Chapter 18
Deployment of Advanced Robotic Solutions: The ROS Mobile Manipulator Laboratories

David St-Onge, Corentin Boucher, and Bruno Belzile

18.1 Introduction

Throughout the book, we introduced the various disciplines and topics involved in the development of robotic systems. Most of them are individually covered in dedicated books showing the extent of the knowledge required in robotics. Fortunately, we learned in Chap. 5 that open-source community-based software ecosystems, such as the Robotic Operating System (ROS), can support several of the integrations and ease the deployment process. ROS provides users with access to the latest research algorithms and software deployment tools for code maintenance, visualization, simulation, and more. Thanks to ROS, we expect all readers of this book to be able to complete the challenging advanced mobile manipulator tasks of this chapter.

For that purpose, we designed a robotic platform and we built a custom dedicated ROS workspace to support our academic teaching laboratories in robotics, from Gazebo simulations to the physical deployment (see Fig. 18.1). While undergoing a single-semester introduction course in robotics, the students are not expected to install and deploy the ROS workspace on their personal computer, but rather use a server infrastructure available at the university. Nevertheless, we provide the complete ROS workspace with detailed installation instruction,[1] including a Docker container to ease the deployment. The laboratories were designed at École de technologie supérieure, in Montréal, Canada, where we host the physical infrastructure to which these tools are tailored. We have eight robotic platforms for the students

[1] https://github.com/Foundations-of-Robotics/mobile_manip_ws.

D. St-Onge (✉) · C. Boucher · B. Belzile
Department of Mechanical Engineering, École de Technologie Supérieure, Montréal, Canada
e-mail: david.st-onge@etsmtl.ca

B. Belzile
e-mail: bruno.belzile.1@ens.etsmtl.ca

D. Herath and D. St-Onge (eds.), *Foundations of Robotics*,
https://doi.org/10.1007/978-981-19-1983-1_18

Fig. 18.1 A group of students testing the robots outside of the laboratory

to test their algorithms and several preconfigured desktop stations. The stations are accessible remotely to run the full simulation stack including remote visual rendering of the simulation on a web browser. We tested up to three teams running their Gazebo simulations in parallel on a single station without degrading performances.[2] Each station is accessible to all teams and so team's accounts are set on each station. Since most of the students' work is done in Jupyter notebooks, we deploy on each station (the littlest) JupyterHub.[3] Details on users' management and stations configuration, including Gazeboweb (gzweb) and JupyterHub are also available online.[4]

18.1.1 Dingo and Gen3 Lite

The motivation behind this set of assessments is to cover several topics relevant to both mobile robots and manipulator robots with a single integrated robotic platform. Several engineering courses are available with hands-on laboratory on industrial manipulators and custom-made mobile robot prototypes (a great project in this last category is covered in Chap. 17 of this book). What we found was lacking is teaching material leveraging a commercial robotic platform fit (safe) for proximity with the users. Our suggested platform, to which we tailored all this chapter's content, brings

[2] Tested on Dell ThinkStation P340 Tiny i7, 16 Gb RAM with NVidia Quadro1000 4 Gb.

[3] https://tljh.jupyter.org/en/latest/

[4] https://github.com/Foundations-of-Robotics/mobile_manip_ws/tree/master/doc.

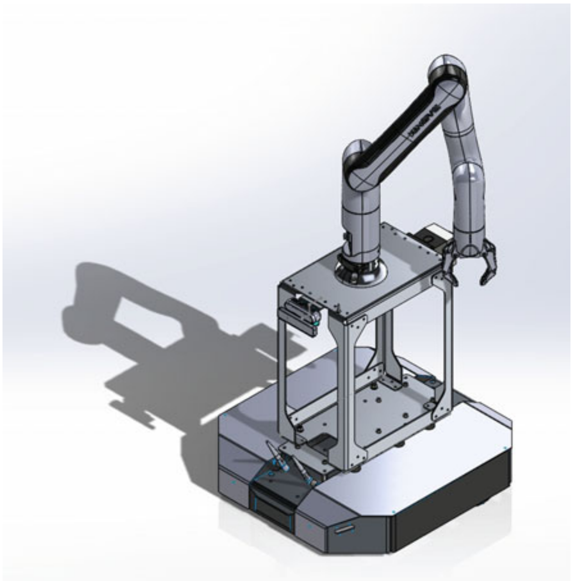


Fig. 18.2 CAD model of the robotic platform with its custom turret

together a Clearpath Dingo[5] differential drive mobile base with a Kinova Gen3 lite[6] six-degree-of-freedom manipulator.

This recent version (2020) of the now-famous Kinova arms has been designed specifically for teaching. The same goes for Clearpath's mobile platform, the Dingo, which was released just a couple of months after the Gen3 lite. Clearpath (Toronto) and Kinova (Montreal) are considered good potential employers for students passioned by robotics and their hardware can be found in thousands of companies and universities around the world. Kinova's arms are deployed in disability assistance centers and hospitals, not to mention human–robot collaborative research laboratories, and the company is currently working to increase its presence in the industrial sector. Clearpath provides products for NASA and several emergency response agencies around the world and in recent years has become a major player in warehouse automation. Among everything, both companies are contributing actively to the ROS community with nodes for their robots and several additional ROS tools to ease robotics deployment (Fig. 18.2).

We put together the arm and the base using a custom-made turret that ensures the arm can easily reach door handles and objects on tables at human height. The turret also hosts a set of Realsense cameras: one for position tracking, the other used for mapping, obstacle avoidance and user teleoperation. To increase the platform safety, we added an emergency stop button on the back of the turret, which can be locked to prevent any motion of the robots. Anyhow, both the arm and the base have

[5] https://clearpathrobotics.com/dingo-indoor-mobile-robot/

[6] https://www.kinovarobotics.com/product/gen3-lite-robots.

their own remote controller for manual control and to take over when autonomous control is not behaving properly.

18.1.2 Recommended Tools and Base Skill Set Required

The reader undergoing the projects of this chapter must have a good understanding of the content of most of the chapters in this book. However, the essentials are the ROS environment (Chap. 5), Python basic programming (Chap. 4), differential drive kinematics (Chap. 8), homogeneous transformation (Chap. 6), Kalman state estimation (Chap. 8), serial manipulator kinematics (Chap. 10), navigation (Chap. 9) and designing a user study (Chap. 13). In the following, the first three projects are well framed in order to help the student's focus on the theoretical content and their learning of the tools, namely ROS and Python. The last two projects are less structured as they are meant to allow the students to explore more advanced topics and integrate the knowledge they acquired in the previous projects and throughout the book. All projects notebooks, such as the one shown in Fig. 18.3, are available online.[7] The following set of instructions take for granted that you have local access to a ROS-configured station and access the robots' onboard computer (also preconfigured) through a local (wireless) network. The ROS workspace repository also contains instructions for the Dingo setup and to launch a simulation on a remote station.[8]

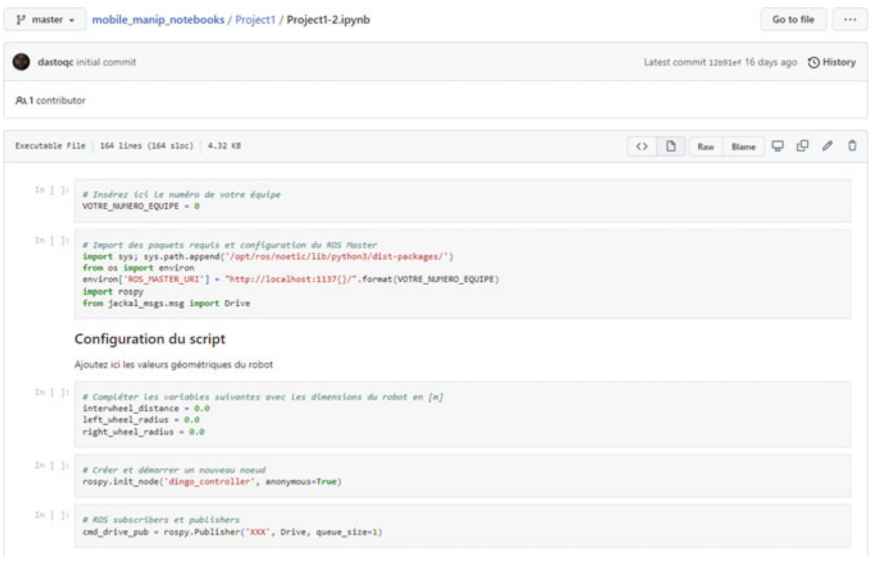
18.2 Project 1: Discovering ROS and the Dingo

18.2.1 Project Objectives

- Get familiar with the ROS environment (basics);
- Get familiar with the Gazebo simulator (visualization only);
- Get familiar with Python programming (from notebooks);
- Control a mobile robot manually;
- Control a robot with simple Python instructions;
- Program a robot's differential drive kinematics.

[7] https://github.com/Foundations-of-Robotics/mobile_manip_notebooks.

[8] https://github.com/Foundations-of-Robotics/mobile_manip_ws/tree/master/doc.


master
mobile_manip_notebooks / Project1 / Project1-2.ipynb
Go to file
dastoqc initial commit
Latest commit 12b91ef 16 days ago
History
1 contributor
Executable File | 164 lines (164 sloc) | 4.32 KB
Raw
Blame

```
In [ ]: # Insérez ici le numéro de votre équipe
        VOTRE_NUMERO_EQUIPE = 0

In [ ]: # Import des paquets requis et configuration du ROS Master
        import sys; sys.path.append('/opt/ros/noetic/lib/python3/dist-packages/')
        from os import environ
        environ['ROS_MASTER_URI'] = "http://localhost:1137{}/".format(VOTRE_NUMERO_EQUIPE)
        import rospy
        from jackal_msgs.msg import Drive
```

Configuration du script

Ajoutez ici les valeurs géométriques du robot

```
In [ ]: # Compléter les variables suivantes avec les dimensions du robot en [m]
        interwheel_distance = 0.0
        left_wheel_radius = 0.0
        right_wheel_radius = 0.0

In [ ]: # Créer et démarrer un nouveau noeud
        rospy.init_node('dingo_controller', anonymous=True)

In [ ]: # ROS subscribers et publishers
        cmd_drive_pub = rospy.Publisher('XXX', Drive, queue_size=1)
```

Fig. 18.3 Notebook example from Project 1 shown on the GitHub repository

18.2.2 Project Description

This project aims at comparing the performance of a mobile robot (Clearpath Dingo) in simulation and in reality, through the extraction of the resulting trajectories' noise. In order to compare the two, an autonomous open-loop trajectory must be programed and to do so the Dingo's differential drive inverse kinematics is required.

18.2.3 First Task: Manual Control in Simulation

- Open a terminal and use this command to start the simulation:
  roslaunch mobile_manip dingo_arenasim.launch
- Open a second terminal and use the command **rostopic** (**list, info** and **echo**) to find the topics' names for the IMU (sensor_msg/imu), the encoders (nav_msgs/odometry), and the velocity command (geometry_msgs/Twist).
- Use these topics to record a rosbag with this command:
  rosbag record /topic1 /topic2 /topic3
- Open a third terminal and run the teleoperation node (Fig. 18.4):
  rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=mobile_manip/cmd_vel

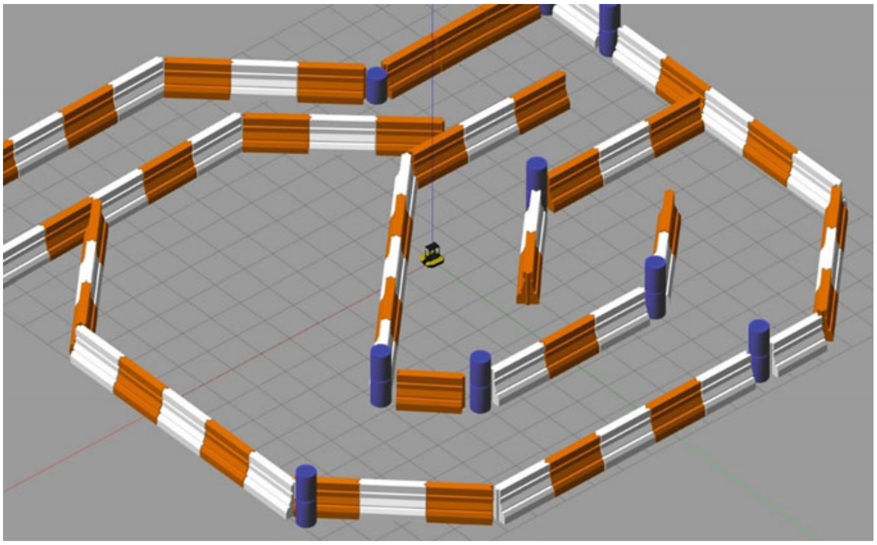


Fig. 18.4 Dingo robot in a simulated maze

Keeping a focus in the last terminal window, you can use the keyboard (*u*, *i*, *o*, *j*, *k*, *l*, *m*, <) to manually control the robot toward the end of the maze. When you are out of the maze, stop the rosbag (control-C), and then the simulation.

18.2.4 First Task: Manual Control in Reality

- Start the robot by pressing the power button and wait for the front lights to turn white.
- Power on the controller. You can now control the robot with the controller (Fig. 18.5).

18.2.5 Second Task: Inverse Kinematics

The controller and the state estimator need information from the robot's sensors to work properly. The provided Python notebook **Project1–2** for this task contains missing ROS topics names and some geometry information of the robot that you need to fill out.

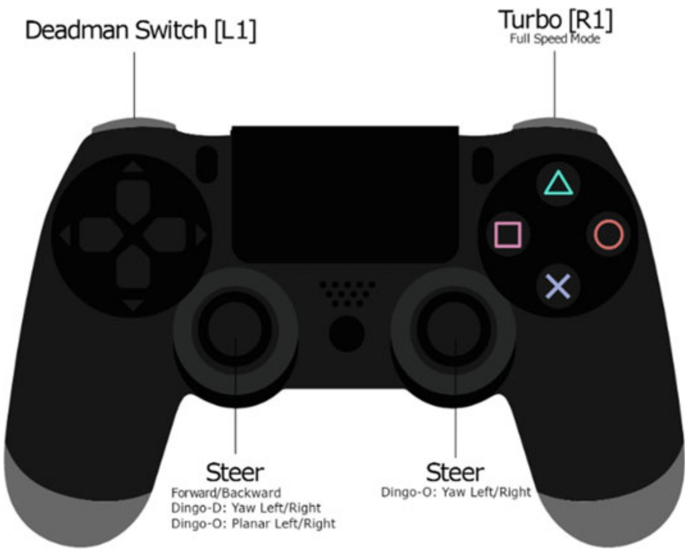


Fig. 18.5 Clearpath remote controller, from the Dingo manual

You can now program your differential drive robot's inverse kinematics. Look for the function "move_robot" in the notebook, it's the one that needs to be modified. This function uses two arguments as inputs:

- **vlin** is the desired linear velocity in m/s in the robot's frame.
- **vang** is the desired angular velocity in rad/s in the robot's frame, positive counterclockwise (Z is up).

Your work consists in using these variables and the robot's geometry to calculate the velocity commands sent to each wheel. The variables **vel_left** and **vel_right** will contain the results of your derivation. The function **move_robot** will then automatically send the commands to the robot whenever its called.

You can test your code by moving the robot in an empty world simulation with:

roslaunch mobile_manip gen3_lite_dingo_emptysim.launch

using the variables **vlin** and **vang** in the last cell of the Python notebook. When you are ready, record a rosbag of the topic **/tf** only (it includes the transforms of each rigid body in the simulation) for the following movements:

- Moving forward in a straight line.
- Moving backward in a straight line.
- Rotate on itself.
- Turn on a circle with a diameter of 1 m.

18.2.6 Third Task: Simulation Versus Reality

Use the **project1–3** notebook and enter the topics' names in the publishers and subscribers where it is needed. Edit the values for the velocity of each wheel to make the robot turn on a circle of 1 m in diameter. This notebook will be used to control the robot both in simulation and in reality. You will need to record a rosbag for each and then compare the results.

To record the **simulation**:

- Open a terminal and run the simulation with
  roslaunch mobile_manip gen3_lite_dingo_emptysim.launch
- With **rostopic** command, find the topics' names for the IMU (sensor_msg/imu), the encoders (nav_msgs/odometry), and the motor command (jackal_msgs/cmd_drive) and edit **project1–3** notebook accordingly.
- The last cell of the notebook records a rosbag with the needed topics while the robot is moving. Make sure to close the rosbag once the circle is done at least once.

To record the **real** robot:

- Turn on the robot using the power button on the back (ensure the emergency stop button is disabled) and wait until the light indicator is on below the Wi-Fi symbol.
- Open a new terminal on your computer and connect to the robot over SSH (where *X* is your Dingo's number):
  ssh mecbot@cpr-ets05-0X (user `mecbot` and the password given by your professor)
- Then when you are connected to it, launch all the custom nodes required for the laboratories' task onboard:
  roslaunch mobile_manip gen3_lite_dingo_real.launch

Finally, to compare the results, open the notebook **Analyse1**, enter the name of your rosbags and edit the code to calculate the variance on the circle trajectory. Using at most one page, answer this question:

- What quantifiable difference(s) can you observe between the circle trajectory in simulations and on the real robot? What can you say about the source(s) of these differences?

18.3 Project 2: Kalman for Differential Drive

18.3.1 Project Objectives

- Design a Kalman filter for position estimation (sensor fusion);
- Estimate sensors' noise from real data.

18.3.2 Project Description

This project aims at developing a state estimator for the mobile base pose from the available onboard sensors' measurements. You will use a Kalman filter to fuse the different measurements and get the best out of each sensor. Since such a piece of code requires quite a bit of debugging, use only the simulation until you get a working behavior:

- To launch a simulation, use this launch file:
  roslaunch mobile_manip gen3_lite_dingo_emptysim.launch

18.3.3 First Task: Extract Encoders Information (Notebook Project2_1)

Your first task is to use the encoders' values to estimate the robot's movements (x, y, Vx, Vy, and the heading θ). The function **encoders_callback** will be called each time a new update is received from the wheels' encoders. Modify its content to estimate progressively the robot's position and velocity from the encoders' readings (given in radians from 0 to infinity).

18.3.4 Second Task: Estimate the Sensor's Noise (CSV_Analyse)

To include sensors in a Kalman filter, you first need to know the noise in the measurements. Previous experimental data has been saved in .csv files and are provided to you alongside a template notebook to help with loading and processing the data (**CSV_Analyse**). You need to find the variance of every measurement you will use in your Kalman filter in order to build your measurement covariance matrix.

18.3.5 Third Task: Design a Kalman Filter (Project2_2)

Now that you extracted meaningful information from the encoders and that you estimated the encoders and IMU noise, you need to fuse these measurements into a more robust state estimator. Indeed, since the wheels can slip, the IMU measurements can improve the pose estimation. This fusion will be made with a Kalman Filter. The given Python notebook already imports the library **filterpy.kalman** which is responsible for most of the Kalman filter implementation, but you need to configure the filter (states, covariance matrices, transition matrices, etc.). Follow the instructions in the notebook and refer to the library documentation for more details.[9]

There is not a unique solution to this task. The filter can be configured in different ways, for example using (or not) command inputs (u). If you choose a configuration without commands, consider the system's variance to be $\boldsymbol{\sigma}_S = 10$.

18.3.6 Fourth Task: Design Justification and Validation

The last cell of the **project2_2** notebook sends commands to move the robot. This cell also creates a rosbag recording the trajectory estimated by your Kalman filter and the ground truth trajectory from gazebo. You can then test your Kalman filter performance with the **KF-Analyse** notebook.

In a page, describe and justify the design of your Kalman filter: Why did you choose these states? Why do you think this model (configuration matrices) is a better fit to the problem? What other possibilities were available?

18.4 Project 3: 3-DoF Kinematics

18.4.1 Project Objectives

- Compute direct and inverse kinematics;
- Calculate the Jacobian and the points of singularity;
- Validate the results with the real robot.

18.4.2 Project Description

This project aims at the application of the direct and inverse kinematics to the use case of a simplified manipulator. The robot used for this laboratory is the Kinova Gen3 Lite manipulator (see Fig. 18.6). In order to simplify this project (and avoid the need for a symbolic computation software), the number of axes to be controlled is reduced from 6 to 3.

As shown in Fig. 18.7, axes 1, 2, and 3 are the ones that can be controlled here. Assume all other joints are fixed in their initial position (0°), as depicted in Fig. 18.6. A 90° angle is applied to the third joint on the left-hand side of the figure for the sake of comparison. The parameters of each of the robot segments are given in Fig. 18.7 (in mm).

[9] https://filterpy.readthedocs.io/en/latest/kalman/KalmanFilter.html.

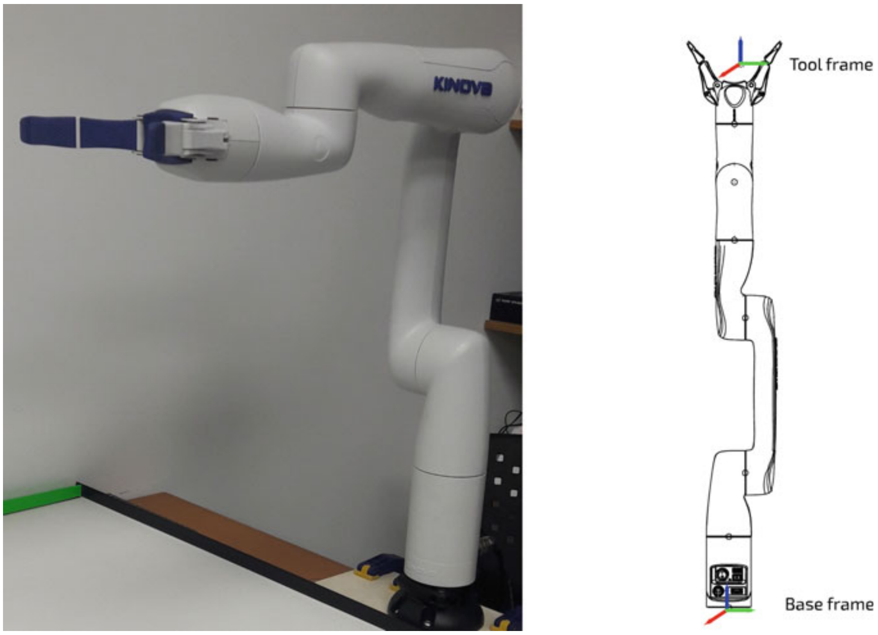


Fig. 18.6 Gen3 lite manipulator: photograph on the left, base, and tool frame locations on the right

18.4.3 First Task: Denavit–Hartenberg Table

You must complete the DH table (again, for the 3-DoF robot) with the parameters given (their numerical values, not only the variables). Add comments if necessary to clarify the meaning of the values used.

18.4.4 Second Task: Transformation Matrices

Compute all sequential homogeneous transformation matrices (Q) from the DH table, then use the resulting matrices to obtain the final concatenated transformation matrix, or direct kinematics.

18.4.5 Third Task: Inverse Kinematics

Derive the equations to compute each joint coordinate explicitly as a function of a Cartesian position (not orientation) to be reached within the workspace with the end-effector. A drawing might help you in your work.

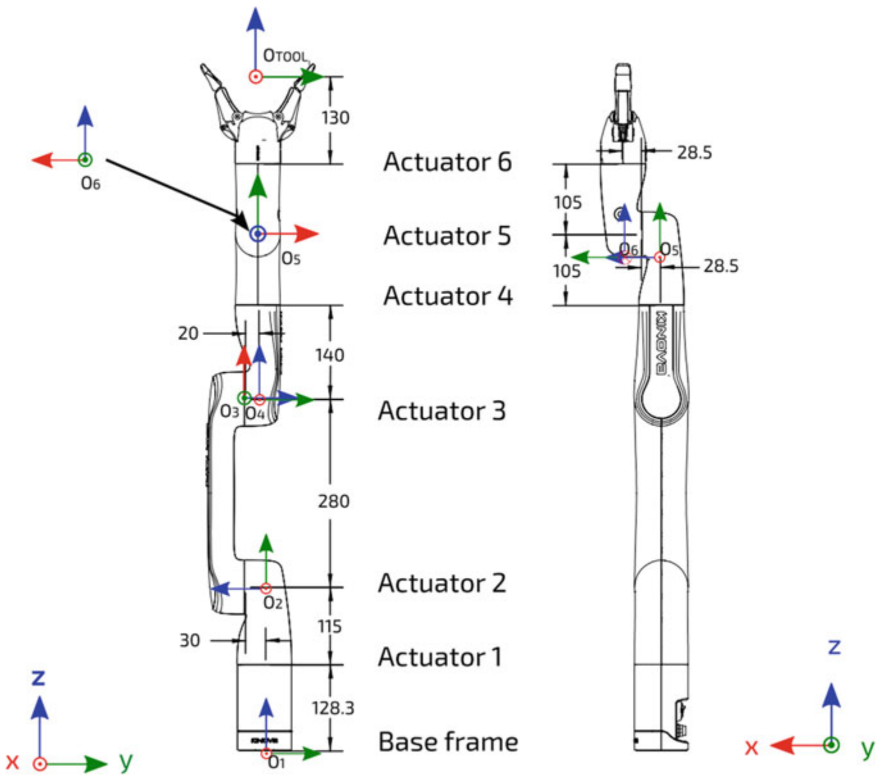


Fig. 18.7 All joints reference frames from the Gen3 lite manual

Suggested procedure:

1. Extract the three components (x, y, z) defining the position from the homogeneous transformation matrix computed in the previous task.
2. Eliminate unknown variables to obtain an equation with only one unknown left. You can do this similarly to the procedure used to solve the inverse position problem of a wrist-partitioned serial manipulator, in Chap. 10.

   (a) Use the equations corresponding to the coordinates x and y, then compute the sum of their squares (to eliminate θ_1) and isolate $\cos(\theta_2)$. Then, after isolating $\sin(\theta_2)$ in the equation of the z-coordinate, use the identity $\sin 2(\theta_2) + \cos 2(\theta_2) = 1$ to make one unknown joint coordinate disappear, namely θ_2.

   (b) Use the Weierstrass substitution with $(\theta_2 - \theta_3)$ in the equation obtained in the previous step and find the roots of the obtained quadratic equation. The possible values of $(\theta_2 - \theta_3)$ can then be computed.

   (c) Going back to the expressions of $\sin\theta_2$ and $\cos\theta_2$ obtained earlier, you can now compute θ_2 with the arctan2 function.

(d) θ_2 and $(\theta_2 - \theta_3)$ known, it is trivial to obtain θ_3.
(e) Finally, return to the equations corresponding to the coordinates x and y, then cast them in matrix form $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{x} = [\sin\theta_1 \ \cos\theta_1]^T$ since θ_1 is the only remaining unknown. Then, after solving for $\boldsymbol{x}$, use arctan2 to find the solution for θ_1.

18.4.6 Fourth Task: Validation

In order to validate your solution, you must now apply the results obtained in the previous step to the robot. To do this, the robot must follow a path passing through the following Cartesian positions:

[mm]	x	y	z
Pose 1	−66.9	−50.2	965.8
Pose 2	223.3	366.8	873.0
Pose 3	10.0	678.0	384.3

Start the robot simulation using the following command:
roslaunch mobile_manip gen3_lite_sim.launch
Use the ***Kinova_3DoF_Joint_Control*** notebook: this file contains a function named ***dof3control(j1,j2,j3)***. This function sends angle commands (in degrees) to the 3 joints of the simulated robot in Gazebo. You need to add your code implementing the solution to the inverse kinematics problem (previous task).

You are encouraged to validate the angle values returned by your inverse kinematics using the ***DirectKinematics*** notebook. Finally, you can confirm that the position reached by the robot is good with the call at the bottom of the ***Kinova_3DoF_Joint_Control*** notebook.

When your code has proven to work properly in Gazebo, you can validate on the real robot by changing the ROS_MASTER_URI in the first cell to the IP of your robot. Then launch the nodes on the real robot with:

- Turn on the robot using the power button on the back (ensure the emergency stop button is disabled) and wait until the light indicator is on below the Wi-Fi symbol.
- When the front lights turn white, turn on the Gen3 Lite (button on the back on the base of the arm)
- Open a new terminal on your computer and connect to the robot over SSH (where X is your Dingo's number):
  ssh mecbot@cpr-ets05-0X (user `mecbot` and password given by your professor)
- Then when you are connected to it, launch all the custom nodes required for the laboratories' task onboard:
  roslaunch mobile_manip gen3_lite_dingo_real.launch

If it launches correctly, you should see the arm reach its home position.

In less than a page, answer the following question: To your knowledge, what criteria could be used to select one solution to the inverse kinematics problem over another? Justify, it is not necessary to have an exhaustive list of criteria.

18.5 Project 4: Let's Bring It Back Together!

18.5.1 Project Objectives

- navigate with a mobile manipulator;
- manipulate the environment;
- identify phenomena potentially dangerous involving mobile manipulators and analyze the risk.

18.5.2 Project Description

This project consists of deploying a mobile manipulator in a real application scenario: the handling of machined parts in a factory. More specifically, your objective is to recover a part that has just been machined (in the corridor) and transfer it to a surface treatment tank (in the laboratory). Two robots are available for this task. The two robots are initially in the laboratory and must therefore open the door themselves to reach the part. All tasks can either be run in simulation or with the real robot using the same Python notebook.

To test any part of your code in a safe environment, do it in simulation. Open a Linux terminal and start the simulation with the command:

roslaunch mobile_manip gen3_lite_dingo_labsim.launch

After you validate your code in the simulation, you can launch it with the real robot:

- Turn on the robot using the power button on the back (ensure the emergency stop button is disabled) and wait until the light indicator is on below the Wi-Fi symbol.
- When the front lights turn white, turn on the Gen3 Lite remote (button on the back on the base of the arm).
- Open a new terminal on your computer and connect to the robot over SSH (where X is your Dingo's number):
  ssh mecbot@cpr-ets05-0X (user `mecbot` and password given by your professor)
- Then when you are connected to it, launch all the custom nodes required for the laboratories' task onboard (Fig. 18.8):
  roslaunch mobile_manip gen3_lite_dingo_real.launch

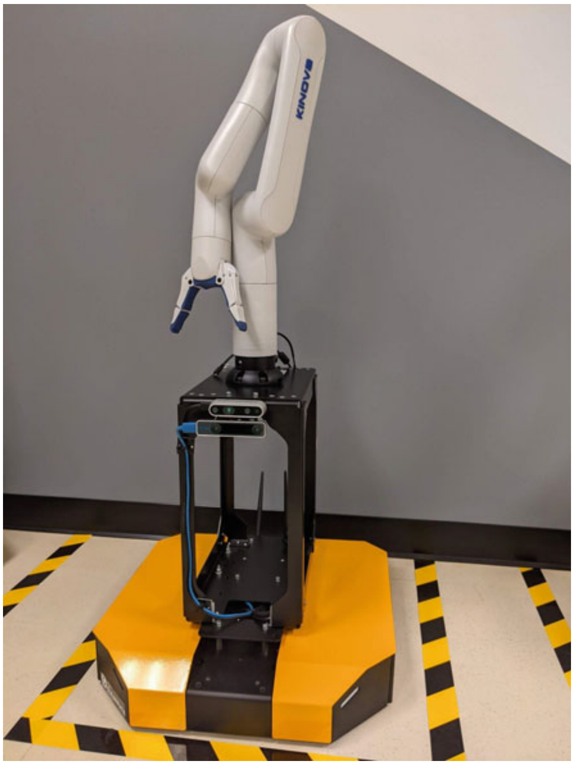


Fig. 18.8 Mobile manipulator platform in its parking space

If it launches correctly, you should see the arm reach its home position. Keep the remote control in your hands at all times: the deadman switch allows you to take back manual control of the robot if your code reacts badly. *Warning! The remote control only interrupts the commands sent to the Dingo not the ones sent to the arm.*

18.5.3 First Task: Teleoperation

Use the notebooks prepared for this task in order to pilot the robot in the room and the corridor in front. You need to test:

- manual piloting with the remote control, following the robot (not available in the simulation);
- remote manual control with the notebook and using visual feedback from the front color camera (**manual-control.ipynb**—remember to test in simulations first!);
- autopilot using the notebook, but immediately regaining control if a collision is imminent. (**autonomous.ipynb**—remember to test in simulations first!).

18.5.4 *Second Task: Hit a Marker!*

Use the **TagTouch** notebook to detect markers in front of the robot using the Realsense T265 fisheye camera. The coordinates obtained are in the reference frame of the camera, you must transpose them into the frame of reference of the manipulator base. The position of the camera in the reference frame of the base of the manipulators is illustrated in Fig. 18.9:

- in the simulation, $[x, y, z] = [0.16, 0.04, 0.463]$ m;
- on the real robot, $[x, y, z] = [0.0, 0.05, -0.137]$ m.

Remember that the frame of reference of an image (camera) always has the z-axis coming out of the camera. Then use the inverse kinematics (Cartesian control) provided by the Gen3 lite controller to touch the marker on the wall. Test first in the simulation, then in the laboratory. You can move the robot manually using one of the strategies from step 1 to position the robot (camera) in front of the marker.

You cannot use Kinova's web service for this and subsequent steps.

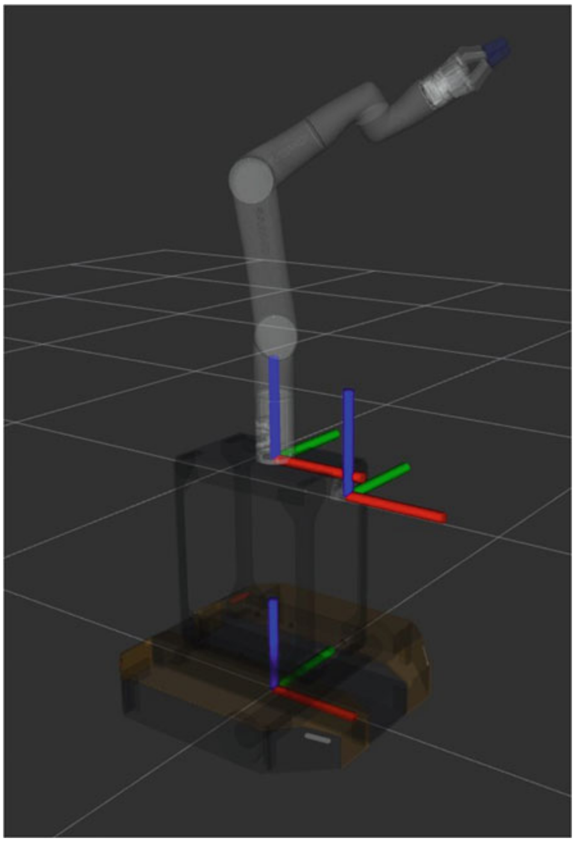


Fig. 18.9 Mobile manipulator model in Gazebo with reference frames for the arm base, the mobile base, and the camera. *X*-axis is shown in red, *Y-axis* in green, and *Z-axis* in blue

Fig. 18.10 Two Apriltags were used for this task, both in the simulation and in the real deployment

18.5.5 Third Task: Grasping

The door handle and the object to be picked up are both at predetermined poses relative to their fiduciary marker, as shown in Fig. 18.10. To the transformation from the previous step, you must now add the transformation required to change frame:

- of the handle (given in the referential of the marker), $[x, y, z] = [0.05, 0.3, 0.1]$ m;
- of the object (given in the referential of the marker), $[x, y, z] = [0.05, -0.45, 0.2]$ m.

When you reach your goal, you must then tighten the gripper on it. Write down the movements required to orient the gripper and apply the required force. You can then use the Dingo and Gen3 lite together to pull the door and bring the object back. **Note:** To send an opening or closing command to the gripper, you must use the ROS service for this purpose (look for *reach_gripper* in the notebooks).

18.5.6 Fourth Task: Risk Assessment

Based on what your learnings from Chap. 14, you have to take care of the risk management for the mobile manipulator in this scenario. You are the integrator, you must therefore deliver an initial analysis of the risks as well as the relevant means of risk mitigation. Keep in mind that the initial analysis is done without considering the existing protection mechanisms on the robotic system. Then, propose risk mitigation measures and clearly state which ones are already included in the robotic system and those which should be added (to the system, to the environment or by the management).

To help you in your assignment, the operator of the mobile manipulator system has shared the following information with you:

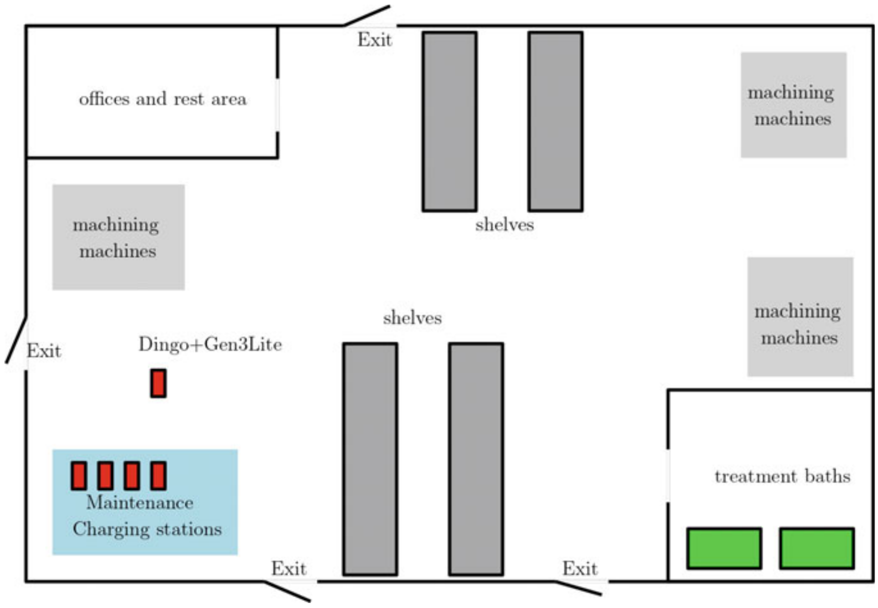


Fig. 18.11 Fictitious plan of a factory cell with mobile manipulators

- the robotic system to be deployed is a Clearpath Dingo equipped with a Kinova Gen3 lite arm;
- the task of the industrial robotic system (Dingo and Gen3 lite) is to pick the parts from various machining machines to place them in surface treatment baths (*machine tending*);
- the machines manufacturing the parts are spread over the entire factory floor, with workers circulating in between regularly;
- treatment baths are in a closed room to meet ventilation standards for the chemicals used;
- production runs 24/7.

To help you with this task, a plan layout is illustrated in Fig. 18.11.

18.6 Project 5: Save the Day!

18.6.1 Project Objectives

- Program autonomous navigation for the Dingo;
- Program obstacle avoidance;
- Design a remote command interface;
- Plan, conduct, and analyze the ergonomy of a command interface.

18.6.2 Project Description

This project consists in studying the ease of use of a teleoperation system you will be designing. You will first need to design a controller and a user interface before conducting a user study. The mission of the participants will be to explore the floor of a building to find some objects. The building is evacuated, only the mobile manipulator is left inside (no human to avoid).

To test any part of your code in a safe environment, do it in simulation. Open a Linux terminal and start the simulation with the command:

roslaunch mobile_manip gen3_lite_dingo_labsim.launch

After you validate your code in the simulation, you can launch it with the real robot:

- Turn on the robot using the power button on the back (ensure the emergency stop button is disabled) and wait until the light indicator is on below the Wi-Fi symbol.
- When the front lights turn white, turn on the Gen3 Lite remote (button on the back on the base of the arm).
- Open a new terminal on your computer and connect to the robot over SSH (where X is your Dingo's number):
  ssh mecbot@cpr-ets05-0X (user `mecbot` and password given by your professor)
- Then when you are connected to it, launch all the custom nodes required for the laboratories' task onboard:
  roslaunch mobile_manip gen3_lite_dingo_real.launch

If it launches correctly, you should see the arm reach its home position. Keep the remote control in your hands at all times: the deadman switch allows you to take back manual control of the robot if your code reacts badly. *Warning! The remote control only interrupts the commands sent to the Dingo not the ones sent to the arm.*

18.6.3 First Task: Autonomous Navigation

Teleoperation requires a good deal of autonomy from the robotic system so it can deal with complex maneuvers and leave the operator to attend more sensitive tasks. You need to design your navigation solution with various levels of autonomy for 1. Path planning, 2. Collision avoidance, and 3. Objects (Apriltags) detection. A set of notebooks is provided with A* and RRT path planners, including an occupancy grid of the environment as well as visualization tools for laser scans (extracted from the depth camera) and video feeds (including tags detection in the camera reference frame). Many solutions are possible for each aspect of the navigation. For the Apriltags detection, remember that the coordinates obtained are in the reference frame of the camera, you must transpose them into the frame of reference of the arm base for manipulation. The position of the camera in the reference frame of the base of the manipulator is:

- in the simulation, $[x, y, z] = [0.16, 0.04, 0.463]$ m;
- on the real robot, $[x, y, z] = [0.0, 0.05, -0.137]$ m.

18.6.4 Second Task: User Interface

Based on the navigation solution you designed in the previous step, code a teleoperation interface for an operator to complete the mission remotely (i.e., without following the real robot and looking at the simulator window). This interface must include visualization of relevant sensor information and input modalities to send commands to the robot. A minimal interface for the manual control is given as an example (see Fig. 18.12).

This task is a creative step where you should try to imagine what would help the user the most. You can then do some research and find what is possible to do in a Python notebook to make the integration in your interface.

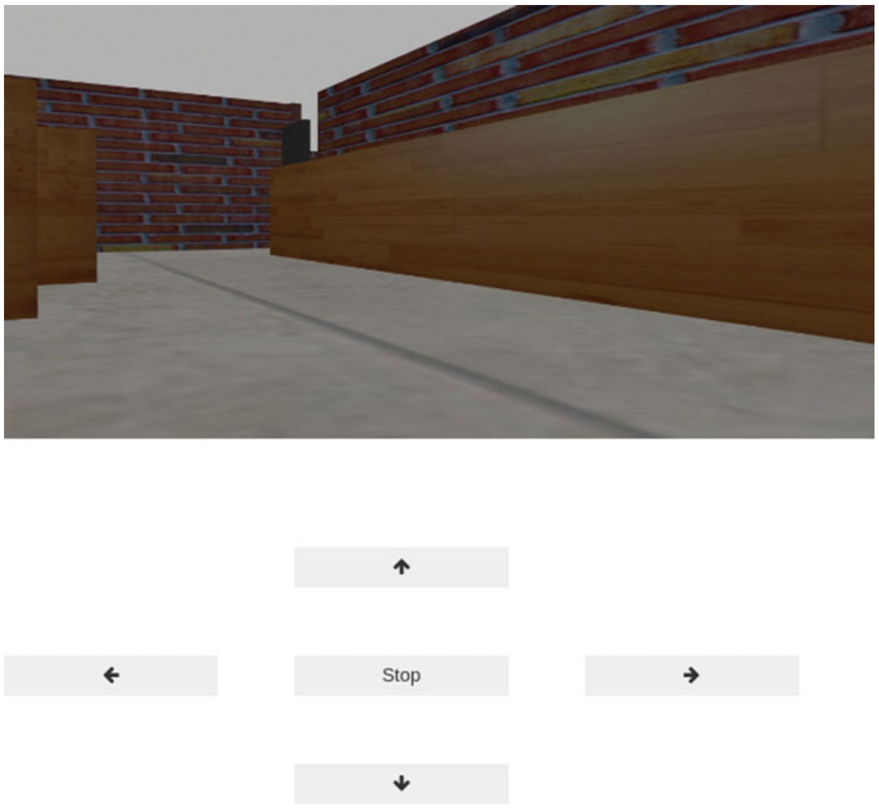


Fig. 18.12 Jupyter notebook minimal interface provided for the teleoperation of a mobile manipulator (camera view from simulation)

18.6.5 Third Task: Ergonomy Study

You can now design a user study (see Chap. 13) to assess the potential of your teleoperation solution, namely the impact of your solution on the operator's performance and his appreciation. You can either do an explorative study or a confirmative (comparative) one. To conduct a comparative study, you need an interface that the users can test in two different conditions (i.e., manual and assisted). In both cases, you need to define and justify the selected statistical tools (see Chap. 6). At the end of the term, you will conduct your study on some students of the course group.

You need to complete a protocol for your user study including the questionnaires and the metrics you will analyze to answer your research question. For instance, a common questionnaire used to measure the cognitive task load is the NASATLX:
NASATLX questionnaire example

Item	Endpoints	Description
Mental demand	1–10 Low/High	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, exacting or forgiving?
Physical demand	1–10 Low/High	How much physical activity was required (e.g., pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
Temporal demand	1–10 Low/High	How much time pressure did you feel due to the rate or pace at which the tasks occurred? Was the pace slow and leisurely or rapid and frantic?
Performance	1–10 Low/High	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
Effort	1–10 Low/High	How hard did you have to work (mentally and physically) to accomplish your level of performance?
Frustration level	1–10 Low/High	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the task?

Your ergonomy study report is expected to follow this structure:

1. INTRODUCTION—Introduce the content of this report, the main characteristics of your navigation and user interface and the results of the study.
2. USER STUDY DESIGN—Describe the elements of the study with the research question, causal or correlated effect sought, impact of the robotic system type, place of the study and the selection of the candidates. You must also list and justify the measures used.

3. STUDY PROTOCOL—Describe how you will proceed in the study sessions.
4. DATA ANALYSIS—Use statistical tools to demonstrate the results distribution.
5. DISCUSSION—Discuss the most important observations you made in the data analysis section. Mention the limitations of the study and add recommendations.

David St-Onge (Ph.D., Mech. Eng.) is an Associate Professor in the Mechanical Engineering Department at the École de technologie supérieure and director of the INIT Robots Lab (initrobots.ca). David's research focuses on human-swarm collaboration more specifically with respect to operators' cognitive load and motion-based interactions. He has over 10 years' experience in the field of interactive media (structure, automatization and sensing) as workshop production director and as R&D engineer. He is an active member of national clusters centered on human-robot interaction (REPARTI) and art-science collaborations (Hexagram). He participates in national training programs for highly qualified personnel for drone services (UTILI), as well as for the deployment of industrial cobots (CoRoM). He led the team effort to present the first large-scale symbiotic integration of robotic art at the IEEE International Conference on Robotics and Automation (ICRA 2019).

Corentin Boucher is a research student at the École de Technologie Supérieure (ÉTS). The interest in robotics that he developed during his studies pushed him to continue his journey and to carry out research in the field.

Bruno Belzile is a postdoctoral fellow at the INIT Robots Lab. of ÉTS Montréal in Canada. He holds a B.Eng. degree and Ph.D. in mechanical engineering from Polytechnique Montréal. His thesis focused on underactuated robotic grippers and proprioceptive tactile sensing. He then worked at the Center for Intelligent Machines at McGill University, where his main areas of research were kinematics, dynamics, and control of parallel robots. At ÉTS Montréal, he aims at creating spherical mobile robots for planetary exploration, from the conceptual design to the prototype.