Robotic Reaching of Objects Based on Verbal Description and Human Gesture in MyGym Simulator

Egor Sarana, 2023

myGym

myGym is a toolkit suitable for the fast prototyping of neural networks in the area of robotic manipulation and navigation.

This work aims for the implementation of Reach gesture task that could be defined using natural language or human gesture.

A basic Reach task is a task where the goal is to minimize the Euclidean distance between the robot's gripper and the desired object.

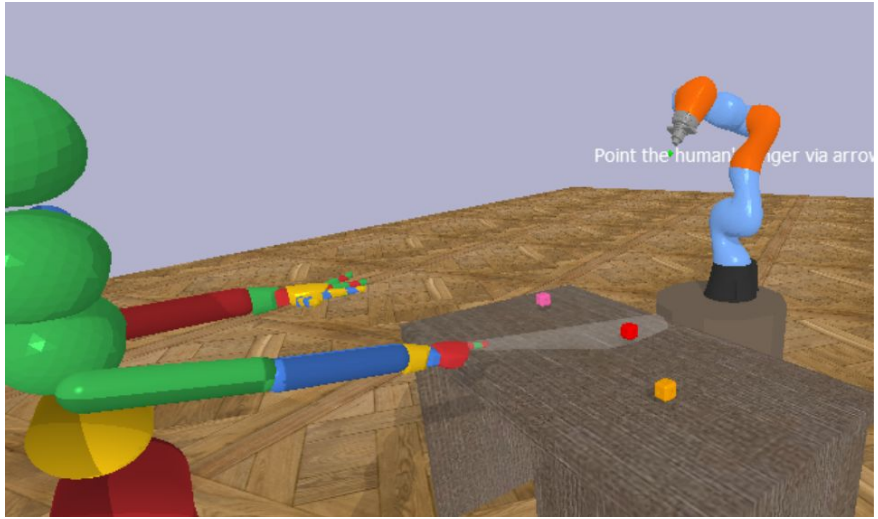
In order to implement that task, the modules described on the next slides are required.

Human module

A Python module that controls a human. It is capable of loading the URDF model and pointing to the desired object with inverse kinematics.

The corresponding human URDF model was updated. Some joints were physically fixed, and some were completely removed for natural pointing. A transparent cone was added to indicate the direction of pointing.

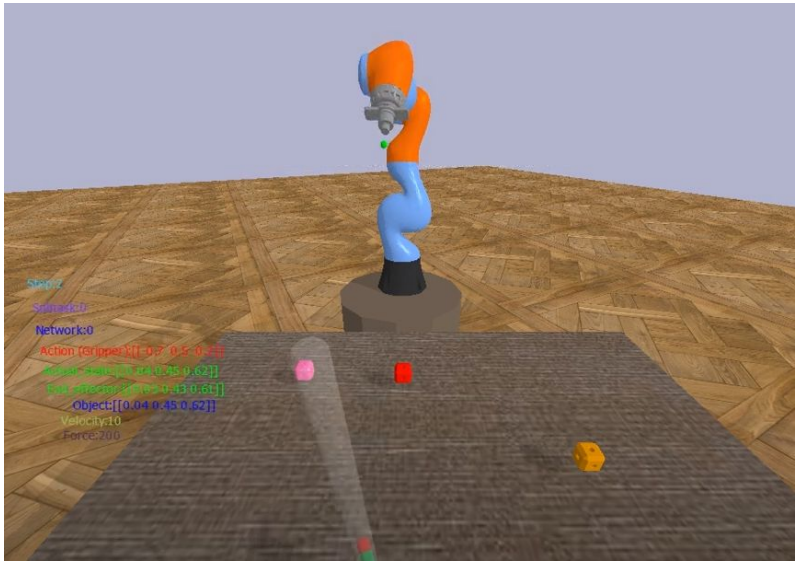
The human module also contains the function for computing the nearest object to the pointing direction.



```
points -= p2.reshape(1, -1)
# move points relative to the vector's beginning (p2)
scalars = np.dot(points, vector)
# scalar product (as a part of computing projections)
points_proj = scalars.reshape(-1, 1) * vector.reshape(1, -1)
# projections on the vector
points_rej = points - points_proj  # rejections
distances = np.linalg.norm(points_rej, axis=1)
return objects[np.argmin(distances)]
```

Reach gesture

Reach gesture task is represented by changes in gym_env.py, train.py, natural_language.py. During training, the human points in a random direction, and depending on that direction, the nearest object is determined. During testing, it is possible to select the goal object manually. When the object is selected, it is just default Reach task.



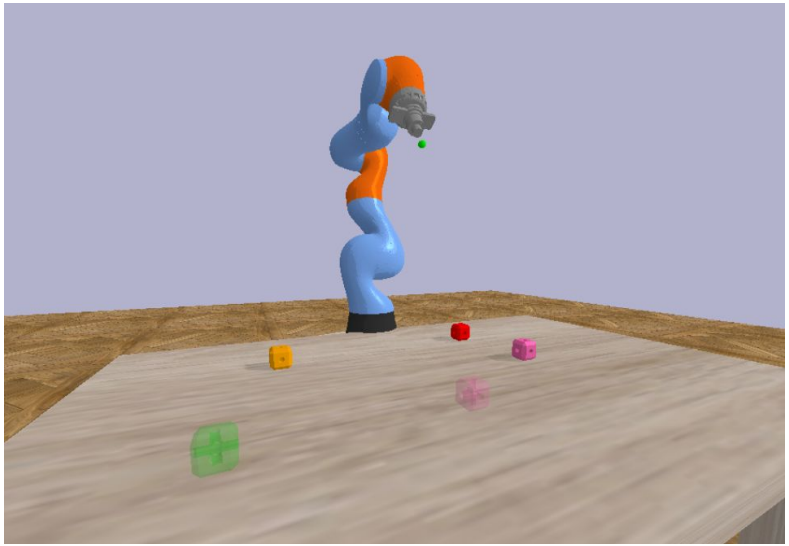

```
if self.p.B3G_LEFT_ARROW in key_press.keys()
    and key_press[self.p.B3G_LEFT_ARROW] == 3:
    self.human.point_finger_at(
        move_factor * np.array([0.01, 0, 0]), relative=True)
```

Color system

In order to implement the natural language module in the future, the color system has been updated. From now on, the color names are used instead of RGBA values, i.e., "green" instead of (0, 1, 0, 0). Moreover, there were added "new colors". Every color has a transparent version, which is illustrated by the picture. Transparent objects are mainly used to indicate a goal for Pick and Place type tasks. In other words, it is used when the final object is "fake," not real, and used just as a dummy object.

There are also some useful functions in the colors.py module for drawing random colors, excluding some, etc.

To allow random drawing for real and goal "dummy" objects, training configuration files were updated. From now, it is enough to define two lists from which the objects will be drawn randomly.

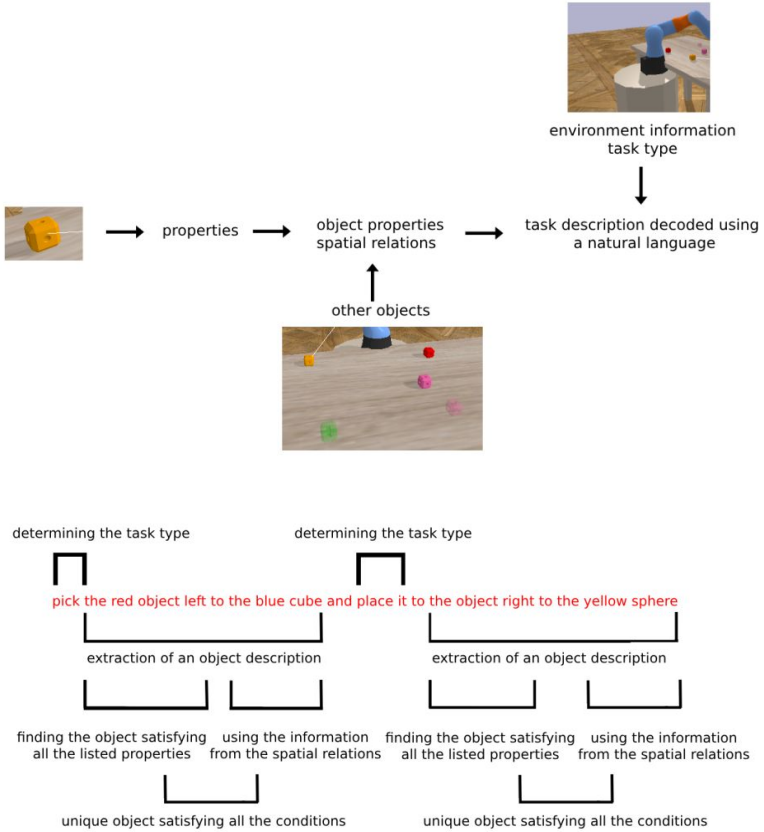


```
#Task
"task_type"            :"pnp",
"natural_language_mode" :1,
"task_objects"         :{"init": [
                       {"obj_name": "cube_holes", "fixed": 0, "rand_rot": 0, "sampling_area":[-0.6, -0.1, 0.2, 0.4, 0.075, 0.075]},
                       {"obj_name": "cube_holes", "fixed": 0, "rand_rot": 0, "sampling_area":[-0.3, 0.3, 0.5, 0.6, 0.075, 0.075]},
                       {"obj_name": "cube_holes", "fixed": 0, "rand_rot": 0, "sampling_area":[0.1, 0.6, 0.2, 0.4, 0.075, 0.075]},
                       ],
                       "goal": [
                       {"obj_name": "cube_target", "fixed": 1, "rand_rot": 0, "sampling_area":[-0.2, 0.2, 0.7, 0.8, 0.075, 0.075]},
                       {"obj_name": "cube_target", "fixed": 1, "rand_rot": 0, "sampling_area":[0.3, 0.5, 0.7, 0.8, 0.075, 0.075]},
                       ]},
```

Language module

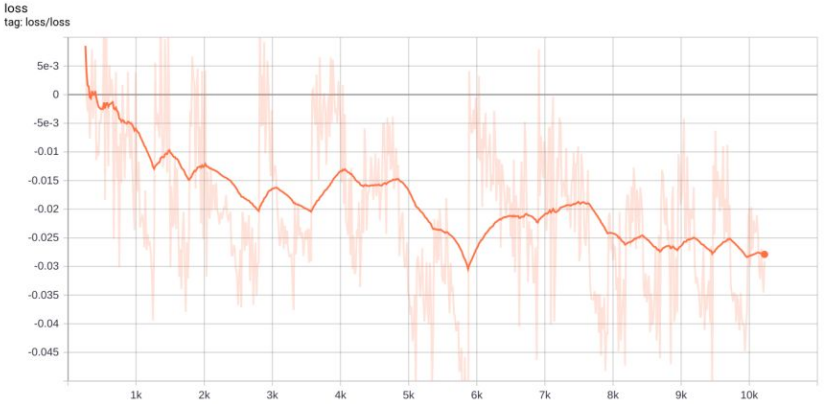
The language module capable of parsing simple expressions was added. An expression usually contains information about object color and/or spatial information (in relation to other objects). LM can extract this information and use it for defining robotic tasks. Moreover, LM can produce a description of already existing tasks.

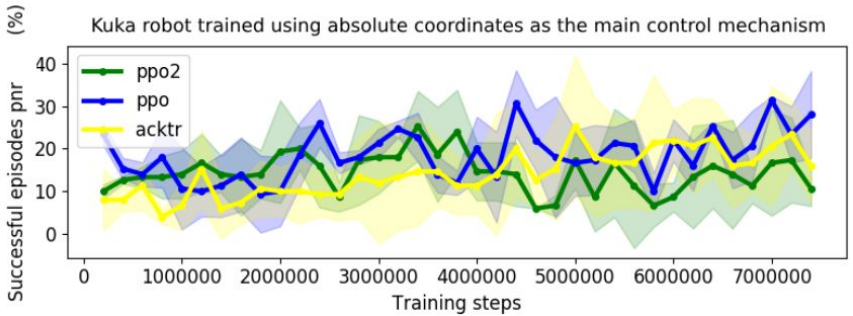
The supported tasks are NL Reach, NL Push, NL Throw, NL Pick and Place, and NL Pick and Place variations (swipe/rotate). One can manually test parsing and task definition based on it during the testing phase.



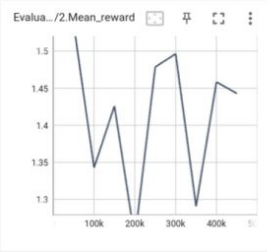
Results

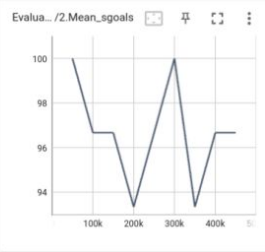
Robots are successfully learning using either the human gesture or verbal descriptions, which is demonstrated by the decreasing loss from the Tensorboard. A more detailed evaluation is represented in the corresponding work. The two robots (Kuka and Panda) were used with different reinforcement learning algorithms in different tasks (Reach gesture, NL Reach, NL Pick, and Place).

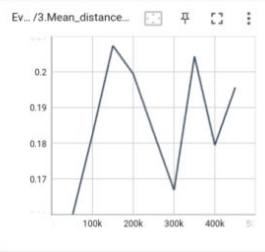


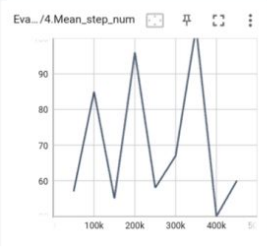


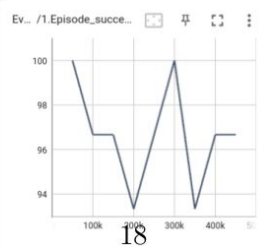
NL Reach gesture











NL Reach