Mobipick Labs System Demonstration

Oscar Lima,*[1] Martin Günther,*[1] Alexander Sung,[1] Sebastian Stock,[1]
Marc Vinci,[1] Amos Smith,[1] Jan Christoph Krause,[1] Joachim Hertzberg[1,2]

[1]German Research Center for Artificial Intelligence (DFKI)
[2]Osnabrück University
{oscar.lima, martin.guenther, alexander.sung, sebastian.stock,
marc.vinci, amos.smith, jan_christoph.krause, joachim.hertzberg}@dfki.de

Abstract

In this demonstration we show the *mobipick_labs* environment, which is a testbed for planning, execution and monitoring algorithms in a robotic domain. It provides a physics-based simulation environment for a mobile robot together with full integration of the robot and easy-to-use interface for the robot's capabilities. This gives users without a strong robotics background the opportunity to apply their planning-related work to a robotic domain. Moreover, the framework includes a simple semantic and numeric environment representation, providing real-time knowledge in the form of "facts" that can be utilized to monitor the execution status. Finally we also demonstrate one possible solution using task planning to search and transport certain objects to a target location.

Video: https://youtu.be/4-GgOg2nuGE

Code: https://github.com/DFKI-NI/mobipick_labs/

Introduction

Mobile robots offer a variety of interesting challenges for the planning community, such as dynamic and partially observable environments with multiple actors and stochastic action outcomes, which results in a high uncertainty about the actual environment state. Dealing with those properties requires the use of sophisticated plan execution and monitoring algorithms. Despite being such an interesting application for planning, it is difficult for researchers to test their algorithms on real robots, due to the high entry barrier of learning robotics and ROS.

The mobipick_labs environment which we demonstrate here, is designed to make the entry into robotics easier for planning researchers by giving easy access to the robots capabilities via a user-friendly high-level Python API. Additionally, it provides a physics-based robot simulator in Gazebo (Koenig and Howard 2004) in order to test planning algorithms as well as the execution and monitoring of the plan. For a detailed description of the mobipick_labs system we refer to (Lima et al. 2023).

Simulation environment

Fig. 1 shows the Mobipick robot in the testing environment simulated in Gazebo. The environment consists of three tables with industrial-like objects, such as a multimeter, screwdriver, relay, power drill and a blue box (see Fig. 2). The positions of the objects can be adapted for different test scenarios. This simulation closely resembles our real-life robot setup. One example goal task for the robot is to transport the objects between the tables.

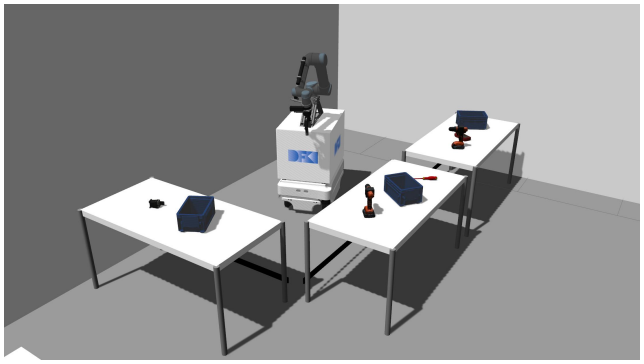

Figure 1: Simulated environment in Gazebo.

Figure 2: Available objects. (Note: objects are not to scale.)

The mobipick_labs environment currently only provides one specific robot (the Mobipick) and a limited set of objects in one simulation environment; adding custom robots, objects and environments requires a certain knowledge of ROS. We plan to provide multi-robot examples, an easier way to integrate new objects and algorithms for procedurally generating new environments in the near future.

Robot capabilities

The Mobipick robot can perform the tasks of autonomous navigation, 6DoF object detection and pose estimation, arm trajectory motion planning and execution, pick, place and insertion of objects including grasp planning, collision detection, and free space place sampling.

*These authors contributed equally.

The navigation action of the robot allows it to autonomously move to a goal positionwhile automatically avoiding dynamic obstacles on the path.

Object perception is done via a deep learning approach on the real Mobipick robot, whereas in simulation a Gazebo "logical camera" sensor is utilized to emulate object recognition, pose estimation, and object anchoring functionalities.

All robotic manipulation within mobipick_labs is based on MoveIt. Using this framework, the robot is able to grasp objects, place the held object on a table or to insert objects into boxes.

In order for the robot to reason about how to solve tasks, it needs the current state of the environment as symbolic facts. These facts can be automatically generated by the provided symbolic fact generator module. The generated facts include information about the current arm pose of the robot, the current position of the robot as well as the location of objects on tables.

Mobipick Robot API

The functionality described above has been configured, developed and tested over the course of several years and is aligned with the state of the art in open source robotics. Using it directly requires expertise in robotics and the Robot Operating System (ROS).

To lower the entry barrier, we have developed a high-level Python API to command the robot with simple instructions, which allows the user to control all capabilities of the robot without using ROS. Our hope is to encourage planning researchers to test their planning algorithms in a physics based simulator that is very close to a real robot. The small example in Listing 1, taken from the robot_api documentation, demonstrates its overall idea.

Listing 1: Robot API basic functionality

```
import robot_api
# Get a Robot object from the ROS namespace
mobipick = robot_api.Robot("mobipick")

# Get the robot's 2D pose from localization
robot_pose = mobipick.base.get_2d_pose()

# Move the robot's arm using MoveIt
mobipick.arm.move("transport")

# Move the robot's base using move_base
mobipick.base.move(21.0, 7.0, 3.141592)
```

This is achieved by using discovery mechanisms at runtime on the available ROS topics and services, so the robot_api can make use of existing ROS components of a system dynamically. Where one would typically set up a ROS subscriber or ROS service client first to establish the connection and communicate with other ROS components, the robot_api performs this in the background. It also calls `rospy.init_node()` on demand, i.e., if there is no ROS node already running in the current process, a new node will be initialized the first time it is needed.

Planning and Execution Example

The provided testing environment can be solved with different variants of planning ranging from classical or hierarchical task planning to temporal planning up to probabilistic planning. In this demo, we demonstrate how we use one of the classical planners provided by the Unified Planning Bridge (Hastam Sathiya Satchi Sadanandam et al. 2023) for an example task of transporting objects. In this example, we give the robot the goal that a multimeter shall be in a blue box on a target table.

We modelled actions for driving to a table, searching on a table for objects, picking up an object, placing an object and transporting an object.

Initially, the robot is aware of the poses of the three tables but it does not know the locations of the objects. Therefore, the created plan contains actions for searching for a multimeter and a box and afterwards to put the multimeter into the box and transport and place the box onto the target table. The resulting plan is handled by a plan executor that iterates through the plan and sequentially dispatches the actions. The actions for searching for an object is not executed directly on the robot but is handled in the execution loop by creating a sub-planning problem which is again solved by the task planner. That sub-plan includes driving to the different tables and searching for objects on each of them. This sub-plan is again executed sequentially in an execution loop that stops if the object is found. Those sub-plans are created based on the robot's current knowledge of the environment. For example, when searching for the box it takes into account that it did not see a box on the first two tables when it was searching for the multimeter. In our example, the robot successfully places the multimeter in to the box that is found on the third table and afterwards transports the box to the target table.

Users of the mobipick_labs simulation environment can swap in their own planners by either integrating them into the Unified Planning Bridge or by executing the plans using the high-level Robot API.

Conclusion

The mobipick_labs environment serves as a testbed for researchers interested in planning, acting, and monitoring problems in robotic domains. It eliminates the need for expert knowledge in robotics or ROS, enabling researchers to focus on decision-making and execution aspects. Additionally, we present one example for using classical task planning in the domain.

Unlike e.g. ROSPlan(Cashmore et al. 2015), the Mobipick Labs system is not a planning framework, but rather a realistic robotics use case, which can be used to support researchers interested in decision making and online execution, enabling them to easily test their algorithms without much effort on the robotics side.

Acknowledgments

This work is supported by the CoPDA and InCoRAP projects through grants of the German Federal Ministry of Education and Research (BMBF) with Grant Numbers 01IW19003 and 0IW19002. It is also supported by the AI-Plan4EU project which has received funding from the European Union's Horizon 2020 research and innovation programme under GA no. 101016442. Additionally, we acknowledge the support from the APRIL EU project via the grant from the European Commission 870142. The DFKI Niedersachsen (DFKI NI) is sponsored by the Ministry of Science and Culture of Lower Saxony and the Volkswagen-Stiftung.

References

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. Rosplan: Planning in the robot operating system. In *Proceedings of the international conference on automated planning and scheduling*, volume 25, 333–341.

Hastam Sathiya Satchi Sadanandam, S.; Stock, S.; Sung, A.; Ingrand, F.; Lima, O.; Vinci, M.; and Hertzberg, J. 2023. A Closed-Loop Framework-Independent Bridge from AI-Plan4EU's Unified Planning Platform to Embedded Systems. In *ICAPS Workshop on Planning and Robotics (PlanRob 2023)*.

Koenig, N.; and Howard, A. 2004. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IROS 2004*, volume 3, 2149–2154. IEEE.

Lima, O.; Günther, M.; Sung, A.; Stock, S.; Vinci, M.; Smith, A.; Krause, J. C.; and Hertzberg, J. 2023. A Physics-Based Simulated Robotics Testbed for Planning and Acting Research. In *ICAPS Workshop on Planning and Robotics (PlanRob 2023)*.