

Mission Manual: Ladder Climbing Events

(Will Be continuously Updated as the Development of Each Component)



INDIANA UNIVERSITY

Yajia Zhang, Jingru Luo

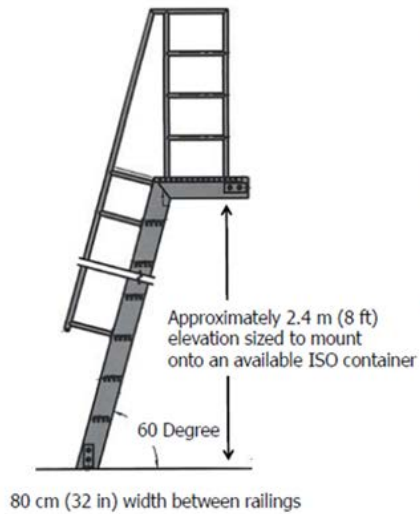


Andy Park, Manas Paldhe

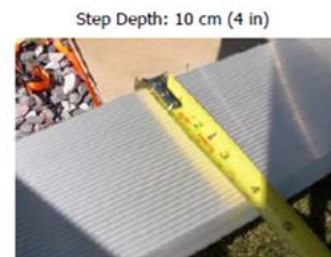
Events Summary

- **Events Description:** mount, climb and dismount an industry ladder

- **DARPA's specification:**



Step Height: 30 cm (12 in)



Step Depth: 10 cm (4 in)

Figure 1 DARPA's ladder specification

- **Key parameters:**
 - Inclination : **60** degree
 - Rung space: **30** cm
 - Rung depth: **10** cm
 - Ladder width: **80** cm
 - Rung Number: **9**
 - Ladder top height: **240** m

Events Pipeline

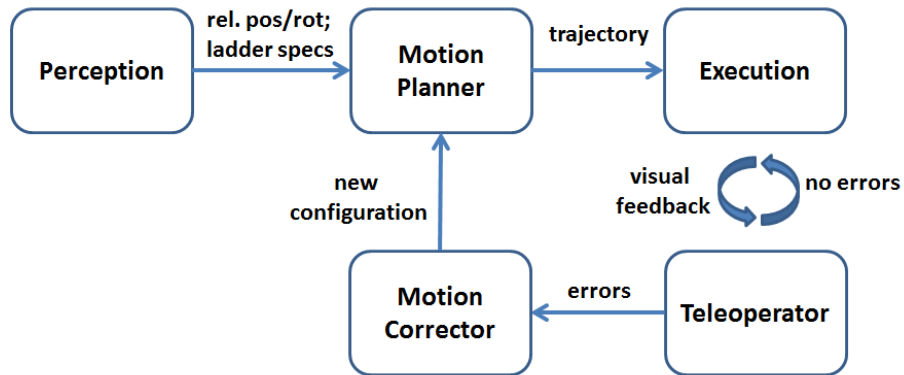


Figure 2 Flow chart of pipeline.

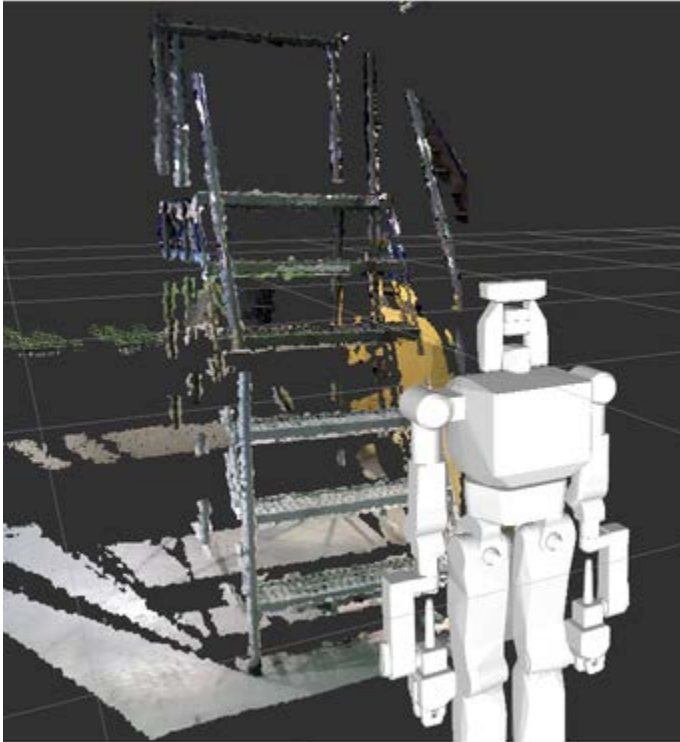
- **Interface** (under development)

An RVIZ panel running in ROS integrating the above components:

- Display point cloud/RGBD world from Hubo head
- Button confirming ladder specifications obtained from perception
- Button trigger motion planner and animation in RobotSim
- Button for sending trajectory to robot
- Button for starting trajectory executing
- Button/key press for pausing and resuming motion
- Button for correcting hand/foot pose

Perception

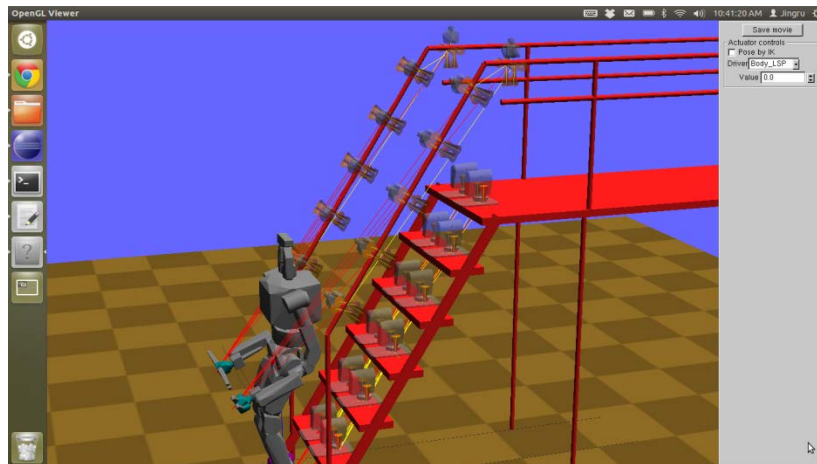
- **Input:** point cloud/RGBD world from Hubo head



- **Output:** ladder specifications for planning
 - first priority (for planning the motion)
 - ladder mesh
 - ladder position/orientation w.r.t the robot
 - slope, rail height, rail width (distance between two side rails), position of rails relative to ladder
 - vertical first and rest rung spacing (may be different), rung depth, number of rungs
 - first priority (during the motion)
 - the robot feet position/orientation on the rung
 - the hand position/orientation on the rail
 - second priority
 - rail diameter (thickness) for gripping distance

Planning

- **Planner description:**
 - Input: ladder specs, robot initial position/orientation
 - Output: a motion trajectory that can perform mount, climb and dismount.
- **Software Package: RobotSim**
 - Installation: (to be updated) contact Jingru Luo (luojing@indiana.edu) for details
 - Key Files:
 - Motion planner **LadderClimbingPlanner** is located in “RobotSim/DRC_Yajia”
 - Scenario file describing the world of robot and ladder: ShipLadder.xml
 - Ladder Spec file: shipladder_setting.xml
 - Generate executable programs:
 - AniTest program: In the root folder of RobotSim, type “make AniTest” to generate an executable file “AniTest”
 - SimTest program: In the root folder of RobotSim, type “make SimTest” to generate an executable file “SimTest”
- **AniTest program:** call **LadderClimbingPlanner** to generate motion for climbing; animate the generated trajectory.
 - Start the program
 - ./AniTest scenario_file laddersetting_file
 - E.g., ./AniTest ShipLadder.xml shipladder_setting.xml



- Usage:
 - Press: “P” for generating trajectory
 - Press: “L” for loading trajectory for animation
 - Press: “S” for animating climbing motion
- Output: A .txt file named “q_path.txt” which contains a list of configurations will be generated.

- Format conversion from RobotSim format “q_path.txt” to hubo-read-trajectory format:
 - Change filename to what you want inside config2huboach.py
 - python config2huboach.py
 - be ready to run path “huboach_q_path.txt”
- **SimTest program:** physically *simulate* the generated trajectory.
 - ./SimTest scenario_file
 - Key press: “P” for generating trajectory
 - Key press: “L” for loading trajectory for animation
 - Key press: “S” for animating climbing motion

Execution:

Person to contact: Manas Paldhe (mpaldhe@purdue.edu 303-717-4048)

Software used:

- Hubo-ach:
Repository: <https://github.com/hubo/hubo-ach>
- Hubo-motion-rt:
Repository: <https://github.com/hubo/hubo-motion-rt>
Secondary repository: <https://github.com/manaspaldhe12/hubo-motion-rt>
- Hubo-read-trajectory:
Repository: <https://github.com/hubo/hubo-read-trajectory>
Secondary repository: <https://github.com/manaspaldhe12/hubo-read-trajectory>
- Hubo-walk:
Repository: https://github.com/hubo/hubo_walk
Secondary repository: https://github.com/manaspaldhe12/hubo_walk
- Hubo-init
Repository: <https://github.com/hubo/hubo-init>

How to run the trajectory on the robot:

1. Install all the repositories as mentioned in their readme files
2. Copy the trajectory file to the hubo. You can use scp to do this.

```
scp /path/to_the_file hubo@ip://path_to_hubo_read_trajectory/filename
```
3. Goto hubo-read-trajectory
4. Start hubo-ach:

```
sudo hubo-ach start drc
```
5. Home all and initialize sensors
6. Start logging the sensor data:
 In another terminal, ssh into hubo and go to hubo-ach
 Do script filename.log
 Do sudo ./hubo-read

After the trajectory is over, do “ctrl+c” and then “exit”

Send over the filename.log to Jingru Luo and Yajia Zhang

7. Run the trajectory:

```
sudo ./hubo-read-trajectory -f 100 -n filename -c -p
```

Here, 100 is the frequency. You can use 10, 25, 50, 100 or 200

filename is the trajectory file to run

Use -c if you want the compliance mode. If you use this all upper body will be in compliance mode. If you want to change the compliance gains, do:

```
sudo vim //etc/hubo-ach/drc-hubo.joint.table
```

and change the gains as you want. Remember after this you have to restart hubo-ach.

Use -p flag to enable pause and play mode. In this mode you can press p to pause or to continue the trajectory.

Future work:

Currently the trajectory has no feedback and it does not use the hubo-motion-rt. We plan to use a GUI and hubo-motion-rt to have robust feedback control.

The hubo-walk panel has a “Ladder Tab” that we will use to input the ladder parameters. This will be replaced by perception. On pressing “Plan and Run” (the UI will change soon) the ladder parameters are sent to the planner. The planner receives them, plans the trajectory and sends it to the hubo. The hubo will execute the trajectory with slight modifications by the feedback controller.