitch, Yaw) in radians for the Grasp or Ungrasp position [1]. The reference frame is the base of the robot.
- **end_EF_rpy** Vector of 3 floats with the desired end effector rotations (Roll, Pitch, Yaw) in radians for the PostGrasp or PostUngrasp position.
- **pick** Boolean to select if it is a pick or place action. The only difference is when the algorithm has to open or close the end effector.
- **execute** Boolean to select if we want to move the robot (execute=true) or just calculate the Inverse Kinematics to know if the pick or place action is possible.

The actionlib server also provides as a feedback the pick or place state machine internal variable. Besides, once the action is finished a result is sent upon completion of the goal.

2.2 iri_wam_generic_pickandplace

This node is an actionlib that calls twice `iri_wam_generic_pickorplace` to perform a pick and a place action.

Dependencies

This node dependends on `iri_wam_generic_pickorplace` package.

Action Specification

To perform a generic pick and place action the following Goal has to be defined:

- **pregrasp_point** Vector of 3 floats (X, Y, Z) with the desired PreGrasp position. The reference frame is the base of the robot.
- **grasp_point** Vector of 3 floats with the desired Grasp position.
- **postgrasp_point** Vector of 3 floats with the desired PostGrasp position.
- **ini_grasp_EF_rpy** Vector of 3 floats with the desired end effector rotations (Roll, Pitch, Yaw) in radians for the Grasp position [2]. The reference frame is the base of the robot.
- **end_grasp_EF_rpy** Vector of 3 floats with the desired end effector rotations in radians for the PostGrasp position.
- **preungrasp_point** Vector of 3 floats with the desired PreUngrasp position.
- **ungrasp_point** Vector of 3 floats with the desired Ungrasp position.
- **postungrasp_point** Vector of 3 floats with the desired PostUngrasp position.
- **ini_ungrasp_EF_rpy** Vector of 3 floats with the desired end effector rotations in radians for the Ungrasp position.
- **end_ungrasp_EF_rpy** Vector of 3 floats with the desired end effector rotation in radians for the PostUngrasp position.

[1] ROS Postmultiplicate Yaw, Pitch, Roll
[2] ROS Postmultiplicate Yaw, Pitch, Roll

The actionlib server provides as a feedback the pick and place state machine internal variable and also the state machine pick or place variable. Besides, once the action is finished a result is sent upon completion of the goal.

2.3 iri_pickandplace_deformable

This package is an actionlib that calls `iri_wam_generic_pickandplace` to do the specific action of pick and place deformables. Instead of defining 10 different parameters in this case only three are needed.

Dependencies

This node has the following dependencies:

- **WAM Joints Movement:** iri_common_drivers_msgs/QueryJointsMovement.srv
- **Generic pick and place node:** `iri_wam_generic_pickandplace`
- **Calibration Camera-Robot:** This node transforms the reference frame of several points from the camera to the robot. The calibration is set in the camera roslaunch.

Action Specification

The pick approach is done in a vertical direction. Once the cloth is grasped, the robot moves to a middle point, between pick and place, following a diagonal trajectory. The place approach is also done following a diagonal trajectory. Finally, when the cloth is placed, the robot leaves that position following a vertical direction.

Due to the specification of this node only 3 parameters have to be defined.

- **pick_pos:** Vector of 3 floats (X, Y, Z) with the desired Pick position. The reference frame is the camera rgb base link.
- **place_pos:** Vector of 3 floats with the desired Place position. The reference frame is the camera rgb base link.
- **Yaw:** orientation of the end effector in radians. The reference frame is the robot base link.

The actionlib server provides as a feedback the pick and place deformable state machine internal variable, the state machine pick and place variable, and also the state machine pick or place variable. Besides, once the action is finished a result is sent upon completion of the goal.

2.4 iri_pickandplace_tableware

This package is an actionlib that calls `iri_wam_generic_pickandplace` to do the specific action of pick and place tableware. Instead of defining 10 different parameters in this case only four are needed.

Dependencies

This node has the following dependencies:

- **WAM Joints Movement:** iri_common_drivers_msgs/QueryJointsMovement.srv
- **Generic pick and place node:** `iri_wam_generic_pickandplace`
- **Calibration Camera-Robot:** This node transforms the reference frame of several points from the camera to the robot. The calibration is set in the camera roslaunch.

Action Specification

In contrast with `iri_pickandplace_tableware`, the transition from post-pick and pre-place positions is done following an horizontal trajectory.

Due to the specification of this node only 4 parameters have to be defined.

- **pick_pos:** Vector of 3 floats (X, Y, Z) with the desired Pick position. The reference frame is the camera rgb base link.
- **place_pos:** Vector of 3 floats with the desired Place position. The reference frame is the camera rgb base link.
- **centroid_pos:** Vector of 3 floats with the centroid position of the object. The centroid is used to calculate the orientation of the end effector. The reference frame is the camera rgb base link.
- **object_type:** Char to indicate which kind of object is going to pick the robot. This is used also to calculate the orientation of the end effector. Plate = 'p', Glass = 'g'

The actionlib server provides as a feedback the pick and place tableware state machine internal variable, the state machine pick and place variable, and also the state machine pick or place variable. Besides, once the action is finished a result is sent upon completion of the goal.

2.5 iri_color_interesting_points_deformable

A cloth with eight different colored corners is used to facilitate the obtaining of cloth interest points. This package is a service server that process the image captured with a kinect and return all the interested points found in the image.

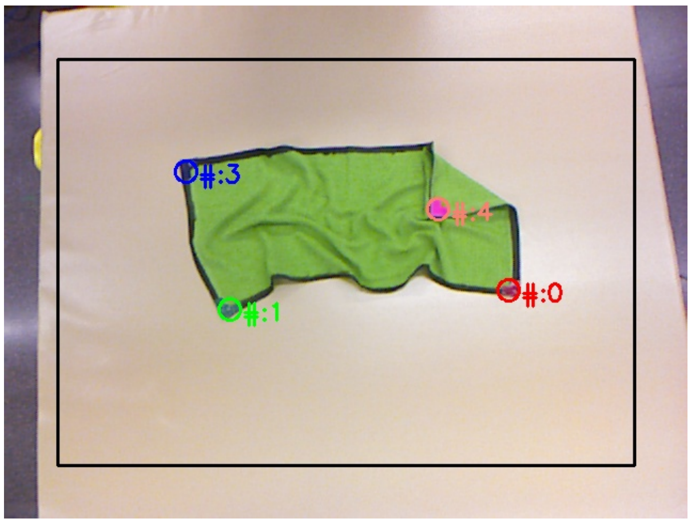


Figure 2: Deformable used with its interest points

Dependencies

This package requires a kinect, the kinect driver could be OpenNI. [3]

[3]http://wiki.ros.org/openni_camera

Server Specification

The service call request takes no arguments, and the response returns the following data:

- **Color:** Vector of integers with the color ID detected.
- **U:** Vector of integers with the U position in the image.
- **V:** Vector of integers with the V position in the image.
- **X:** Vector of floats with the X position in the scene. The reference frame is the camera rgb base link.
- **Y:** Vector of floats with the Y position in the scene. The reference frame is the camera rgb base link.
- **Z:** Vector of floats with the Z position in the scene. The reference frame is the camera rgb base link.
- **object_type** Vector of chars that indicates the type of object. In this case always will be 'C' meaning centroid
- **amount** Integer with the number of different points obtained.

2.6 iri_color_interesting_points_tableware

A dark blue cup and an orange plate are used to facilitate the identification and the obtaining of tableware interest points. This package is a service server that process the image captured with a kinect and returns four grasping points for each object found in the image.

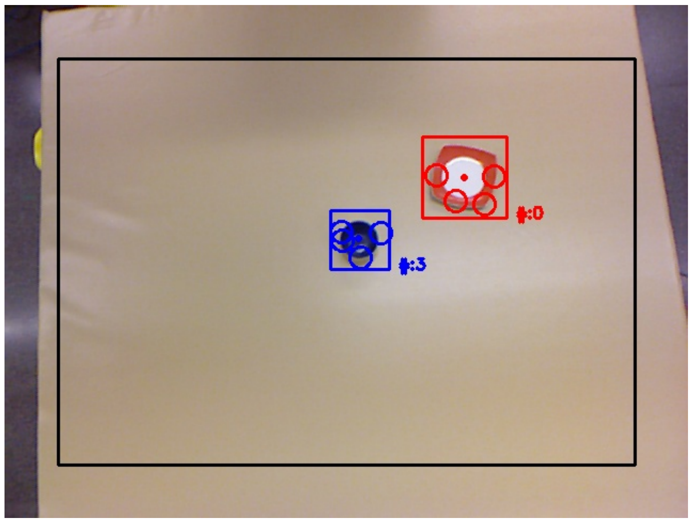


Figure 3: Tableware used with their interest points

Dependencies

This package requires a kinect, the kinect driver could be OpenNI.

Server Specification

The service call request takes no arguments, and the response returns the following data:

- **Color:** Vector of integers with the ID color detected.
- **U:** Vector of integers with the U position in the image.
- **V:** Vector of integers with the V position in the image.

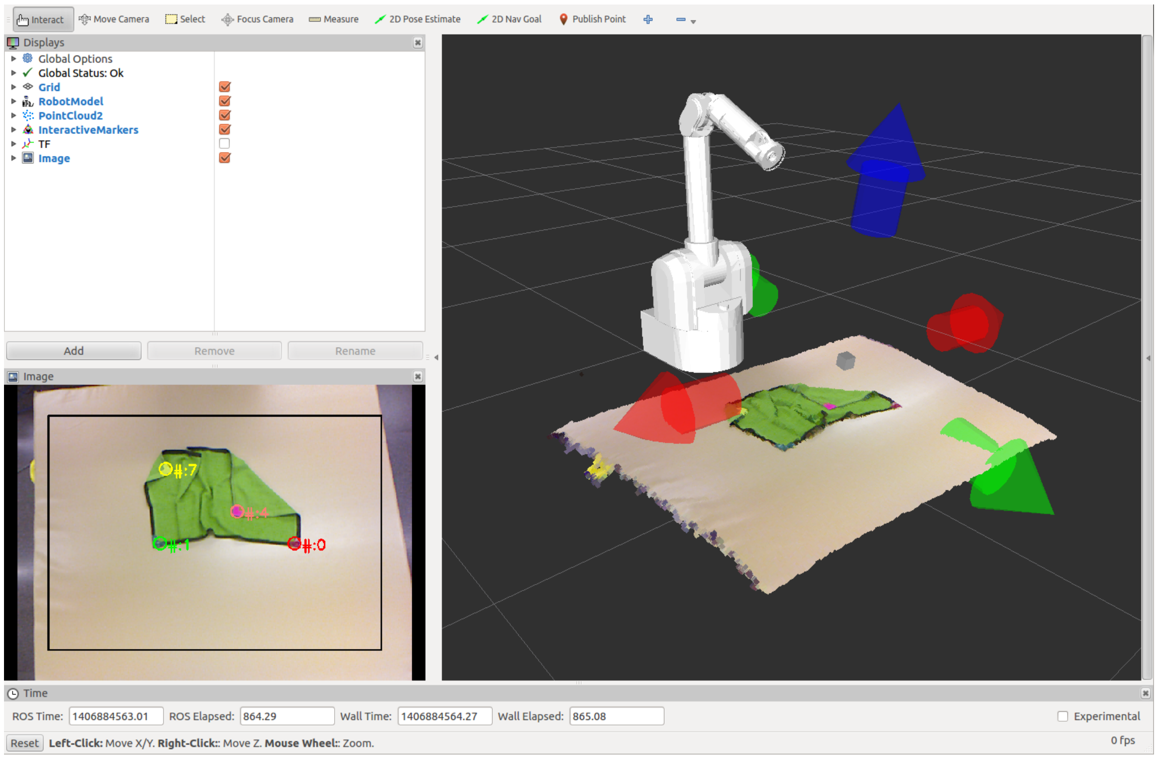
- **X:** Vector of floats with the X position in the scene. The reference frame is the camera rgb base link.
- **Y:** Vector of floats with the Y position in the scene. The reference frame is the camera rgb base link.
- **Z:** Vector of floats with the Z position in the scene. The reference frame is the camera rgb base link.
- **object_type** Vector of chars that indicates the type of object. Plate = 'p', Glass = 'g', and Centroid = 'C'
- **amount** Integer with the number of different points obtained.

2.7 iri_human_decision_maker_deformable

The interaction between the operator and the robot is done in this node.

First of all `iri_color_interesting_points_deformable` service is called to obtain interest points. Then the operator enter which point pick and where to place. The pick selection is done through the Linux console, and the place selection using an interactive marker through rviz.

Finally, `iri_pickandplace_deformable` is called to perform the desired pick and place.



ROS Time: 1406884563.01 ROS Elapsed: 864.29 Wall Time: 1406884564.27 Wall Elapsed: 865.08

Left-Click: Move X/Y. **Right-Click:**: Move Z. **Mouse Wheel:**: Zoom.

Figure 4: Rviz with the interactive marker to select the cloth corner desired position

Dependencies

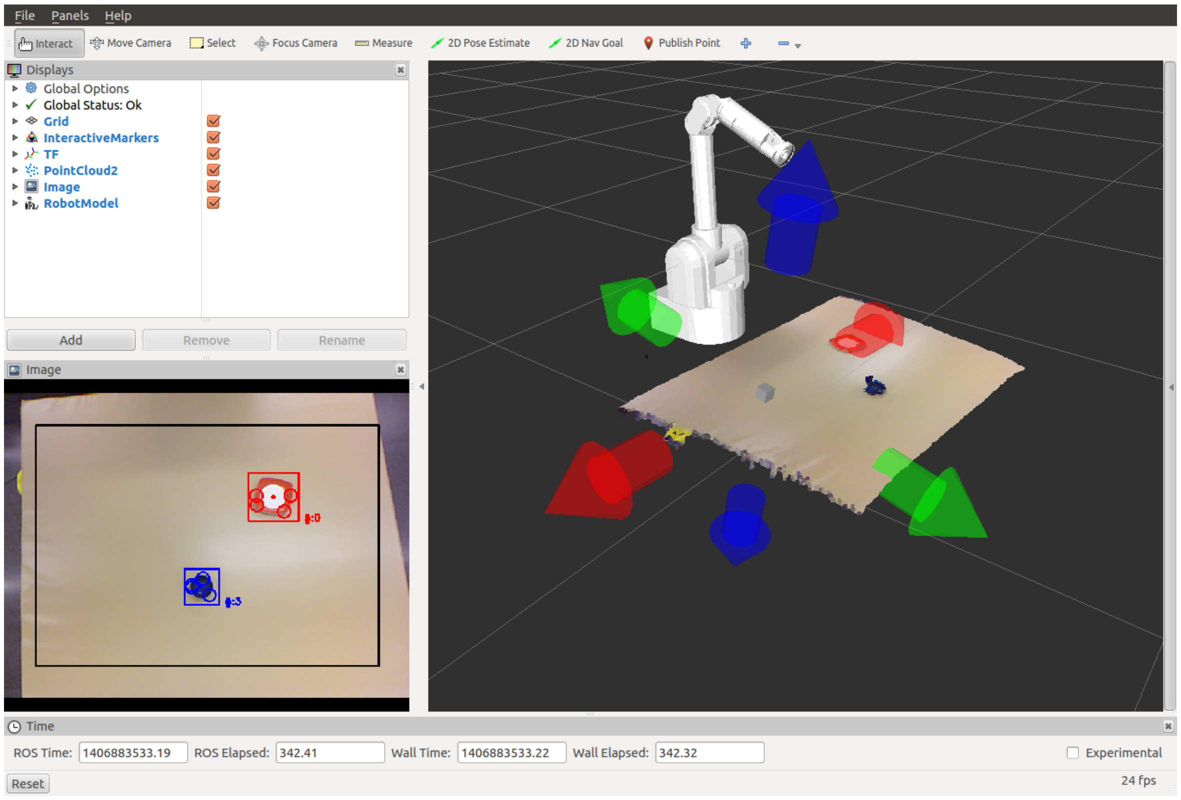
- `iri_color_interesting_points_deformable`
- `iri_pickandplace_deformable`

2.8 iri_human_decision_maker_tableware

The interaction between the operator and the robot is done in this node.

First of all `iri_color_interesting_points_tableware` service is called to obtain interest points. Then the operator enter which point pick and where to place. The pick selection is done through the Linux console, and the place selection using an interactive marker through rviz.

Finally, `iri_pickandplace_tableware` is called to perform the desired pick and place.



ROS Time: 1406883533.19 ROS Elapsed: 342.41 Wall Time: 1406883533.22 Wall Elapsed: 342.32

Figure 5: Rviz with the interactive marker to select the tableware object desired position

Dependencies

- `iri_color_interesting_points_tableware`
- `iri_pickandplace_tableware`

3 Nodes Structure

3.1 Pick and Place Deformable

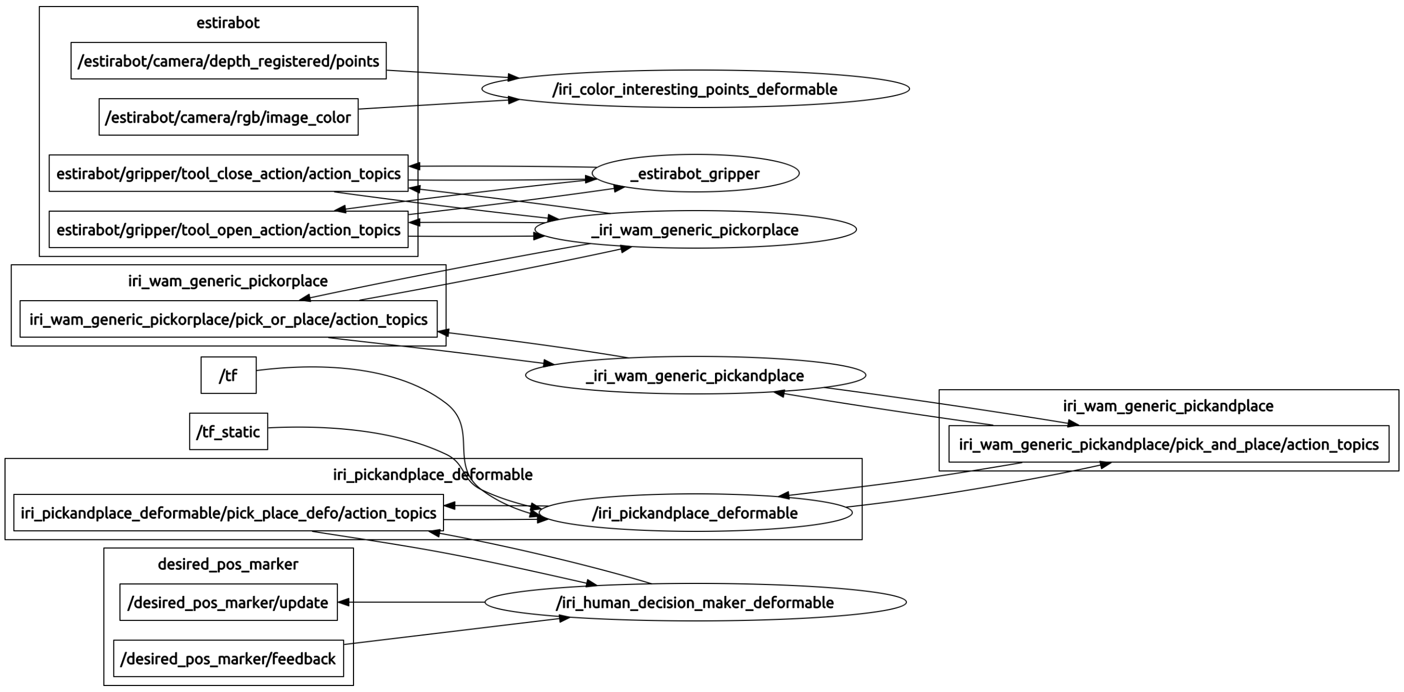

estirabot
/estirabot/camera/depth_registered/points
/estirabot/camera/rgb/image_color
estirabot/gripper/tool_close_action/action_topics
estirabot/gripper/tool_open_action/action_topics
/iri_color_interesting_points_deformable
_estirabot_gripper
_iri_wam_generic_pickorplace
iri_wam_generic_pickorplace
iri_wam_generic_pickorplace/pick_or_place/action_topics
/tf
/tf_static
_iri_wam_generic_pickandplace
iri_wam_generic_pickandplace
iri_wam_generic_pickandplace/pick_and_place/action_topics
iri_pickandplace_deformable
iri_pickandplace_deformable/pick_place_defo/action_topics
/iri_pickandplace_deformable
desired_pos_marker
/desired_pos_marker/update
/desired_pos_marker/feedback
/iri_human_decision_maker_deformable

Figure 6: Pick and Place Deformable Topics and Nodes

3.2 Pick and Place Tableware

The node structure for pick and place tableware is exactly the same than for pick and place deformable, the unique difference is the nodes name.

4 Sequence diagram

4.1 Pick and Place Deformable

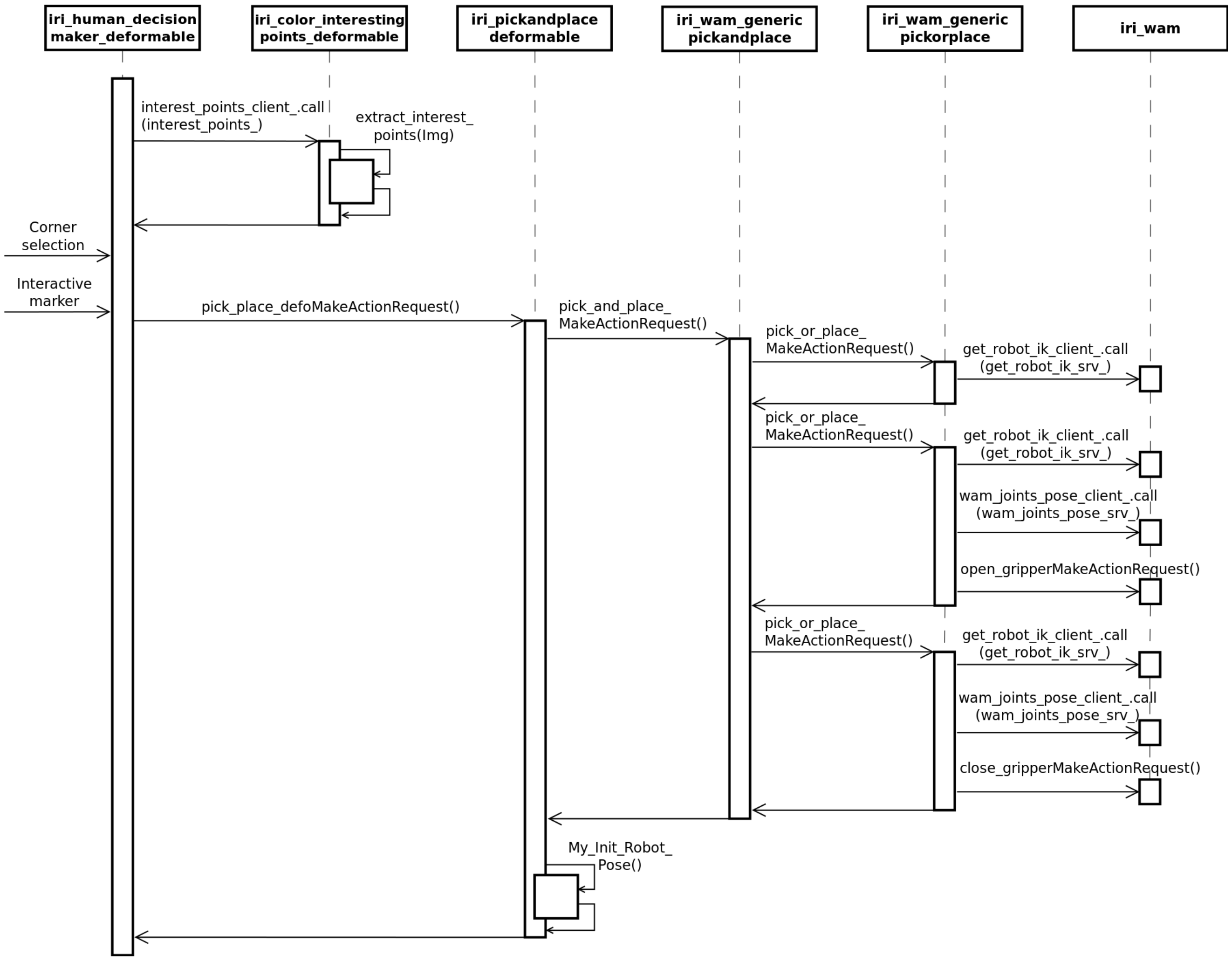

iri_human_decision maker_deformable
iri_color_interesting points_deformable
iri_pickandplace deformable
iri_wam_generic pickandplace
iri_wam_generic pickorplace
iri_wam
interest_points_client_.call (interest_points_)
extract_interest_ points(Img)
Corner selection
Interactive marker
pick_place_defoMakeActionRequest()
pick_and_place_ MakeActionRequest()
pick_or_place_ MakeActionRequest()
get_robot_ik_client_.call (get_robot_ik_srv_)
pick_or_place_ MakeActionRequest()
get_robot_ik_client_.call (get_robot_ik_srv_)
wam_joints_pose_client_.call (wam_joints_pose_srv_)
open_gripperMakeActionRequest()
pick_or_place_ MakeActionRequest()
get_robot_ik_client_.call (get_robot_ik_srv_)
wam_joints_pose_client_.call (wam_joints_pose_srv_)
close_gripperMakeActionRequest()
My_Init_Robot_ Pose()

Figure 7: Pick and Place Deformable Sequence diagram

4.2 Pick and Place Tableware

The Sequence diagram for pick and place tableware is exactly the same than for pick and place deformable, the unique difference is the functions name.

5 How to use

In this chapter is explained how to install and run the packages.

5.1 Installation

External dependencies

These algorithms have been implemented in ROS[4]. Therefore, make sure you have installed at least ROS Hydro Medusa[5] in your computer.

Furthermore, kinect drivers are also needed. A good one is OpenNI that has a ROS driver named openni_camera[6]

Internal IRI dependencies

Make sure that you have created the catkin workspace and instaled all the IRI dependencies [7]

Downloading packages

All the packages are in the IRI repository[8]. So, if your workspace is well configured you will find all the packages typing:

```
$ cd ~/iri-lab/iri_ws/src/estirabot_robot
```

Otherwise, all the packages can be downloaded from github typing:

```
$ git clone https://github.com/FelipMarti/iri_wam_generic_pickorplace.git
$ git clone https://github.com/FelipMarti/iri_wam_generic_pickandplace.git
$ git clone https://github.com/FelipMarti/iri_pickandplace_deformable.git
$ git clone https://github.com/FelipMarti/iri_pickandplace_tableware.git
$ git clone https://github.com/FelipMarti/iri_human_decision_maker_deformable.git
$ git clone https://github.com/FelipMarti/iri_human_decision_maker_tableware.git
$ git clone https://github.com/FelipMarti/iri_color_interesting_points_deformable.git
$ git clone https://github.com/FelipMarti/iri_color_interesting_points_tableware.git
```

Compiling packages

All the packages have to be compiled, to compile all of them type in your catkin workspace:

```
$ catkin_make
```

or to compile a package and dependencies separately type:

```
$ catkin_make --only-pkg-with-deps iri_name_of_package
```

5.2 Running nodes

Using packages to pick and place deformables

Open 1 terminal (1st one), connect through ssh to bawse and execute a roscore

```
$ ssh robot@bawse
$ roscore
```

[4]http://www.ros.org/
[5]http://wiki.ros.org/hydro
[6]http://wiki.ros.org/openni_camera
[7]http://wiki.iri.upc.edu/index.php/LabRobotica_ROS_Software_Installation
[8]https://devel.iri.upc.edu/pub/labrobotica/ros/iri-ros-pkg_hydro/metapackages/estirabot_robot/

Open another terminal (2nd one), connect through ssh to bawse and launch the gripper

```
$ ssh robot@bawse
$ roslaunch iri_wam_bringup iri_wam_bringup_gripper_nc.launch ROBOT:=estirabot
```

Open another terminal (3rd one), connect through ssh to bawse and launch the robot

```
$ ssh robot@bawse
$ roslaunch iri_wam_bringup estirabot.launch
```

Open another terminal (4th one), set the ROS variable ROS_MASTER_URI and launch the kinect with the perception node `iri_color_interesting_points_deformable`

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_color_interesting_points_deformable \
iri_color_interesting_points_deformable_and_CAM.launch
```

Open another terminal (5th one), set the ROS variable ROS_MASTER_URI and launch the pick and place node `iri_pickandplace_deformable`

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_pickandplace_deformable iri_pickandplace_deformable.launch
```

Open another terminal (6th one), set the ROS variable ROS_MASTER_URI and launch the decision maker node `iri_human_decision_maker_deformable`

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_human_decision_maker_deformable \
iri_human_decision_maker_deformable.launch
```

Open another terminal (7th, last but not least), set the ROS variable ROS_MASTER_URI and run `rviz` to position the interactive marker

```
$ ROS_MASTER_URI="http://bawse:11311"
$ rosrun rviz rviz
```

Using packages to pick and place tableware

Open 1 terminal (1st one), connect through ssh to bawse and execute a roscore

```
$ ssh robot@bawse
$ roscore
```

Open another terminal (2nd one), connect through ssh to bawse and launch the gripper

```
$ ssh robot@bawse
$ roslaunch iri_wam_bringup iri_wam_bringup_gripper_nc.launch ROBOT:=estirabot
```

Open another terminal (3rd one), connect through ssh to bawse and launch the robot

```
$ ssh robot@bawse
$ roslaunch iri_wam_bringup estirabot.launch
```

Open another terminal (4th one), set the ROS variable ROS_MASTER_URI and launch the kinect with the perception node `iri_color_interesting_points_tableware`

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_color_interesting_points_tableware \
iri_color_interesting_points_tableware_and_CAM.launch
```

Open another terminal (5th one), set the ROS variable ROS_MASTER_URI and launch the pick and place node `iri_pickandplace_tableware`

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_pickandplace_tableware iri_pickandplace_tableware.launch
```

Open another terminal (6th one), set the ROS variable ROS_MASTER_URI and launch the decision maker node `iri_human_decision_maker_tableware`

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_human_decision_maker_tableware \
iri_human_decision_maker_tableware.launch
```

Open another terminal (7th, last but not least), set the ROS variable ROS_MASTER_URI and run `rviz` to position the interactive marker

```
$ ROS_MASTER_URI="http://bawse:11311"
$ rosrun rviz rviz
```

Using generic pick and place algorithms

Generic algorithms don't require the camera neither the decision maker algortihms. The actionlib could be checked using an axclient.py

Open 1 terminal (1st one), connect through ssh to bawse and execute a roscore

```
$ ssh robot@bawse
$ roscore
```

Open another terminal (2nd one), connect through ssh to bawse and launch the gripper

```
$ ssh robot@bawse
$ roslaunch iri_wam_bringup iri_wam_bringup_gripper_nc.launch ROBOT:=estirabot
```

Open another terminal (3rd one), connect through ssh to bawse and launch the robot

```
$ ssh robot@bawse
$ roslaunch iri_wam_bringup estirabot.launch
```

Open another terminal (4th one), set the ROS variable ROS_MASTER_URI and launch the generic algorithm.

```
$ ROS_MASTER_URI="http://bawse:11311"
$ roslaunch iri_wam_generic_pickandplace iri_wam_generic_pickandplace.launch
```
or
```
$ roslaunch iri_wam_generic_pickorplace iri_wam_generic_pickorplace.launch
```

Open another terminal (5th one), set the ROS variable ROS_MASTER_URI and run the actionlib axclient.py

```
$ ROS_MASTER_URI="http://bawse:11311"
$ rosrun actionlib axclient.py iri_wam_generic_pickandplace/pick_and_place
```
or
```
$ rosrun actionlib axclient.py iri_wam_generic_pickorplace/pick_or_place
```

5.3 How to expand

Adding new color

Colors are detected using HSV representation. Therefore, high and low Hue, Saturation, and Value are needed to detect colors. If the image is obtained through a kinect these parameters can be easily obtained using the following program:
`https://github.com/FelipMarti/ColorDetection_kinect.git`

Colors to be detected are stored in `MyColors` variable of the `extract_interest_points()` function that is located in `color_interesting_points_deformable_alg.cpp` file. In case of tableware node, the function is located in `color_interesting_points_tableware_alg.cpp`

Adding new tableware object

Tableware is identified by color. Therefore, first of all a new color has to be added.

Secondly, `iri_color_interesting_points_tableware` service has to return what kind of object is in the scene, so, a new char has to be assigned.

Finally, in `iri_pickandplace_tableware` actionlib a new grasping configuration for the new object has to be implemented to perform the pick and place action.

6 Conclusions/Results

The pick and place algorithms perform the action in a open-loop control system. Therefore, the main problem encountered has been to find a very good camera calibration because the robot doesn't correct the desired pick and place positions. Despite of the calibration problems, the robot has been able to perform several movements to fold and unfold a cloth.

Due to the end effector lack of force and the weight of some objects, some troubles have appeared with the manipulation of tableware. The Wam robot can perform several pick and place actions of a single object. However, once we have a stack of two objects the end effector doesn't have force enough to move the stack.