Institut de Robòtica
i Informàtica Industrial

Robot-person accompaniment simulator tutorial (ASP-SI): Docker installation and launch

Ely Repiso, Anaís Garrell, Alberto Sanfeliu.

European Commission | Horizon 2020 European Union funding for Research & Innovation

AI4EU TERRINet The European Robotics Research Infrastructure Network

This work has received partial founding from European Union's Horizon 2020 research and innovation program under the projects AI4EU (H2020-ICT-2018-2-825619) and TERRINet (H2020-INFRAIA-2017-1-two-stage-730994)

CSIC Consejo Superior de Investigaciones Científicas

UPC Universitat Politècnica de Catalunya BarcelonaTech

1. Brief Description

This document describes how to install the set of C++ ROS libraries of the Robot-Person accompaniment simulator.

The document includes a Docker installation and launch, which is like a virtual machine. The installation is explained for Ubuntu18.4 with ROS-Melodic. However, the Docker can be installed in other Ubuntu operating systems, but maybe some functionalities have reduced capabilities.

The installation instructions are also included in the README.md file which is in the docker with the name: iri_companion_docker_melodic_ana_y_dabo.zip.

With this system you can do simulations of robot accompaniment for a person, while avoiding other moving and static people or obstacles. For more information please take a look in the publication: The Adaptive Social Planner using a Side-by-side Individual accompaniment (ASP-SI) in Repiso et al. IROS2017.

This work has been developed at Institut de Robòtica i Informàtica Industrial (IRI, CSIC-UPC) (www.iri.upc.edu). Contact: erepiso@iri.upc.edu.

2. Docker Installation

2.1. Install the docker: Open a terminal on Ubuntu.

```
/home$ sudo apt install docker.io

/home$  sudo systemctl start docker

/home$  sudo systemctl enable docker
```

2.2. Install plugin-docker:

```
/home$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \

   && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \

   && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee
/etc/apt/sources.list.d/nvidia-docker.list

/home$ curl -s -L https://nvidia.github.io/nvidia-container-
runtime/experimental/$distribution/nvidia-container-runtime.list | sudo tee
/etc/apt/sources.list.d/nvidia-container-runtime.list

/home$  sudo apt-get update
```

/home$ sudo apt-get install -y nvidia-docker2

/home$ sudo apt-get update

/home$ sudo apt-get upgrade

2.3. Enable docker run without need of sudo:

/home$ sudo usermod -aG docker ${USER}

/home$ su - ${USER}

/home$ sudo systemctl restart docker

3. Create the Docker Image and Compile the system

Open a terminal on Ubuntu.

/home$ cd /path_to_Dockerfile   // arrive to the docker folder.

/home$ docker build -t iri_companion .

4. How to launch the system

4.1.1. Run container from created image: Open a terminal on Ubuntu.

/home$ cd /path_to_Dockerfile

/home$ ./pal_docker.sh -it iri_companion /bin/bash # creates a new container

4.1.2. Inside the Docker image (virtual machine), launch gazebo for Ana-robot in FME: Same terminal as before.

/home$ roslaunch iri_ana_gazebo sim_sample_companion_with_person.launch world_name:=fme_door_open

4.2. Open a new terminal inside the container and launch the nav, detection and tracking nodes for Ana-robot (FME):

/home$ cd /path_to_Dockerfile

/home$ docker exec -it $(docker ps -l -q) bash   #open the previous container.

/home$ roslaunch iri_robot_aspsi tracker_nav_and_detection_for_ASPSI.launch map_name:=fme_door_open

Note: This launch also launches the Rviz visualization. In Rviz you can visualize the scene with the map, robot position, person detections, laserscan data, etc.

4.3. Open a new terminal inside the container and run the accompaniment for Ana-robot (FME):

/home$ cd /path_to_Dockerfile

/home$ docker exec -it $(docker ps -l -q) bash

/home$ roslaunch iri_robot_ aspsi gazebo_ASPSI_OK_ana.launch

Note: This launch also launches the rqt_reconfigure. Use the rqt_reconfigure to dynamically change parameters.

4.4 Open a new terminal inside the container and run a person teleoperation node:

/home$ cd /path/to/Dockerfile

/home$ docker exec -it $(docker ps -l -q) bash

/home$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/person1/cmd_vel __name:=person1

Note: The last command will allow you to teleoperate `person1`. Modify to teleoperate another available person on the scene.

5. How to use the accompaniment

<u>**5.1 To start running the accompaniment system you will need to:**</u>

5.1.1. On the rqt_reconfigure window, you need to reach in the menu to the AkpLocalPlanner reconfigure page (ana > move_base > AkpLocalPlanner), and specify the id_person_companion value to match the desired person id (visible on Rviz), or double-click the `select_near_pers_as_companion_one_person`, so the closest person id is set.

5.1.2. On Rviz, use the top `2D Nav Goal` tool and click somewhere free in the map.

5.1.3. On the 3rd terminal where you ran the teleop_twist_keyboard, focus it and follow terminal instructions to move the person around.

5.1.4. The robot should start accompanying the person.

<u>**5.2. Brief description of the simulation markers:**</u>

Our plan is multi-path, in addition to predicting the movement of people. If you want to see the markers in Rviz corresponding to that data, you have to increase the value of *vis_mode* in rqt_reconfigure to:

vis_mode=1) Planner Local window + Best path + static_destinations of local-planner.

vis_mode=2) + Force markers + Laser obstacles

vis_mode=3) + all the planned paths in 2d

vis_mode=4) + People prediction markers

vis_mode=5) + all paths in 3d (both include space + time)

6. Other settings to launch the system with different robots and maps

<u>**6.1 To Launch all the system for robot Ana in BRL environment:**</u>

<u>New terminal:</u>

/home$ cd /path/to/Dockerfile

/home$ ./pal_docker.sh -it iri_companion /bin/bash

/home$ roslaunch iri_ana_gazebo sim_sample_companion_with_person_brl.launch world_name:=master_big

New terminal:

/home$ cd /path/to/Dockerfile

/home$ docker exec -it $(docker ps -l -q) bash

/home$ roslaunch iri_robot_ aspsi tracker_nav_and_detection_for_ASPSI_brl.launch map_name:=master_big

New terminal:

/home$ cd /path/to/Dockerfile

/home$ docker exec -it $(docker ps -l -q) bash

/home$ roslaunch iri_robot_aspsi gazebo_ASPSI_OK_ana_brl.launch

6.2 To Launch all the system for robot Dabo in BRL (Barcelona Robot lab) environment:

New terminal:

/home$ cd /path/to/Dockerfile

/home$ ./pal_docker.sh -it iri_companion /bin/bash

/home$ roslaunch iri_dabo_gazebo sim_gazebo_dabo_companion.launch

New terminal:

/home$ cd /path/to/Dockerfile

/home$ docker exec -it $(docker ps -l -q) bash

/home$ roslaunch iri_robot_aspsi gazebo_ASPSI_BRL_OK.launch

Also if you want to launch all the system with only one launch you can use it in a terminal. To Launch the hole system with Dabo robot do the following:

/home$ cd /path/to/Dockerfile

/home$ ./pal_docker.sh -it iri_companion /bin/bash

/home$ roslaunch iri_dabo_gazebo sim_gazebo_dabo_companion_all.launch

7. Possible problems external to the proper behavior of the system and how to solve them

<u>**7.1 Problems due to computational cost overload:**</u>

In the system that uses the Dabo-robot, you can see this problematic behavior. In this case it is due to something wrong in the tf's of the robot model. If you see the messages when you launch the node, you can see how the tf's transformations take more time than needed.

<u>**IMPORTANT:**</u> Take care of this problem, because it may cause collisions with objects or people. <u>**7.2**</u> section describes a solution to avoid these collisions to ensure security when you use the system in a real robotic platform.

Sometimes, due to the mentioned computational cost overload, the robot may show a "S" navigation behavior which is not appropriate.

If you want to remove it, you could do the following:

<u>**7.1.1. If the problem is due to reduced computation capabilities of the Docker/virtual-machine:**</u> Use the system directly on your Ubuntu operating system (you have not to use the docker/virtual-machine). To install the system on your Ubuntu, you need to adapt to your computer the commands included in iri_companion_docker_melodic_ana_y_dabo/Dockerfile.

<u>**7.1.2. If the problem persists and it is due to a high number of static or dynamic obstacles**</u>**:** You have several solutions for this problem. Let us go to see some of them.

> <u>**7.1.2.1.1. Static obstacles, how to reduce the robot's radius that is used for detecting static obstacles in the rqt_reconfigure:**</u> We detect the obstacles close to the robot using a circle centered in the robot geometric position. This circle is the range-laser detection from the robot to the obstacles. You can reduce this radius by decreasing the value of the variable detection_laser_obstacle_distances in the rqt_reconfigure. This distance is set up in meters. For more details, see the Document: ASP-SI_Tutorial_Capabilities_Document.pdf.
>
> <u>**7.1.2.1.2. Static obstacles, how to reduce the computational cost in the detection of static obstacles (a second solution) (in this case, it implies additional code from your side):**</u> Also, you can change the detector of the static obstacles by another more efficient detector. The present detector in the simulator, detects an obstacle as a group of small cylinders instead of an unique obstacle, which from the computational point of view could be not as efficient as if the detector detects only the unique object.
>
> <u>**7.1.2.2. Dynamic obstacles (people), how to reduce the distance in rqt_reconfigure**</u>**:** Reduce radius in the meters around the robot position to detect people, by reducing the value of the variable *radi_to_detect_other_people_no_companions* in the rqt_reconfigure. Sometimes, it is better to reduce it directly in the detector. In this case, you have to go to the lpd page in the rqt_reconfigure and reduce the value of meters of the variable *filterR* and set up *filterPosesMode=true*.

7.1.3. If the problem is directly the computational cost to generate all the paths: You can reduce the local window of the local planner and the number of vertex of all the paths. In the rqt_reconfigure, reduce the value of the *horizon_time* to diminish the size of the local window, and in the rqt_reconfigure, reduce the value of the *number_vertex* to diminish the number of vertices of all the paths. By default, these parameters are set up to the minimum value for a good robot behavior, *horizon_time=4 segs* and number_vertex=200. If your system can not handle these minimum values, you need a more powerful computer to do the simulations. The default parameters used in all our real experiments are: *horizon_time=5 segs* and number_vertex=500. We test the simulator in an Intel Core 2 Quad CPU @ 2.66 and 3.00 GHz, but if you need additional information, please refer to the publication of Repiso et al. IJSR2019.

7.1.4. If the problem is due to the high number of markers to visualize in Rviz: You can solve it by disabling the markers that you do not need to see: detector, detector-filtered by the map, etc. Also, you can reduce the visualization markers of the planner that implements the accompaniment diminishing the value of the variable *vis_mode* in the rqt_reconfigure.

7.2. Problems of robot proximity to any person due to system overload (this may cause collisions): If the system has a high computational load and the controller can not meet its desired rate, then the controller does not control the robot in real time. Then, it is good for safety reasons to include a new node that stops the robot at a safety distance, for example 0.3 cm. In our real-life system, we use a ROS-node that does this behavior, but the version of this node is not included in the system due to delays on the migration to ROS-melodic.

7.3. Problems of collisions of the dynamic final destination with the map's obstacles: When the simulator is computing the path of the accompaniment group (robot and person) to the final destination and then there are people interfering with the path or in general when there are obstacles that do not allow that the group reaches the final destination in a direct way, then the simulator has to create a “temporal final destination”. This new “temporal final destination”, which is denominated as “dynamic final destination”, is close to the final static destination and is separated by a specified distance that can has to be defined (see the article E. Repiso IROS2019 for additional explanation). The present simulator has already a default distance value that can be modified. The global ROS planner is very sensitive to this parameter, and can stop the robot' planner path if it detects that it can not reach the final destination for the aforementioned reasons. Solutions: (1) disable the dynamic goal computation; or (2) reduce the distance that separates the dynamic final destination from the environment's final static destination. To disable the dynamic final destination computation do the following: in the rqt_reconfigure, change to false the boolean in_change_dynamically_final_dest. To reduce the dynamic goal's distance from the final static destination do the following: in the rqt_reconfigure, you have to reduce the value of the variable distance_to_dynamic_goal_Vform. This distance is in meters.

7.4. The robot movement is not smooth: We have a set-up of robot's velocities and accelerations that could be different from your robot. Because your robot has other characteristics of mass, robot's controller, robot's dimensions, etc., you have to customize the linear, angular velocities and accelerations of the method. This information is included in the rqt_reconfigure, under the label “*Parameters to adjust the robot's accelerations and velocities”*.