Funded by the European Union

Robotics Coordination Action
for Europe Two (RockEU2)
Horizon 2020

ERL-PSR

–European Robotic League for Professional Service Robots –

II: ERL-PSR Rule Book

Revision: 1.1
Date: July 5, 2020

Editors: Deebul Nair, Gerhard Kraetzschmar, Sven Schneider

Contributors: Aamir Ahmad, Francesco Amigoni, Iman Awaad,
Jakob Berghofer, Tim Friedrich, Rainer Bischoff,
Rhama Dwiputra, Nico Hochgeschwender, Frederik Hegger,
Sven Schneider, Gerhard Kraetzschmar
Luca Iocchi, Giulio Fontana, Andrea Bonarini,
Pedro Lima, Matteo Matteucci, Daniele Nardi,
Viola Schiaffonati

Hochschule
Bonn-Rhein-Sieg

POLITECNICO
DI MILANO

Contents

1 Introduction to ERL-PSR

The objective of the European Robotics League is to organize several indoor robot competition events per year, ensuring a scientific competition format, around the following two challenges: ERL-CSR and ERL-PSR. Those indoor robot competitions will be focused on two major challenges addressed by H2020: societal challenges (service robots helping and interacting with humans at home, especially the elderly and those with motor disabilities) and industrial leadership (industrial robots addressing the flexible factories of the future and modern automation issues). These challenges were addressed by RoCKIn and RockEU2 and will be extended in the European Robotics League by building on the current version of the rule books and testbeds designed and used during RockEU2's project lifetime.

Greater automation in broader application domains than today is essential for ensuring European industry remains competitive, production processes are flexible to custom demands and factories can operate safely in harsh or dangerous environments. In the ERL-PSR competition, robots will assist with the assembly of a drive axle - a key component of the robot itself and therefore a step towards self-replicating robots. Tasks include locating, transporting and assembling necessary parts, checking their quality and preparing them for other machines and workers. By combining the versatility of human workers and the accuracy, reliability and robustness of mobile robot assistants, the entire production process is able to be optimized.

The ERL-PSR competition is looking to make these innovative and flexible manufacturing systems, such as that required by the RoCKIn'N'RoLLIn factory, a reality. This is the inspiration behind the challenge and the following scenario description. A more detailed account of the ERL-PSR competition, but still targeted towards a general audience, is given in the ERL-PSR in a Nutshell document, which gives a brief introduction to the very idea of the European Robotics League and the ERL-PSR competition, the underlying user story, and surveys the scenario, including the environment for user story, the tasks to be performed, and the robots targeted. Furthermore, this document gives general descriptions of the task benchmarks and the functional benchmarks that make up ERL-PSR. The document on hand is the rule book for ERL-PSR, and it is assumed that the reader has already read the nutshell document. The audience for the current document are teams who want to participate in the competition, the organizers of events where the ERL-PSR competition is supposed to be executed, and the developers of simulation software, who want to provide their customers and users with ready-to-use models of the environment.

The remainder of this document is structured as follows: Section 2, ***award categories*** surveys the number and kind of awards that will be awarded and how the ranking of the award categories is determined based on individual benchmark results. The ***testbed*** for ERL-PSR competitions is described in some detail in the next section (Section 3). Subsections are devoted to the specification of the structure of the environment and its properties (Section 3.1), to the mechanical parts and objects in the environment which can be manipulated (Section 3.2.1), to objects in the environment that need to be recognized for completing the task (Section **??**), to the networked devices embedded in the environment and accessible to the robot (Section 3.3), and to the benchmarking equipment which we plan to install in the environment and which may impose additional constraints to the robot's behavior (equipment presenting obstacles to avoid) or add further perceptual noise (visible equipment, see Section 3.5). Next (Section 4), we provide some specifications and constraints applying to the ***robots and teams*** permitted to participate in ERL-PSR. The European Robotics League consortium is striving to minimize such constraints, but for reasons of safety and practicality such constraints are required. After that, the next two sections describe in detail the ***task benchmarks*** (Section 5) and the ***functionality benchmarks*** (Section 6) comprising the ERL-PSR competition, while information on scoring and ranking the performance of participating teams on each benchmark is already provided in the benchmark descriptions.

2 ERL-PSR Award Categories

Awards will be given to the best teams in each of the ERL-PSR *task benchmarks* and *functionality benchmarks* that are described in Sections 5 and 6. For every local/major tournament, and for every task and functionality benchmark, a score is computed by taking the median of the best (up to 5) trials. The final end of year score is computed by taking the median of the pooled trials that were used for scoring the best two Local/Major tournaments and teams are ranked based on this score. The European Robotics League Competition awards will be given in the form of cups for the best teams. Every team will also receive a plaquette with the European Robotics League logo and a certificate.

Please note that teams need to participate in a minimum of two tournaments (Local and/or Major) per year in order to obtain a score for the TBMs and/or FBMs that they intend to enter.

2.1 Awards for Task Benchmarks

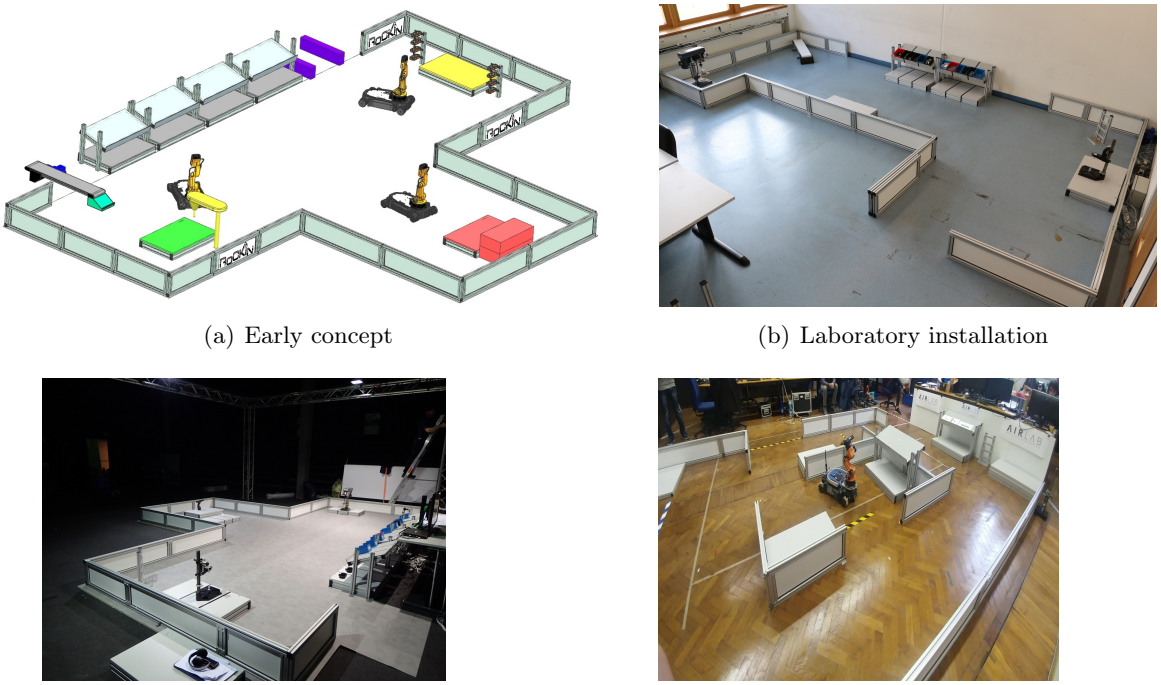
The team with the highest score in each of the three *task benchmarks* will be awarded a cup ("ERL-PSR Best-in-class Task Benchmark <*task benchmark* title>"). When a single team participates in a given *task benchmark*, the corresponding *task benchmark* award will only be given to that team if the Executive and Technical Committees consider the team performance of exceptional level.

2.2 Awards for Functionality Benchmarks

The teams with the highest score ranking for each of the three *functionality benchmarks* will be awarded a cup ("ERL-PSR Best-in-Class Functionality Benchmark <*functionality benchmark* title>" and 'ERL-PSR Second-Best-in-Class Functionality Benchmark <*functionality benchmark* title>"). When less than three teams participate in a given *functionality benchmark*, only the "ERL-PSR Best-in-class Functionality Benchmark <*functionality benchmark* title>" award will be given to a team, and only if the Executive and Technical Committees consider that team's performance as excellent.

3 The ERL-PSR Testbed

The testbed for the ERL-PSR competition consists of the arena (e.g. walls, workstation), networked devices and task-related objects. The robot can communicate and interact with the networked devices, which allow the robot to exert control on the testbed to a certain extend. Figure 1 shows the evolution of the ERL-PSR environment from its early concept in RoCKIn@Work to its implementation in the RoCKIn@Work event in Lisbon in 2015 and RoCKIn@Work last event in Polimi in 2017. Participating teams should assume the competition environment to be similar to those shown in Figure 1; deviations should only occur if on-site constraints (space available, safety regulations) enforce them.



(a) Early concept

(b) Laboratory installation

(c) RoCKIn@Work 2015

(d) RoCKIn@Work 2017

Figure 1: The evolution of the ERL-PSR environment

3.1 Environment Structure and Properties

The following set of scenario specifications must be met by the ERL-PSR environment.

Environment Specification 3.1 (*Structured Environment*)

The environment can consists of various numbers of spatial areas:

1. *rows of shelves*
2. *rows of workstations*

Figure 2 shows an example of these areas in the ERL-PSR environment. The spatial areas extend beyond the space occupied by the respective workstations or objects and include the surrounding area as well.

Environment Specification 3.2 (*Flat Environment*)

All spatial areas are located on the same level, except where specified otherwise. There are no stairs in the environment.

Environment Specification 3.3 (*Spatial Areas and Rooms*)

The factory is a single, large open space; there are no rooms separated by walls in the environment. Spatial areas can be partially separated by dividing or protective walls or other objects present in the factory (e.g. shelves, workstations, platforms, tables).

Figure 2: Example of the spatial areas in the ERL-PSR environment. The spatial areas are shelves (red), force fitting workstation (blue), conveyor belt (green), drilling workstation (orange) and assembly workstation (yellow).

Environment Specification 3.4 (*Dimensions*)

The precise dimensions and the arrangement of the spatial areas are not predefined, but estimated sizes are given. The estimated sizes of the spatial areas are as follows: workstations $2m \times 2m$ *and shelves* $5m \times 0,5m$*. The bounding box of the environment has a minimum area of* $16m^2$ *and a maximum area of* $100m^2$*. More space is used, when areas and workstations are doubled for teams working in parallel.*

Environment Specification 3.5 (*Set of Shelves*)

The shelves-area is a set of connected shelves and each shelves has two level (upper level and lower level). The robot can take and/or deliver objects from the shelves (through the containers or directly onto shelves). Figure 2 shows an example of the shelves-area made from two set of shelves.

Environment Specification 3.6 (*Workstations*)

Workstations are used as storage areas for objects. They may be accessible from different locations, i.e. it might be possible to reach a workstation from two or more sides. Additional, there are workstations of different heights present in the environment, ranging from 0 cm up to 15 cm. If a workstation has a height of 0 cm, a tape will mark the area (see Figure 3). The tape will be taped on the floor and is blue/white striped. This tape may be crossed by the robot and does not count as a collision.

Figure 3: Barrier tape used to mark workstations with 0 cm height.

Environment Specification 3.7 (*Barrier Tape as Virtual Walls*)

The environment may include virtual walls marked by either striped yellow/black or white/red barrier tape on the floor (see Figure 4). If any part of a robot passes over such a tape it is considered as a collision with a obstacle. The red/white tape is used to frame the entrance and exit area. The robot is allowed to cross this kind of barrier only at the beginning of a test to enter arena and at the end for leaving. In contrast, the yellow/black one denotes an obstacle which the robot is never allowed to cross.

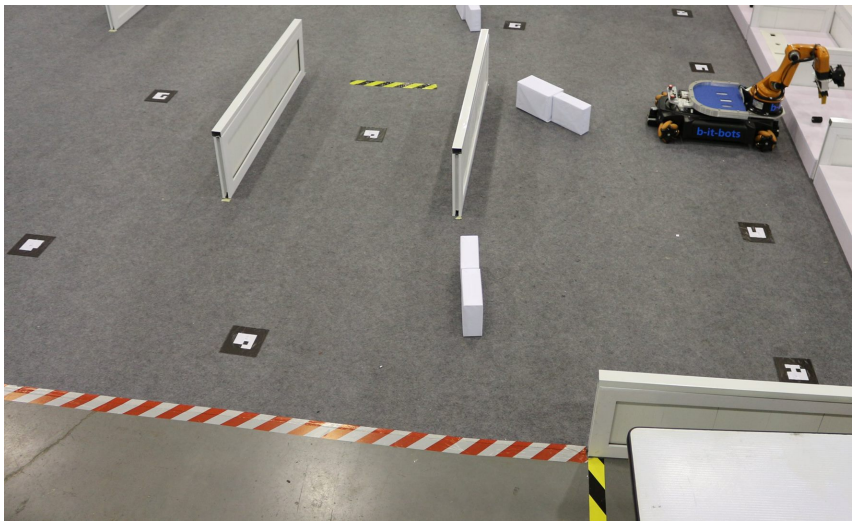


Figure 4: Example of the different kind of barrier tapes. The red/white tape is used for the entrance and exit, while the yellow/black one denotes an obstacle.

3.2 Objects in the Environment

The objects to be manipulated will be selected by the organizer for each tournament. The following lists describe the different categories of objects that can be used by the organizer to use during a tournament depending upon the scenario being choosen. The different categories of objects are:

- Robocup objects
- T-LESS dataset objects
- Chocolate objects

(a) Bearing Box (b) Motor (c) Axis (d) Bearing

(e) F20_20_B (f) F20_20_G (g) S40_40_B (h) S40_40_G (i) M20_100 (j) M20 (k) Distance_Tube

Figure 5: Robocup Objects

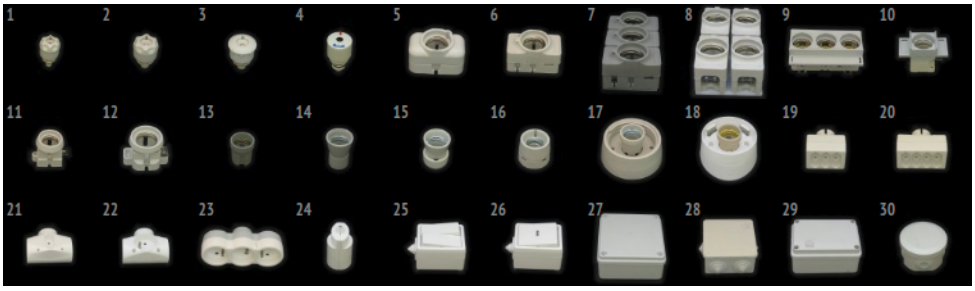


Figure 6: T-less dataset

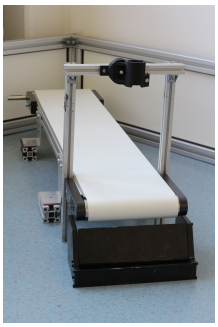
(a) (b) (c)

Figure 7: Chocolate objects

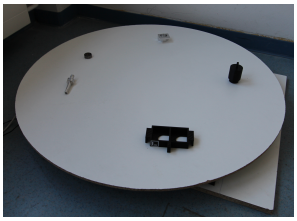
3.2.1 Manipulation Objects

3.3 Networked Devices in the Environment

There are various networked devices available in the ERL-PSR environment. The following description provides an overview on the capability of each networked device. An interface description for each networked device is provided in detail in the task benchmark section (see Section 5). Both networked devices are used for delivering parts to the ERL-PSR arena and they are operated by the referee(s).



(a) Conveyor belt



(b) Rotating table

Figure 8: Networked devices within the ERL-PSR environment.

3.4 Central Factory Hub

The main idea of the ERL-PSR testbed software infrastructure is to have a central server-like hub (the ERL-PSR Central Factory Hub) that serves all the services that are needed for executing and scoring tasks and successfully realize the competition. This hub is derived from software systems well known in industrial business (e.g. SAP). It provides the robots with information regarding the specific tasks and tracks the production process as well as stock and logistics information of the RoCKIn'N'RoLLIn factory. It is a plug-in driven software system. Each plug-in is responsible for a specific task, benchmarking or other functionality.
The following set of specifications must be met by the ERL-PSR Central Factory Hub(CFH).

CFH Specification 3.1 (*Central Factory Hub*)

Central managing of all services needed for controlling data, devices and robots. In the following paragraphs, several plug-in will be described. In future, a web-based user interface would be useful.

CFH Specification 3.2 (*Benchmarking Plug-in*)

A plug-in to serve general benchmark functionalities or interact with a separated BM software.

CFH Specification 3.3 (*Production Tracking Database Plug-in*)

An automatic plug-in with the central database tracking all identifiers, status of finished products, sub-assemblies, stock level etc.

The CFH-related sources and client example can be obtained from the following locations:

- CFH: `https://github.com/industrial-robotics/atwork_central_factory_hub`
- Client example: `https://github.com/industrial-robotics/atwork_refbox_ros_client`

3.5 Benchmarking Equipment in the Environment

ERL-PSR benchmarking is based on the processing of data collected in two ways:

- **internal benchmarking data**, collected by the robot system under test (see Section 4);
- **external benchmarking data**, collected by the equipment embedded into the testbed.

External benchmarking data is generated by the ERL-PSR testbed with a multitude of methods, depending on their nature.

One of the types of external benchmarking data used by ERL-PSR are pose data about robots and/or their constituent parts. To acquire these, ERL-PSR uses a camera-based commercial motion capture system (e.g. NaturalPoint OptiTrack), composed of dedicated hardware and software. Benchmarking data has the form of a time series of poses of rigid elements of the robot (such as the base or the wrist). There is a case where the motion capture system influences the robot perception system. It is the teams' responsibility to check this effect and coordinate with the benchmarking team in order to minimize this effect. Once generated by the OptiTrack system, pose data are acquired and logged by a customized external software system based on ROS (Robot Operating System): more precisely, logged data is saved as *bagfiles* created with the *rosbag* utility provided by ROS. Pose data is especially significant because it is used for multiple benchmarks. There are other types of external benchmarking data that ERL-PSR acquires: however, these are usually collected using devices that are specific to the benchmark. For this reason, such devices are described in the context of the associated benchmark, rather than here.

Finally, equipment to collect external benchmarking data includes any *server* which is part of the testbed and that the robot subjected to a benchmark has to access as part of the benchmark. Communication between servers and robot is performed via the testbed's own wireless network (see Section 4.2).

4 Robots and Teams

The purpose of this section is twofold:

1. It specifies information about various robot features that can be derived from the environment and the targeted tasks. These features are to be considered at least as desirable, if not required for a proper solution of the task. Nevertheless, we will try to leave the design space for solutions as large as possible and to avoid premature and unjustified constraints.

2. The robot features specified here should be supplied in detail for any robot participating in the competition. This is necessary in order to allow better assessment of competition and benchmark results later on.

The description of the robot should be included in the team description paper.

4.1 General Specifications and Constraints on Robots and Teams

Robot Specification 4.1 (*System*)

A competing team may use a single robot or multiple robots acting as a team. It is not required that the robots are certified for industrial use. At least one of the robots entered by a team is capable of:

- *mobility and autonomous navigation.*
- *manipulate and grasp at least several different task-relevant objects. The specific kind of manipulation and grasping activity required is to be derived from the task specifications.*

The robot subsystems (mobility, manipulation and grasping) should work with the environment and objects specified in this rule book.

Robot Specification 4.2 (*Sensor Subsystems*)

Any robot used by a team may use any kind of ***onboard*** *sensor subsystem, provided that the sensor system is admitted for use in the general public, its operation is safe at all times, and it does not interfere with other teams or the environment infrastructure. A team may use the sensor system in the environment provided by the organizer by using a wireless communication protocol specified for such purpose. Sensor systems used for benchmarking and any other systems intended for exclusive use of the organizers are not accessible by the robot system. Teams are not allowed to modify the environment or to install their own embedded devices in the environment, e.g., additional sensors or actuators.*

Robot Specification 4.3 (*Communication Subsystems*)

Any robot used by a team may ***internally*** *use any kind of communication subsystem, provided that the communication system is admitted for use in the general public, its operation is safe at all times, and it does not interfere with other teams or the environment infrastructure. A robot team must be able to use the communication system provided* ***as part of the environment*** *by correctly using a protocol specified for such purpose and provided as part of the scenario.*

Robot Specification 4.4 (*Power Supply*)

Any mobile device (esp. robots) must be designed to be usable with an onboard power supply (e.g. a battery). The power supply should be sufficient to guarantee electrical autonomy for a duration exceeding the periods foreseen in the various benchmarks, before recharging of batteries is necessary. Charging of robot batteries must be done outside of the competition

environment. The team members are responsible for safe recharging of batteries. If a team plans to use inductive power transmission devices for charging the robots, they need to request permission from the event organizers in advance and at least three months before the competition. Detailed specifications about the inductive device need to be supplied with the request for permission.

Robot Constraint 4.1 (*Computational Subsystems*)

Any robot or device used by a team as part of their solution approach must be suitably equipped with computational devices (such as onboard PCs, microcontrollers, or similar) with sufficient computational power to ensure safe autonomous operation. Robots and other devices may use external computational facilities, including Internet services and cloud computing to provide richer functionalities, but the safe operation of robots and devices may not depend on the availability of communication bandwidth and the status of external services.

Robot Constraint 4.2 (*Safety and Security Aspects*)

For any device a team brings into the environment and/or the team area, and which features at least one actuator of any kind (mobility subsystems, robot manipulators, grasping devices, actuated sensors, signal-emitting devices, etc.), a mechanisms must be provided to immediately stop its operation in case of an emergency (emergency stop). For any device a team brings into the environment and/or the team area, it must guarantee safe and secure operation at all times. Event officials must be instructed about the means to stop such devices operating and how to switch them off in case of emergency situations.

Robot Constraint 4.3 (*Operation*)

In the competition, the robot should perform the tasks autonomously. An external device is allowed for additional computational power. It must be clear at all times that no manual or remote control is exerted to influence the behavior of the robots during the execution of tasks.

Robot Constraint 4.4 (*Environmental Aspects*)

Robots, devices, and apparatus causing pollution of air, such as combustion engines, or other mechanisms using chemical processes impacting the air, are not allowed. Robots, devices, and any apparatus used should minimize noise pollution. In particular, very loud noise as well as well-audible constant noises (humming, etc.) should be avoided. The regulations of the country in which a competition or benchmark is taking place must be obeyed at all times. The event organizers will provide specific information in advance, if applicable. Robots, devices, and any apparatus used should not be the cause of effects that are perceived as a nuisance to humans in the environment. Examples of such effects include causing wind and drafts, strong heat sources or sinks, stenches, or sources for allergic reactions.

4.2 Benchmarking Equipment in the Robots

Preliminary Remark: Whenever teams are required to install some element provided by ERL-PSR on (or in) their robots, such element will be carefully chosen in order to minimize the work required from teams and the impact on robot performance.

Hardware As a general rule, ERL-PSR does not require that teams install additional robotic hardware on their robots. Moreover, permanent change to the robot's hardware is never required. However, ERL-PSR may require that additional standard PC hardware (such as an external, USB-connected hard disk for logging) is temporarily added to the robot in order to collect

internal benchmarking data. When this is the case, the additional hardware is provided by ERL-PSR during the competition, and its configuration for use is either automatically performed by the operating system, or very simple.

To allow the acquisition of external benchmarking data about their pose, robots need to be fitted with special reflective markers, mounted in known positions. The teams will be required to prepare their robots to ease the mounting of the markers. Teams will be required to provide the geometric transformation from the position of the marker to the odometric center of the robot[1].

Software ERL-PSR may require that robots run ERL-PSR-provided (or publicly available) software during benchmarks. A typical example of such software is a package that logs data provided by the robot, or a client that interfaces with a ERL-PSR server via the wireless network of the testbed. Whenever a team is required to install and run such a package, it will be provided as source code, its usage will be most simple, and complete instruction for installation and use will be provided along with it. All ERL-PSR software is written to have a minimal impact on the performance of a robot, both in terms of required processing power and in terms of (lack of) interaction with other modules. When required by a benchmark, the relevant ERL-PSR software to be run by participating robots is provided well in advance with respect to the competition.

ERL-PSR will make any effort to avoid imposing constraints on the teams participating to the competition in terms of software architecture of their robots. This means that any provided piece of software will be designed to have the widest generality of application. However, this does not mean that the difficulty of incorporating such software into the software architecture of a robot will be independent from such architecture: for technical reasons, differences may emerge. A significant example is that of software for data logging. At the moment, it appears likely that any such software by ERL-PSR will be based on the established *rosbag* software tool, library and file format. As rosbag is part of ROS (Robot Operating System), robots based on ROS can use it to log data without any modification; on the contrary, robots not using ROS will be required to employ the rosbag library to create rosbag files (*bagfiles*) or to develop ad-hoc code to convert their well established logging format into the rosbag one by using the rosbag API. If this will be the case, ERL-PSR will provide tools to ease the introduction of software modules for creation of bagfiles into any software architecture; yet, teams not using ROS will probably have to perform some additional work to use such tools.

4.3 Robot Communication with Benchmarking Equipment

For some types of internal benchmarking data (i.e. provided by the robot), logging is done on board the robot, and data are collected after the benchmark (for instance, via USB stick). Other types of internal benchmarking data, instead, are communicated by the robot to the testbed during the benchmark. In such cases, communication is done by interfacing the robot with standard wireless network devices (e.g. IEEE 802.11n) that are part of the testbed, and which therefore become a part of the benchmarking equipment of the testbed. However, it must be noted that network equipment is not strictly dedicated to benchmarking: for some benchmarks, in fact, the WLAN may be (or exclusively) used to perform interaction between the robot and the testbed.

Due to the need to communicate with the testbed via the WLAN, all robots participating to the ERL-PSR competition are required to:

1. possess a fully functional IEEE 802.11n network interface[2];

[1] Benchmarking data related to poses will refer to the marker position: this is why additional information is required to know the position of the base.

[2] It must be stressed that full functionality requires that the network interface must not be hampered by electromagnetic obstacles, for instance by mounting it within a metal structure and/or by employing inadequate antenna arrangements. Network spectrum in the competition area is typically very crowded, and network equipment with

2. be able to keep the wireless network interface permanently connected to the testbed WLAN for the whole duration of the benchmarks.

4.4 YAML Data File Specification

The subsequent paragraphs specify the YAML file format that can be converted to ROS bag files. This closely follows the data items described in D-2.1.7 [**?**]. The YAML format was chosen because it is a simple format, easy to produce without using any special library. Furthermore, the ROS messages format is already defined: as produced by the `rostopic echo` command.

4.4.1 File Format

The YAML file should be composed of a single list of messages. Each message should have four items:

- `topic` - The topic name.
- `secs` - Timestamp of the message, in number of seconds since 1970.
- `nsecs` - Nanoseconds component of the timestamp.
- `message` - The message, according to the topic type.

The message should be formatted in YAML, according to its structure. This is the same as the output of `rostopic echo`. However, binary fields may be specified in base 64 encoding for much smaller files. You can copy the file `src/base64.hpp` to your project, it depends only on boost to encode base 64.

And example for a file generated according to above specification could look as follows:

```
- topic: pose2d
  secs: 1397024209
  nsecs: 156423000
  message:
   x: 5.5
   y: 6
   theta: 6.4
- topic: image
  secs: 1397024210
  nsecs: 53585000
  message:
   header:
     seq: 306
     stamp:
       secs: 1397024210
       nsecs: 53585000
     frame_id: ''
   height: 4
   width: 4
   encoding: bgr8
   is_bigendian: 0
   step: 12
   data:
     !!binary JaU8JY0kGXUIAZ0UDWzgAXjgAb0kIglwbkGsnkWwoiWUfiGUhi2olhmUgc1YRaUw
```

impaired radio capabilities may not be capable of accessing the testbed WLAN, even if correctly working in less critical conditions.

4.4.2 YAML-to-ROSbag Conversion Tool

A tool to convert European Robotics League YAML files into ROS bag files is available at the RoCKIn Github repository:

`https://github.com/rockin-robot-challenge/benchmark_and_scoring_converter`

5 Task Benchmarks

Details concerning rules, procedures, as well as scoring and benchmarking methods, are common to all task benchmarks.

Rules and Procedures Every run of each of the task benchmark will be preceded by a safety-check, outlined as follows:

1. The team members must ensure and inform at least one of the organizing committee (OC) member, present during the execution of the task, that they have an emergency stop button on the robot which is fully functional. Any member of the OC can ask the team to stop their robot at any time which must be done immediately.
2. A member of the OC present during the execution of the task will make sure that the robot complies with the other safety-related rules and robot specifications presented in Section 4.

All teams are required to perform each task according to the steps mentioned in the rules and procedures sub-subsections for the tasks. During the competition, all teams are required to repeat the task benchmarks several times. On the last day, only a selected number of top teams will be allowed to perform the task benchmarks again. Maximum time allowed for one task benchmark is 10 minutes.

Acquisition of Benchmarking Data Following some general notes on the acquisition of benchmarking data are described. They are valid for all task benchmarks, as well as for the functional benchmarks.

- **Calibration parameters** Important! Calibration parameters for cameras must be saved. This must be done for other sensors (e.g., Kinect) that require calibration as well, if a calibration procedure has been applied instead of using the default values (e.g., those provided by OpenNI).
- **Notes on data saving** The specific data that the robot must save is described in the benchmark section. In general some data streams (those with the highest bitrate) must be logged only in the time intervals when they are actually used by the robot to perform the activities required by the benchmark. In this way, system load and data bulk are minimized. For instance, whenever a benchmark includes object recognition activities, video and point cloud data must be logged by the robot only in the time intervals when it is actually performing object recognition.
- **Use of data** The logged data is not used during the competition. In particular, it is not used for scoring. The data is processed by European Robotics League consortium members after the end of the competition. It is used for in-depth analysis and/or to produce datasets to be published for the benefit of the robotics community.
- **Where and when to save data** Robots must save the data as specified in the section "Acquisition of Benchmarking Data" of their respective TBM/FBM on a USB stick provided by ERL-PSR staff. The USB stick is given to the team immediately before the start of the benchmark, and must be returned (with the required data on it) at the end of the benchmark. Each time a teams robot executes a benchmark, the team must:
  1. Create, in the root directory of the USB stick, a new directory named
     - NameOfTheTeam_FBMx_DD_HH-MM (for FBM) or
     - NameOfTheTeam_TBMx_DD_HH-MM (for TBM)
  2. Configure the robot to save the data files in these directories.

In the filenames above x denotes the number of the benchmark, DD is the day of the month and HH, MM represent the time of the day (hours and minutes).

All files produced by the robot that are associated with the execution of the benchmark must be written in this directory. Please note that a new directory must be created for each benchmark executed by the robot. This holds true even when the benchmark is a new run of one that the robot already executed.

During the execution of the benchmark, the following data will be collected[3]. In brackets the expected ROS topics are named. Corresponding data types can be stored in a YAML file (see Section 4.4) or rosbag. Following the list of **offline data** to be logged:

Topic	Type	Frame Id	Notes
/rockin/robot_pose[4]	geometry_msgs/PoseStamped	/map	10 Hz
/rockin/marker_pose[5]	geometry_msgsPoseStamped	/map	10 Hz
/rockin/trajectory[6]	nav_msgs/Path	/map	Each (re)plan
/rockin/<device>/image[7]	sensor_msgs/Image	/<device>_frame	–
/rockin/<device>/camera_info[8]	sensor_msgs/CameraInfo	–	–
/rockin/depth_<id>/pointcloud[9]	sensor_msgs/PointCloud2	/depth_<id>_frame	–
/rockin/scan_<id>[10]	sensor_msgs/LaserScan	/laser_<id>_frame	10-40Hz
tf[11]	tf	–	–

Some robots might not have some of the sensors or they might have multiple instances of the previous data (e.g., multiple rgb cameras or multiple laser scanner), in this case you append the number of the device to the topic and the frame (e.g., /erlir/scan_0 in /laser_frame_0). It is possible not to log some of the data, if the task does not require it.

The **online** data part can be found in the description of the respective benchmark.

Communication with CFH The following steps describe the part of the CFH communication that is applicable for all TBMs.

1. The robot sends a **BeaconSignal** message at least every second.
2. The robot waits for **BenchmarkState** messages. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.
3. The robot waits for an **Inventory** message from the CFH (which is continuously sent out by the CFH) in order to receive the initial distribution of objects and their locations in the environment.

[3] In the following, 'offline' identifies data produced by the robot, and stored locally on the robot, that will be collected by the referees when the execution of the benchmark ends (e.g., as files on a USB stick), while 'online' identifies data that the robot has to transmit to the CFH during the execution of the benchmark. Data marked neither with 'offline' nor 'online' is generated outside the robot.

[4] The 2D robot pose at the floor level, i.e., $z = 0$ and only yaw rotation.

[5] The 3D pose of the marker in 6 degrees of freedom.

[6] Trajectories planned by the robot including when replanning.

[7] Image processed for object perception; <device> must be any of stereo_left, stereo_right, rgb; if multiple devices of type <device> are available on your robot, you can append "_0", "_1", and so on to the device name: e.g., "rgb_0", "stereo_left_2", and so on.

[8] Calibration info for /erlir/<device>/image.

[9] Point cloud processed for object perception; <id> is a counter starting from 0 to take into account the fact that multiple depth camera could be present on the robot: e.g., "depth_0", "depth_1", and so on.

[10] Laser scans, <id> is a counter starting from 0 to take into account the fact that multiple laser range finders could be present on the robot: e.g., "scan_0", "scan_1", and so on.

[11] The tf topic on the robot; the tf tree needs to contain the frames described in this table properly connected through the /base_frame which is the odometric center of the robot.

4. The robot waits for an **Order** message from the CFH (which is sent out continuously by the CFH) in order to receive the actual task, i.e., where the objects should be at the end.
5. The task benchmark ends when all objects are at their final location as specified in the **Order** message. After that the robot sends a message of type **BenchmarkFeedback** to the CFH with the *phase_ to_ terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *state* field has changed.

The messages to be sent and to be received can be seen on the Github repository located at [**?**].

Scoring and Ranking Evaluation of the performance of a robot according to this task benchmark is based on performance equivalence classes and they are related to the fact that the robot has done the required task or not.

The criterion defining the performance equivalence class of robots is based on the concept of *tasks required achievements*. While the ranking of the robot within each equivalence class is obtained by looking at the performance criteria. In particular:

- The performance of any robot belonging to performance class N is considered as better than the performance of any robot belonging to performance class M whenever $M < N$
- Considering two robots belonging to the same class, then a penalization criterion (penalties are defined according to task performance criteria) is used and the performance of the one which received less penalizations is considered as better
- If the two robots received the same amount of penalizations, the performance of the one which finished the task more quickly is considered as better (unless not being able to reach a given achievement within a given time is explicitly considered as a penalty).

Performance equivalence classes and in-class ranking of the robots are determined according to three sets:

- A set A of **achievements**, i.e., things that should happen (what the robot is expected to do).
- A set PB of **penalized behaviors**, i.e., robot behaviors that are penalized, if they happen, (e.g., hitting furniture).
- A set DB of **disqualifying behaviors**, i.e., robot behaviors that absolutely must not happen (e.g., hitting people).

Scoring is implemented with the following 3-step sorting algorithm:

1. If one or more of the elements of set DB occur during task execution, the robot gets disqualified (i.e. assigned to the lowest possible performance class, called class 0), and no further scoring procedures are performed.
2. Performance equivalence class X is assigned to the robot, where X corresponds to the number of achievements in set A that have been accomplished.
3. Whenever an element of set PB occurs, a penalization is assigned to the robot (without changing its performance class).

One key property of this scoring system is that a robot that executes the required task completely will always be placed into a higher performance class than a robot that executes the task partially. Moreover the penalties do not make a robot change class (also in the case of incomplete task).

Penalized Behaviors and Disqualifying Behaviors The penalizing behaviors for all task benchmarks are:

- The robot collides with obstacles in the testbed.
- The robot drops an object.
- The robot stops working.
- The robot accidentally place an object on top of another object.

The disqualifying behaviors for all task benchmarks are:

- The robot damages or destroys the objects requested to manipulate.
- The robot damages the testbed.

The achievements for each task are unique and are described in the respective task benchmark section.

5.1 Task *Fill a Box with Parts for Manual Assembly (Shared with RoboCup@Work)*

This task reflects one of the primary requisites of a mobile robotic service assistant, i.e., to work together with humans. In this case the goal is to assist humans at a manual assembly workstation. The robots have to deal with flexible task specifications, especially concerning information about object constellations in source and target locations, and task constraints such as limits on the number of objects allowed to be carried simultaneously, etc.

5.1.1 Task Description

The robot has to pick up several parts from different source locations and deliver them to several destination locations.

5.1.2 Feature Variation

All objects defined in Section 3.2.1 will be used in this task. Several containers (see Section **??**) can be present in the environment and are always associated with a workstation or shelf. It is possible that more than one container is placed on top of a single workstation. Currently, a container itself does to not need to be manipulated or transported by the robot.

For any major competition in which ERL-PSR will be present, additional objects may be used. Those objects can be found in the rule descriptions of the major competition (e.g. for RoboCup 2016)

5.1.3 Input Provided

The team will be provided with the following information:

- The list of possible parts used in the task;
- The list of source service areas
- The list of destination service areas

5.1.4 Expected Robot Behavior or Output

The task execution is triggered by the robot receiving the inventory- and order list from the CFH. The robot proceeds with collecting the parts from the source locations and transports them to the destination service areas.

5.1.5 Procedures and Rules

There can be multiple obstacles and barrier tapes (color: yellow/black) present in the environment that might block the direct path of the competing robot. The robot must avoid all obstacles or even other robots during the execution of the complete task.

Step 1 The robot will receive an order and inventory from the CFH containing a list of objects to be collected and delivered.

Step 2 The robot must plan the best path to the designated workstation, passing through each storage area where the required objects for a requested product in the list can be found.

Step 3 The robot must execute the above path, collect the objects and then deliver them to the designated destination locations.

Step 4 The Steps 2 and 3 above must be done for all the objects in the list mentioned above in Step 1.

5.1.6 Communication with CFH

For this task benchmark the robot does not have to control any networked device in the environment. Thus, the robot is only supposed to receive the static inventory and order information once. Currently, the robot does not need to update the inventory information.

5.1.7 Acquisition of Benchmarking Data

General information on the acquisition of benchmarking data is described in Section 5.

Online Data

- No online benchmarking data has to be sent to the CFH during this task benchmark.

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/notification[12]	std_msgs/String	–	–

5.1.8 Scoring and Ranking

Evaluation of the performance of a robot according to this task benchmark is based on performance equivalence classes. Classes are defined in dependence to the number of parts of the product to be assembled actually provided by the robot to the human worker and their order according to the desired one.

Achievements The set A of achievements for this task consists of:

- The robot communicates with the CFH throughout the test;
- The team submits the benchmarking data by the end of the test;
- The robot picks up a required object from its storage location;
- The robot places the required objects in its destination area;
- The robot has correctly collected and delivered all objects

[12]The string with the notification of the perceived object should be in a tab separated string: CLASS OBJECT_ID X Y THETA

6 Functionality Benchmarks

Communication with CFH Every functionality benchmark will be preceded by a safety-check similar to that described for the task benchmark procedures. All teams are required to perform each functionality benchmark according to the steps mentioned in their respective section. During the competition, all teams are required to repeat it the functionality benchmark several times. On the last day, only a selected number of top teams will be allowed to perform it.

6.1 Object Perception Functionality

6.1.1 Functionality Description

This functionality benchmark evaluates the capabilities of a robot to extract information about observed objects. Objects presented to the robot in this functionality benchmark are based on the the ERL-PSR factory scenario. Teams are provided with a list of objects (object instances), subdivided into classes (see Section 6.1.3). In this benchmark, the robot is expected to detect and estimate the object's class and identity. Objects that are used here are described in Section 3.

6.1.2 Feature Variation

For this benchmark, the variation space for object features is represented by the (known) set of objects that the robot may be presented with. Variation space for object location is the surface of the benchmarking area where objects are located (see Section 6.1.3).

6.1.3 Input Provided

The set of individual objects that will actually be presented to the robot during the execution of the functionality benchmark is a subset of a larger set of available objects ("object instances"). Object instances are subdivided into classes of objects that have one or more properties in common ("object classes"). Objects of the same class share one or more properties, not necessarily related to their geometry (for instance, a class may include objects that share their application domain). Each object instance and each object class is assigned a unique ID.

The team does not know which object instances will actually be presented to the robot during the benchmark. However, the team will be provided with the following information:

- Descriptions/models of all the object instances in the form of 3D textured models;
- Subdivision of the object instances into object classes (for instance: profiles, screws, joints);
- Reference systems associated to the table surface and to each object instance (to be used to express object poses).

Object descriptions will be expressed according to widely accepted representations, well in advance of the competition.

6.1.4 Expected Robot Behavior or Output

The robot, for each object presented to it, must perform the following:

- Object detection: perception of the presence of an object on the table and association between the perceived object and one of the object classes (see Section 6.1.3).
- Object recognition: association between the perceived object and one of the object instances belonging to the selected class (see Section 6.1.3).

The following list of classes and instances of objects are going to be used in the object perception functionality benchmark:

- Industrial
- Chocolates
- T-LESS

6.1.5 Procedures and Rules

The maximum time allowed for one functionality run is 2 minutes. A run is defined as the detection, recognition and localization of one object.

Step 1 An object of unknown class and unknown instance will be placed in front of the robot

Step 2 The robot must determine the objects class, its instance within that class as well as the 3D pose of the object and save it in the given format (see Section 6.1.7)

Step 3 The preceding steps are repeated until 10 runs have been completed.

6.1.6 Communication with CFH

For this functionality benchmark the robot does not have to control any networked device in the environment. Only the communication as described below is necessary:

Step 1 The robot sends a **BeaconSignal** message at least every second.

Step 2 The robot waits for a **BenchmarkState** message. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.

Step 3 As soon as the robot has finished perceiving the object, it sends a message of type **BenchmarkFeedback** to the CFH with the required results and the *phase_to_terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *state* field has changed.

Step 4 The robot continues with Step 2.

Step 5 The functionality benchmark ends when the **BenchmarkState**'s *state* field is equal to FINISHED.

The messages to be sent and to be received can be seen on the Github repository located at [**?**].

6.1.7 Acquisition of Benchmarking Data

General information on the acquisition of benchmarking data is described in Section 5. There, the **offline** part of the benchmarking data can be found.

Online data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- object_class_name (type: string)
- object_instance_name (type: string))

The **BenchmarkFeedback** message can be found at [**?**].

Topic	Type	Frame Id	Notes
/rockin/notification[13]	std_msgs/String	–	–

Offline data The additional information described in the following table has to be logged:

6.1.8 Scoring and Ranking

Evaluation of the performance of a robot according to this functionality benchmark is based on:

1. The number and percentage of correctly identified objects class;
2. The number and percentage of correctly identified objects object name;
3. Execution time (if less than the maximum allowed for the benchmark).

The previous criteria are in order of importance; the first criterion is applied first and teams will be scored according to their accuracy, the ties are broken by using the second one still applying accuracy metrics, and in case of ties we will resort to execution time.

6.2 Manipulation Pick Functionality

6.2.1 Functionality Description

This functionality benchmark assesses the robot's capability of grasping different objects. An object from a known set of possible objects is presented in the test for the robot to be identified and grasped. After identifying the object, the robot needs to perform the grasping motion, lift the object and notify that the object is acquired.

6.2.2 Feature Variation

The objects used in the benchmark will be selected from the list of parts to manipulate as presented in Section 3.2. Additionally, the precise position of the object differs in each test.

6.2.3 Input Provided

The team will be provided with the following information:

- The list of possible objects used in the functionality benchmark.
- Possible placement of each object used in the functionality benchmark.

6.2.4 Expected Robot Behaviour or Output

The robot is placed in front of the test area (a planar surface). One object will be placed in the test area and the robot will identify the object and move its end effector on top of it. Afterwards, the robot performs the grasping motion of the object and notifies that the grasping has occurred. The task is repeated with different objects.

[13]The string with the notification of the perceived object should be in a tab separated string: CLASS OBJECT_ID X Y THETA

6.2.5 Procedures and Rules

The maximum time allowed for one functionality run is 4 minutes (30 seconds for preparation and 210 seconds for execution). A run consists of (1) a preparation phase where the robot is going to its initial configuration and releases a previously grasp object and (2) an execution phase in which the robot detects, localizes, recognizes and manipulates one object.

Step 1 An object of unknown class and unknown instance will be placed on a table in front of the robot.

Step 2 The robot must determine the object's class and its instance within that class.

Step 3 The robot must grasp the object, lift it, and notify that grasping has occurred.

Step 4 The robot must keep the grip for a given time while the referee verifies the correct manipulation of the object.

Step 5 The preceding steps are repeated with five different objects.

For each presented object, the robot must produce the result consisting of:

- object class name [string], e.g. containers.
- object instance name [string], e.g. EM-02.

6.2.6 Communication with CFH

For this functionality benchmark the robot does not have to control any networked device in the environment. Only the communication as described below is necessary:

Step 1 The robot sends a **BeaconSignal** message at least every second.

Step 2 The robot waits for a **BenchmarkState** message. It starts the preparation procedure when the *phase* field is equal to PREPARATION and the *state* field is equal to RUNNING.

Step 3 As soon as the robot finishes the preparation phase, it sends a message of type **BenchmarkFeedback** to the CFH with the *phase_to_terminate* field set to PREPARATION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 4 The robot waits for a **BenchmarkState** message. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.

Step 5 As soon as the robot has finished manipulating the object, it sends a message of type **BenchmarkFeedback** to the CFH with the required results and the *phase_to_terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 6 The robot continues with Step 2.

Step 7 The functionality benchmark ends when the **BenchmarkState**'s *phase* field is equal to EXECUTION and the *state* field is equal to FINISHED.

The messages to be sent and to be received can be seen on the Github repository located at [**?**].

6.2.7 Acquisition of Benchmarking Data

General information on the acquisition of benchmarking data is described in Section 5. There, the **offline** part of the benchmarking data can be found.

Online data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- grasp_notification (type: bool)
- object_class_name (type: string)
- object_instance_name (type: string)

The **BenchmarkFeedback** message can be found at [**?**].

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/grasping_pose[14]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/gripper_pose[15]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/arm_joints[16]	geometry_msgs/JointState	/base_link	10 Hz

6.2.8 Scoring and Ranking

Evaluation of the performance of a robot according to this functionality benchmark is based on:

1. Number and percentage of correctly identified objects;
2. Number and percentage of correctly grasped objects (the object stops touching the table, see definition below);
3. Execution time (if less than the maximum allowed for the benchmark).

The scoring of teams is based on the number of objects correctly grasped. A correct grasp is defined as the object being lifted from the table so to be possible for the judge to pass a hand below it. For a grasping to be "correct" the position has to be kept for at least 5 seconds from the time the judge has passes the hand below the object. The time the judge will require to verify the lifting of the object might be up to 10 seconds. In case of ties the overall execution time will be taken into account.

6.3 Manipulation Place Functionality

6.3.1 Functionality Description

This functionality benchmark assesses the robot's capability of placing different objects. An object from a known set of possible objects is presented in the test for the robot to be placed. After grasping the object, the robot needs to perform the placing motion, lift the object and place the object in a particular container/box.

6.3.2 Feature Variation

The objects used in the benchmark will be selected from the list of parts to manipulate as presented in Section 3.2. Additionally, the precise position of the object placement differes in each test.

[14] Pose of the grasping position on the object.
[15] Pose of the gripper.
[16] Joints data

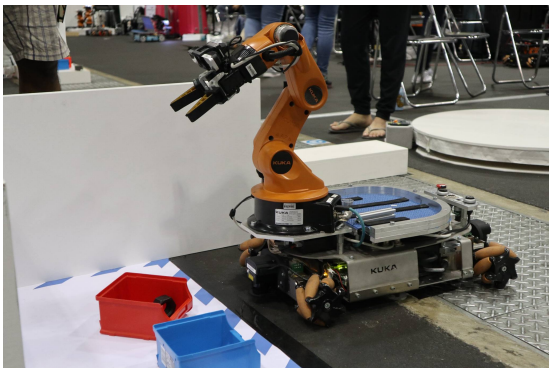


Figure 9: Manipulation Place Functionality.

6.3.3 Input Provided

The team will be provided with the following information:

- The list of possible objects used in the functionality benchmark.
- Possible placement of each object used in the functionality benchmark.

6.3.4 Expected Robot Behaviour or Output

The robot is placed in front of the test area (a planar surface). One object will be placed on the robot or in its gripper. The robot now has to look the test area for any container to be placed. The robot performs the placing motion of the object and places the object in particular container in front of it. After placement it notifies the CFH.

6.3.5 Procedures and Rules

The maximum time allowed for one functionality run is 4 minutes (30 seconds for preparation and 210 seconds for execution). A run consists of (1) a preparation phase where the robot is going to its initial configuration and grasp object and (2) an execution phase in which the robot detects, localizes, recognizes and manipulates one object.

Step 1 An object of known class and known instance will be placed on the robot.

Step 2 The robot must scan the test area and find a container.

Step 3 The robot must place the object inside the container and notify that placing has occurred.

Step 5 The preceding steps are repeated with five different objects.

6.3.6 Communication with CFH

For this functionality benchmark the robot does not have to control any networked device in the environment. Only the communication as described below is necessary:

Step 1 The robot sends a **BeaconSignal** message at least every second.

Step 2 The robot waits for a **BenchmarkState** message. It starts the preparation procedure when the *phase* field is equal to PREPARATION and the *state* field is equal to RUNNING.

Step 3 As soon as the robot finishes the preparation phase, it sends a message of type **BenchmarkFeedback** to the CFH with the *phase_to_terminate* field set to PREPARATION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 4 The robot waits for a **BenchmarkState** message. It starts the benchmark execution when the *phase* field is equal to EXECUTION and the *state* field is equal to RUNNING.

Step 5 As soon as the robot has finished manipulating the object, it sends a message of type **BenchmarkFeedback** to the CFH with the required results and the *phase_to_terminate* field set to EXECUTION. The robot should do this until the **BenchmarkState**'s *phase* and *state* fields have changed.

Step 6 The robot continues with Step 2.

Step 7 The functionality benchmark ends when the **BenchmarkState**'s *phase* field is equal to EXECUTION and the *state* field is equal to FINISHED.

The messages to be sent and to be received can be seen on the Github repository located at [**?**].

6.3.7 Acquisition of Benchmarking Data

General information on the acquisition of benchmarking data is described in Section 5. There, the **offline** part of the benchmarking data can be found.

Online data In order to send online benchmarking data to the CFH, the robot has to use the **BenchmarkFeedback** message. The message contains:

- grasp_notification (type: bool)

The **BenchmarkFeedback** message can be found at [**?**].

Offline data The additional information described in the following table has to be logged:

Topic	Type	Frame Id	Notes
/rockin/grasping_pose[17]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/gripper_pose[18]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/arm_joints[19]	geometry_msgs/JointState	/base_link	10 Hz

6.3.8 Scoring and Ranking

Evaluation of the performance of a robot according to this functionality benchmark is based on:

1. Number and percentage of correctly grasped objects (the object stops touching the table, see definition below);
2. Execution time (if less than the maximum allowed for the benchmark).

The scoring of teams is based on the number of objects correctly placed. A correct place is defined as the object being completely inside the container. In case of ties the overall execution time will be taken into account.

[17] Pose of the grasping position on the object.
[18] Pose of the gripper.
[19] Joints data

6.4 Exploration Functionality (Shared with RoboCup@Work)

6.4.1 Functionality Description

This benchmark evaluates the robot exploration and navigation capability.

6.4.2 Feature Variation

This benchmark has the following feature variation:

- Distinct starting points, waypoints, and goal positions.
- Different number of waypoints to reach the goal.
- Different number of obstacles blocking the path.
- Different arena setup

6.4.3 Input Provided

The robot will receive the following information:

- The starting position.
- A single location the robot must visit.
- The landmark for identifying the location the robot has to visit (QR code or color coding)

6.4.4 Expected Robot Behavior or Output

Teams are required to set their robot on a specific starting position (that will be given to the teams before each run). Then, the robot receives, through the CFH, the start signal, as well as a location name that it must reach. The robot must notify the CFH when it has reached the location. The evaluation of the navigation will take into account the following two points:

- The time spent by the robot to locate the location.
- The number of times that the robot hits each obstacle. If the robot hits the same obstacle more than once, it will count as multiple hits.

The functionality benchmark ends as soon as one of the following situations occurs:

- The robot reaches the specified location.
- The time available for the functionality benchmark expires.
- The robot damages any of the obstacles.
- The robot pushes or continually touches an obstacle for more than 3 seconds.
- The robot forces its path through an obstacle.

6.4.5 Procedures and Rules

All teams are required to perform this functionality benchmark according to the steps mentioned below. The task must be performed exclusively in autonomous mode. No teleoperation is allowed. Teams will have up to ten minutes to complete the functionality benchmark. Two minutes to move the robot to the correct starting position plus eight minutes to do the benchmarking.

Step 1 The team is required to start their robot on a pre-defined starting position. This starting position will be given to the teams before each run.

Step 2 When the procedure starts, the robot receives a the location it has to go.

Step 3 Robot explores the arena and when it reaches the desired location sends back a signal to the CFH. The robot must avoid hitting any obstacle it encounters in its path.

Step 3 The procedure stops when the robot notifies it has reached the goal position, when the time given to complete the test expires, or when the robot hard-hits an obstacle.

6.4.6 Acquisition of Benchmarking Data

This functionality benchmark will be fully automated (no human operation will be allowed) and, for that, the robot has to communicate with the CFH. It will receive the target location from the CFH and it must send back a signal, each time it reaches a waypoint.

Topic	Type	Notes
/roaw_cfh/goal	geometry_msgs/Pose2D	List of waypoints, sent by the CFH to the robot, when starting the task.
/roaw_cfh/reached_waypoint	roaw_cfh_comm_ros/UInt8	Message sent by the robot to the CFH, when reaching a point. It must include the number of the respective waypoint in the sequence (starting from zero).

6.4.7 Scoring and Ranking

Evaluation of the performance of the robot(a.k.a scoring) in the benchmark is based on:

1. Time taken by the robot to find the location.

