Funded by the European Union

Robot Competitions Kick Innovation in Cognitive Systems and Robotics
FP7-ICT-601012

General evaluation criteria, modules and metrics for benchmarking through competitions

---

Deliverable:	D-1.2
Due Date:	June 30, 2014
Latest Update:	July 10, 2014
Revision:	1.0

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

Contents

Overview

This Deliverable describes the main elements defining the RoCKIn Competitions. The foundational concepts for such elements were presented, in a general way, by Deliverable D1.1. Deliverable D1.2 brings such concepts to actuality by providing the guidelines for the setup of the RoCKIn Competitions. To do this, D1.2 exploits both the theoretical work and the practical experience done by project RoCKIn after the publication of D1.1. Special importance is taken by the RoCKIn Camp 2014 (Rome, January 26th-30th, 2014), where both the general approach and the actual implementation of Consortium ideas were successfully tested.

The practical implementation of the criteria, modules, and metrics presented here is provided by the Rulebooks for the 2014 RoCKIn Competitions (i.e., Deliverables D2.1.1 and D2.1.4). While D1.2 comes later in the life of the RoCKIn project, the concepts described in this document had a strong influence on the design of the Competitions. The objective of this document is that of providing both the methodological justification behind such design, and the details needed to run the RoCKIn Benchmarking Competitions.

The following of this document is organized as follows. Chapter 1 explores the possibilities and limitations inherent in RoCKIn's use of robot competitions as benchmarking tools. Chapter 2 describes the particular approach towards *benchmarking competitions* chosen by RoCKIn, based on the idea of combining *Task Benchmarks* and *Functionality Benchmarks*. Finally, Chapters 3 and 4 describe scoring mechanisms and metrics for Task Benchmarks and Functionality Benchmarks respectively.

Chapter 1

Benchmarking Through Competitions

1.1 Robot competitions as benchmarking tools

In recent years, a point of view that considers robotic competitions as experiments has emerged both within the academic community and at the level of the European Commission, as competitions are considered excellent vehicles for advancing the state of the art in terms of new algorithms and techniques in the context of a common problem domain [1, 4, 6, 9].

Between competitions and experiments there are some differences: an experiment is aimed at evaluating objectively a specific hypothesis, while a competition is aimed at defining a ranking and winners. Moreover, competitions push to development of solutions, while experiments aim at exploring phenomena and sharing results. Nevertheless, there are a number of reasons for recasting robotics competitions as experiments, considering traditional experimental principles (comparison, repeatability, reproducibility, justification, etc.) as guidelines.

Comparison is to know what has been already done in the field, to avoid the repetition of uninteresting experiments, and to get hints on promising issues to tackle. Reproducibility is the possibility for independent scientists to verify the results of a given experiment by repeating it, while repeatability is the property of an experiment that yields the same outcome when performed at different times and/or in different places. Justification and explanation deal with the necessity of interpreting experimental data in order to derive correct implications. Competitions usually provide controlled environments where approaches to solve specific problems can be compared. Furthermore, they require integrated implementations of complete robotic systems, promoting a new experimental paradigm trying to complement the rigorous evaluation of specific modules in isolation (typical of most laboratory research).

An experiment-oriented perspective on competitions could help to reach the aims of both research and demonstration, while providing a common ground for comparison of different solutions. Reframing competitions as experiments increases their scientific rigour while trying to maintain their distinctive aspects. First of all, competitions are appealing (people like to compete) and they take place with regularity and precise timing, showcasing the current state-of-the-art in research and industry. More importantly, competitions challenge their participants to become creative and to coordinate towards solving difficult problems, attempting to overcome other participants' similar efforts, ultimately leading to the development and sharing of novel solutions. Finally, competitions promote critical analysis of experiments out of labs and they share among participants the cost and effort

of setting up complex experimental installations.

1.2 Difficulties, limitations, and perspectives

Although competitions can be considered as a way of comparing the performance of robots, their character of one-time and, to some degree, unique events puts some limits on the generalizability and replicability of their results, and do not necessarily prove that some robotic systems are better than others in general. As it has been already noticed [5], robotic competitions are not necessarily experimental procedures but, rather, some of their features may not fit an assessed experimental methodology. A competition can be considered as a kind of experiment only if its settings and scoring are properly defined.

Experiments in computing can be intended as the empirical practice to gain and check knowledge about a system and can be conceptualized in five different ways [10], listed below ordered by increasing complexity of execution and, more importantly, of general scientific significance of the results.

- *Feasibility experiment.* It is the loosest use of the term experiment that can be found in many works reporting and describing new techniques and tools. Typically, the term experiment is used in this case with the meaning of empirical demonstration, intended as an existence of proof of the ability to build a tool or a system.

- *Trial experiment.* This requires the evaluation of various aspects of a system using some predefined variables, which are often measured in laboratory, but can occur also in real contexts of use, possibly given some limitations.

- *Field experiment.* It is similar to trial experiment in its aim of evaluating the performance of a system against some measures, but it takes place outside the laboratory in complex socio-technical contexts of use. The system under investigation is thus tested in a live environment, and features such as performance, usability, or robustness, are measured.

- *Comparison experiment.* In this case the term experiment refers to comparing different solutions with the goal of looking for the best solution of a specific problem. Typically, comparison is made in some setup and is based on some measures and criteria to assess the performance. Thus, alternative systems are compared and, to make this comparison as rigorous as possible, standard tests and publicly available data are introduced.

- *Controlled experiment.* It is the golden standard of experimentation of traditional scientific disciplines and refers to the original idea of experiment as controlled experience, where the activity of rigorously controlling (by implementing experimental principles such as reproducibility or repeatability) the factors that are under investigation is central, while eliminating the confounding factors, and allowing for generalization and prediction.

A more complete theoretical work and a more extensive practical experience will be needed to determine if, and under what conditions, real-world robot competitions can be considered as scientific experiments. However, many existing robot competitions are designed in such a way that their position within the above experimental hierarchy is not higher

than *field experiments*. This cannot be considered as a flaw of such competitions, since they are usually not aimed at being recognized as scientific experiments. The aspiration of the RoCKIn project is to provide a *benchmarking competition*, where individual tasks and functionality benchmarks performed during the competition can be classified as *comparison experiments* or even, possibly, *controlled experiments*.

1.3 The RoCKIn benchmarking competitions

Recasting robotic competitions and challenges as scientific experiments offers the opportunity for research groups to benchmark approaches against each other so to enhance the understanding of relative advantages and shortcomings and to have clear measures of success. In particular, by focusing on benchmarking as a way to objectively evaluate the performance of a robotic system or subsystem under controlled circumstances, RoCKIn introduces the idea of *benchmarking experiments* as scientific experiments.

In RoCKIn, benchmarking experiments are a specific way of performing experimental evaluation, of comparing different systems on a common, predefined, setting, and of providing a set of metrics (e.g., measures, numerical scores, pass or fail,, etc.), together with a proper interpretation to perform an objective evaluation, with the goal of enabling the reproducibility and repeatability of experiments.

As reported in the forthcoming Chapter 2, RoCKIn's approach to benchmarking experiments is based on the definition of two separate, but interconnected, types of benchmarks:

- **Functionality Benchmarks**, which evaluate the performance of hardware/software modules dedicated to single, specific functionalities in the context of experiments focused on such functionalities;
- **Task Benchmarks**, which assess the performance of integrated robot systems facing complex tasks that usually require the interaction of different functionalities.

Of the two types, Functionality Benchmarks are certainly the closest to a scientific experiment. This is due to their much more controlled setting and execution. On the other side, these specific aspects of Functionality Benchmarks limit their capability to capture all the important aspects of the overall robot performance in a systemic way. More specifically, emerging system-level properties, such as the quality of integration between modules, cannot be assessed with Functional Benchmarks alone. For this reason, the RoCKIn Competitions integrate them with Task Benchmarks. As reported in Chapter 2, the integration of the "'more scientific" Functional Benchmarks with Task Benchmarks is not only useful from the viewpoint of setting up appealing robot competitions, but also has important methodological justifications.

Chapter 2

Benchmarking Modules and Systems

One of the distinctive features of RoCKIn is that, in trying to define competitions that act as benchmarking tools, it defines two classes of benchmarks: **Task Benchmarks** and **Functionality Benchmarks**. This section is dedicated to explaining the rationale behind this design choice, and its consequences on the structure of the RoCKIn Competitions.

2.1 Rationale and benefits

One of the key limitations of available robot competitions and benchmarks is that they focus on either integrated systems or specific modules. For instance, RoboCup@Home and RoboCup@Work assess the performance of integrated robot systems executing specific tasks in domestic or factory environments, while the Rawseeds Benchmarking Toolkit (`http://www.rawseeds.org/`) is dedicated to benchmarking software modules that implement specific functionalities, such as self-localization, mapping, and SLAM. Unfortunately, focusing only on one of these two approaches (system or module analysis) strongly limits the possibility to gain useful insight about the performance, limitations, and shortcomings of a complete robot system.

In particular, evaluating only the performance of integrated system is interesting for the application, but it does not allow to evaluate the single modules that are contributing to the global performance, nor to put in evidence the aspects needed to push their development forward. On the other side, the good performance of a module does not necessarily mean that it will perform well in the integrated system. For this reason, the RoCKIn Benchmarking Competitions target both aspects, and enable a deeper analysis of a robot system by combining system-level and module-level benchmarking.

System-level and module-level tests do not investigate the same properties of a robot. Module-level testing has the benefit of focusing only on the specific functionality that a module is devoted to, removing interferences due to the performance of other modules which are intrinsically connected at the system level. For instance, if the grasping performance of a mobile manipulator is tested by having it autonomously navigate to the grasping position, visually identify the item to be picked up, and finally grasp it, the effectiveness of the grasping functionality is affected by the actual position where the navigation module stopped the robot, and by the precision of the vision module in retrieving the pose and shape of the item. On the other side, if the grasping test is executed by placing the robot in a predefined known position and by feeding it with precise information about the item to be picked up, the final result will be almost exclusively due to the

performance of the grasping module itself. The first benchmark can be considered as a "system-level" benchmark, because it involves more than one functionality of the robot, and thus has limited worth as a benchmark of the grasping functionality. On the contrary, the latter test can assess the performance of the grasping module with minimal interference from other modules and a high repeatability: it can be classified as "module-level" benchmark.

However, there are issues that module-level testing cannot assess, though they have a major impact on real-world robot performance. For instance, the interactions among the navigation, vision, and grasping modules, which can be considered as disturbance factors in evaluating the performance of the grasping module, take a crucial role in defining the actual performance of a robot system in a real setting where grasping is needed. Performing an experiment that excludes such interactions implies a major loss of useful information. Here lies the specific worth of system-level robot testing, the only way to make system-level properties apparent. We already cited the most obvious of them (i.e., direct interactions among modules), but more subtle ones exist. One of the most important of these, though very difficult to be measured, is the quality of the integration between modules. Indeed, autonomous robots are systems of sufficiently high complexity to make emerging properties having an important role in the definition of the overall performance of the integrated system. For robots, the performance of single modules (as assessed by module-level experiments) does not provide reliable, or sufficient, information about the performance of the complete system once these modules are put together.

For the above described reasons, one of the key elements of the RoCKIn Competitions is the joint presence of module-level and system-level benchmarks. These are called Functionality benchmarks and Task Benchmarks, respectively. Please note that, although focused on the module level, Functionality Benchmarks refer to specific functionalities of a Robot System, they do not address specific software or hardware components of the Robotic System. In this way, they are not linked to a particular robot architecture with specific software modules to implement each functionality.

To pass a Functionality Benchmark, the relevant Functional Module of a Robot System has to face one or more Benchmarking Experiments. Benchmarking Experiments for Functionality Benchmarks are tightly controlled: i.e., they are set up in such a way that the involvement of the elements of the Robot System that are not part of the Functional Module under test is mimimized. To pass a Task Benchmark, a robot system requires to face a single Benchmarking Experiment requiring the execution of a task in a specified Scenario. Benchmarking Experiments for Task Benchmarks are less tightly controlled: the setting is closely controlled, but no strong constraints are set on the execution of the task.

2.2 Matrix representation of functionality and task benchmarks

The considerations reported in Section 2.1 can be represented in matrix form. Let us consider an imaginary, simplified RoCKIn Competition including five tasks (T1, T2, ..., T5). Figure 2.1 describes such imaginary competition as a matrix, showing the tasks as columns while the lines correspond to the functionalities in principle needed to successfully execute the tasks.

For the execution of the whole set of tasks of this imaginary RoCKIn Competition, four

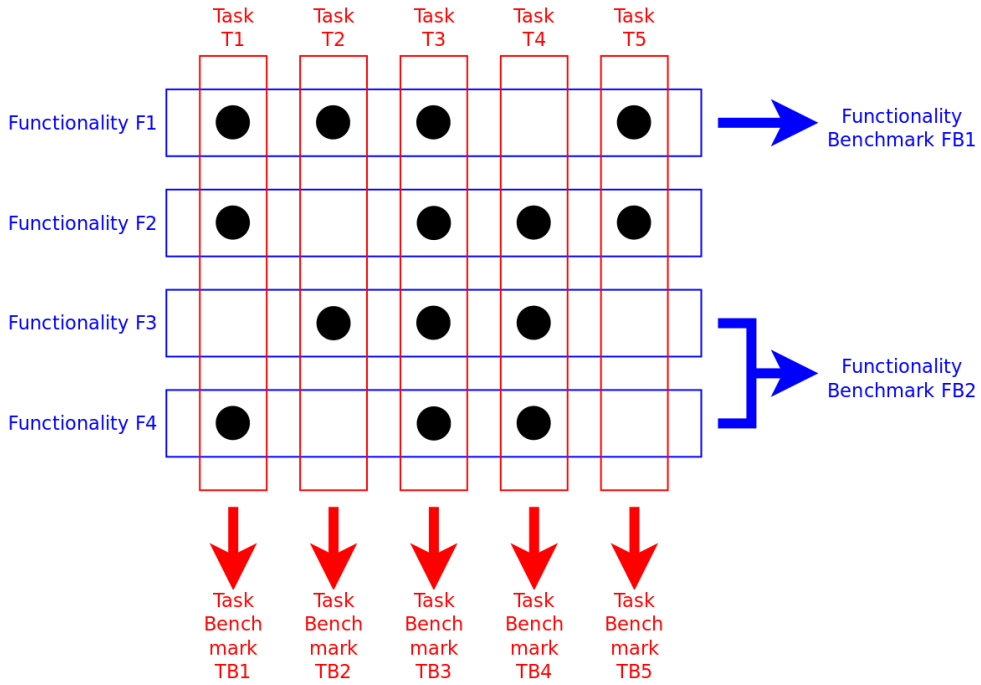

Figure 2.1: Horizontal (functionality) and vertical (task) benchmarking

different functionalities (F1, . . . , F4) are required; however, a single task usually requires only a subset of these functionalities. In Figure 2.1, task Tx requires functionality Fy if a black dot is present at the crossing between column x and row y. For instance, task T2 does not require functionalities F2 and F4, while task T4 does not require functionality F1.

Two additional observations conclude this example. First of all, that the RoCKIn Competitions always include a Task Benchmark for each task, but they do not include necessarily a Functionality Benchmark for each of the functionalities required by the tasks. Secondly, it is possible in some cases to design a Functionality Benchmark so that it tests more than one functionality at the same time, while allowing to separate their contributions. This is what happens to FB2 in Figure 2.1, which tests both functionalities F2 and F4.

2.3 Benchmarks of the 2014 RoCKIn competitions

In this section, we describe the Task Benchmarks and Functionality Benchmarks of the RoCKIn 2014 Competitions[1]. Their relations will be highlighted by using the matrix representation presented in the previous section.

RoCKIn@Home and RoCKIn@Work include three Tasks Benchmarks and three Functionality Benchmarks each. Table 2.1 lists them and points out which functionalities are required for the execution of the tasks, including the functionalities which are required, but for which no Functionality Benchmark has been defined for the very same RoCKIn competition event. All the tasks defined by Table 2.1 will be benchmarked in the 2014 RoCKIn Competitions: in other words, a Task Benchmark has been defined for each

[1]What follows refers to version 1.0 of the RoCKIn@Home rulebook and version 1.1 of the RoCKIn@Work rulebook.

	Catering for Annie's comfort	Welcoming visitors	Getting to know my home	Prepare assembly tray	Plate drilling	Fill box with parts
Task Planning and Scheduling	X	-	-	X	X	X
2D Mapping	-	-	(X)	-	-	-
2D Self-Localization	X	X	X	X	X	X
Path Planning (robot base)	X	X	X	X	X	X
Path Following (robot base)	X	X	X	X	X	X
Object/Face Detection	X	X	X	X	X	X
Object/Face Recognition	X	X	X	X	X	-
Object/Face Localization	X	-	X	X	X	-
Object/Person Tracking	-	X	-	-	-	-
Path Planning (robot arm)	X	-	X	X	X	X
Path Following (robot arm)	X	-	X	X	X	X
Grasp Planning	X	-	X	X	X	X
Grasp Execution	X	(X)	X	X	X	X
Visual servoing (robot arm)	(X)	-	(X)	(X)	(X)	(X)
Input from Humans through Speech	X	X	X	-	-	-
Interactive Communication with Humans	-	X	-	-	-	-

Table 2.1: Matrix representation of the set of tasks and functionalities considered by the 2014 RoCKIn Competition. X = required; (X) = optional (it can be substituted by offline procedures or by alternative functionalities or combinations of functionalities); - = not required (or required, but performed by the testbed).

one of them. In the very same competition some of the functionalities from Table 2.1 will undergo Functionality Benchmarks; those Functionality Benchmarks are reported in Table 2.2, where non-benchmarked functionalities have been omitted and the remaining functionalities grouped according to the Functional Benchmarks used to assess them.

2.4 Scoring framework for modules and systems

In order to design a scientifically-grounded competition, the definition of an appropriate scoring is fundamental. Scoring should be easy, clear, unambiguous, and result in a ranking. In the case of RoCKIn Competitions the problem of defining the scoring is particularly complex, as separate scores for Task Benchmarks and Functionality Benchmarks have been defined. As functionalities vs. tasks dependencies are highly non-linear, cumulative scores are not used. Moreover, interdependencies between tasks and functionalities have to be considered, for instance, to ensure that teams that excel in some subsystems (i.e., teams with good scores according to Functional Benchmarks) do not necessarily end

	Catering for Annie's comfort	Welcoming visitors	Getting to know my home	Prepare assembly tray	Plate drilling	Fill box with parts
FB Planning And Scheduling (@Work)						
Task Planning and Scheduling	X	-	-	X	X	X
FB Object Perception (@Home)						
Object/Face Detection	X	X	X	-	-	-
Object/Face Recognition	X	X	X	-	-	-
Object/Face Localization	X	-	X	-	-	-
FB Object Perception (@Work)						
Object/Face Detection	-	-	-	X	X	X
Object/Face Recognition	-	-	-	X	X	X
Object/Face Localization	-	-	-	X	X	X
FB Object Manipulation (@Home)						
Path Planning (robot arm)	X	-	X	-	-	-
Path Following (robot arm)	X	-	X	-	-	-
Grasp Planning	X	-	X	-	-	-
Grasp Execution	X	(X)	X	-	-	-
Visual servoing (robot arm)	-	-	-	-	-	-
FB Visual Servoing (@Work)						
Path Planning (robot arm)	-	-	-	X	X	X
Path Following (robot arm)	-	-	-	X	X	X
Grasp Planning	-	-	-	X	X	X
Grasp Execution	-	-	-	X	X	X
Visual servoing (robot arm)	-	-	-	(X)	(X)	(X)
FB Speech Understanding (@Home)						
Input from Humans through Speech	X	X	X	-	-	-

Table 2.2: The 2014 RoCKIn Competitions described in matrix form.

up being the winners at the system level (i.e., according to Task Benchmarks), unless they deserve so. While separate rankings for each Functional Benchmark and each Task Benchmark will be given at the Competition, it is nonetheless interesting to provide comprehensive rankings for robot systems that take into consideration their performance in multiple Task and/or Functionality Benchmarks. This is a non-trivial problem, and will be tackled by steps in the following sections of this document.

Chapter 3 is dedicated to the methodology and the metrics designed by RoCKIn to define scoring and ranking for Task Benchmarks. First, scoring for single Task Benchmarks will be considered; then, the approach, and methodology, to the combination of multiple Task Benchmark rankings will be presented.

Chapter 4 does a similar analysis for Functionality Benchmarks. Differently from Task Benchmarks, scoring methodologies change significantly from functionality to functionality. For this reason, defining meaningful criteria to combine rankings associated to different Functionality Benchmarks into a global ranking, as well as combining these ranking with those associated to Task Benchmark, is not easy. Nonetheless, some insight about how this could be done will be provided at the end of the chapter, although the RoCKIn competition for 2014 does not foresee a single ranking combining those.

To make further progress on combining rankings and scores for real-world robot competitions, actual data are needed. For this reason, it will be necessary to wait after the 2014 RoCKIn Competition to see whether and how the proposed criteria provide good results with real-world data, and (possibly) to devise ways to improve upon such criteria.

Another potential improvement to the 2014 RoCKIn Competition concerns the data used to define the rankings. In the 2014 Competition, in fact, only *external benchmarking data* (i.e., data generated by the testbed and/or the referees) will be used. *Internal benchmarking data* (i.e., data generated by the robot systems under test) will be collected but not used for the rankings[2]. After the 2014 Competition, internal benchmarking data will be used to investigate the relations between the performance of the same robot according to Functionality and Task Benchmarks. Among the issues which will be investigated is the opportunity of including some types of internal benchmarking data in the data used for the rankings of the 2015 RoCKIn Competition.

[2]Additional information about these issues can be found in Deliverable 2.1.7.

Chapter 3

Task Benchmarks

This chapter, as introduced in Section 2.4, is concerned with the scoring methodologies and metrics for Task Benchmarks. It is meant to provide both the rationale and practical hints on how RoCKIn tasks will be benchmarked during the RoCKIn 2014 Competition.

3.1 Scoring framework for task benchmarks

The scoring framework for the evaluation of the Task Benchmarks in the RoCKIn@Home and RoCKIn@Work competitions is the same for all Task Benchmarks of RoCKIn@Home and RoCKIn@Work, and it is based on the concept of **performance classes** used for the ranking of robot performance in a specific task.

The performance class that a robot is assigned to is determined by the number of **achievements** (or goals) that the robot reaches during its execution of the task. Within each class (i.e., a performance equivalence class), ranking is defined according to the number of **penalties** assigned to the robot. These are assigned to robots that, in the process of executing the assigned task, make one or more of the errors defined by a task-specific list associated to the Task Benchmark. More formally:

- The ranking of any robot belonging to performance class N is considered as better than the performance of any robot belonging to performance class M whenever $M < N$. Class 0 is the lowest performance class.
- Among robots belonging to the same performance class, a penalization criterion is used to define ranking: the robot which received less penalties is considered as higher in rank.
- Among robots belonging to the same class and with the same number of penalties, the ranking of the one which accomplished the task in a shorter time is considered as higher (unless specific constraints on execution time are given as achievements or penalties).

Performance classes and penalties for a Task Benchmark are indeed task-specific, but they are grouped according to the following three sets (of which here we define the semantics; the actual content is specific to each Benchmark):

- set DB = **disqualifying behaviors**, i.e. things that the robot *must not do*;
- set A = **achievements** (also called goals), i.e., things that the robot *should do*;

- set PB = **penalizing behaviors**, i.e., things that the robot *should not do*.

Once the content of each of the previous sets is provided as part of the specifications of the relevant Task Benchmark, to apply the RoCKIn scoring framework the following 3-step sorting algorithm can be used:

1. if one or more of the disqualifying behaviors of set DB occur during task execution, the robot gets disqualified (i.e., assigned to class 0, the lowest possible performance class), and no further scoring procedures are performed for it;

2. the robot is assigned to performance class X, where X corresponds to the number of achievements of set A which have been accomplished by the robot[1];

3. a penalization is assigned to the robot for each behavior of the robot belonging to set PB that occurs during the execution of the task[2].

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. In fact, penalties do not change the performance class assigned to a robot and only influence intra-class ranking. It is also possible to envisage the use of differently weighted penalties; however, this makes the ranking criteria less easy to understand; therefore weighing will not be implemented unless a clear need for it will emerge after the 2014 RoCKIn Competition.

3.2 An example of task benchmark scoring

This section will act both as example of the scoring framework described by Section 3.1, and as an illustration of the flexibility of such framework in dealing with different situations. Additional operative details may be found in the Rulebooks for the 2014 RoCKIn Competition.

We begin by selecting an example Task Benchmark: for instance the first of RoCKIn@Home, "Catering for Granny Annie's comfort". The task requires that the robot, executing commands issued by voice by Annie[3], first operates a household device, then brings to Annie an object, for instance, the glasses that (as she says to the robot) she left on the kitchen table.

For simplicity, in the following we will focus on the second part of the task (finding and fetching the object) and from a very limited set of achievements an penalties; the sets used to define performance equivalence classes are the following[4].

- set DB (disqualifying behaviors) is composed of the following events:

[1] These sets do not contain repetitions, thus is a given achievement has to be accomplished multiple times, there will be as many distinctive instances of that achievement as required by the task. For instance, if the task requires to serve 4 guests during dinner, there will be 4 items in set A, one for each guest.

[2] Unless clearly specified the errors must occur once to give a penalty, and repeated errors do not cumulate.

[3] Annie is the fictional character of an elderly lady (played by a physical person) that is described in the RoCKIn@Home Rulebook. For some Task Benchmarks, Annie interacts with robots and assigns duties to them.

[4] What follows refers to version 1.0 of the RoCKIn@Home Rulebook. It is possible that the sets will be integrated with new elements in subsequent editions.

  - the robot hits Annie or another person in the environment
  - the robot damages or destroys the objects requested to be manipulated
  - the robot damages the test bed

- set A (achievements) includes the following goals:
  - the robot understands Annie's command
  - the robot operates correctly the right device
  - the robot finds the right object
  - the robot the robot brings to Annie the right object

- set PB (penalizing behaviors) for this task is composed of:
  - the robot bumps into the furniture
  - the robot stops working

The following are some possible examples of performance classification:

a) The robot understands Annie at the first attempt, goes to the kitchen, grasps the right object. Then it brings it back to Annie: performance class 5 (command understood, robot in the kitchen, correct object reported, correct object on the robot, correct object handled to Annie). No penalties.

b) The robot understands Annie after three attempts, goes to the kitchen, but grasps the wrong object. Then it brings it back to Annie: performance class 2 (command understood, robot in the kitchen). No penalties.

c) The robot understands Annie after three attempts, goes to the kitchen and stops: performance class 2 + 1 penalty (command understood, robot in the right place, but robot stopped working). Within class 2, rank is lower wrt (b).

d) The robot understands Annie at the first attempt, goes to the kitchen, but grasps the wrong object. Then it brings it back to Annie, but it bumps into a chair in the meanwhile: performance class 2 + 1 penalty. Within class 2, rank is lower wrt (b); rank wrt (c) depends on execution time.

e) The robot understands Annie at the first attempt, goes to the kitchen, reports the position of the right object, but it cannot grasp it: performance class 3 (command understood, robot in the kitchen, object position reported). No penalties.

Let now increase the level of details in the Task Benchmark description and then refine the performance classes consistently. We are still in the case where the robot goes to Annie and she asks it to bring her a *specific* pair of glasses which she *probably left in the kitchen*. In this case we are interested in the robot exploiting additional information from Annie and the presence of Networked Devices in the environment. The sets used to define performance equivalence classes now become:

- set DB (disqualifying behaviors) is composed of the following events:
  - the robot hits Annie or another person in the environment

  - the robot damages or destroys the objects requested to be manipulated
  - the robot damages the test bed

- set A (achievements) includes the following goals:
  - the robot understands Annie's command
  - the robot reaches the location suggested by Annie
  - the robot reports the position of the right glasses
  - the robot holds the right glasses
  - the robot handles the right glasses to Annie

- set PB (penalizing behaviors) for this task is composed of:
  - the robot bumps into the furniture
  - the robot understands the command only after several (> 2) repetitions
  - the robot does not exploit Networked Robot Systems (NRS) when required
  - the robot does not take Annie's suggestions into account
  - the robot does not leave the lighting in the same state it found it
  - the robot missclassifies objects
  - the robot stops working

The following are some possible examples of performance classification in this more complex Task Benchmark scoring specification assuming all the achievements have been reached:

a) Robot understands Annie only after 3 repetitions (when only two are allowed without penalty): performance class 5 + 1 penalty (+ time in case of ties)

b) Robot is not exploiting NRS devices when required: performance class 5 + 1 penalty (+ time in case of ties)

c) Robot collides with 2 chairs: performance class 5 + 1 penalty (but possibly less time is needed)

d) Robot is not taking into account the suggestions from Annie: class 5 + 1 penalty (+ possibly more time is needed in case of ties)

e) Robot does not leave the lighting in the same state it found it: performance class 5 + 1 penalty (+ time in case of ties)

f) Robot hits Annie: performance class 0

g) Robot drops the glasses (breaking them): performance class 0

The ranking in the previous simulations is mostly dominated by time and by the fact that most teams acquired a penalty. Let us observe what happens when a robot is not able to accomplish all required achievements:

h) Robot stops working while carrying the (right) glasses: class 4 (assumed: command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses, robot holds the right glasses)

i) Robot brings the wrong glasses and stops execution: class 3 (assumed: command by Annie understood, robot reaches the location suggested by Annie)

j) Robot drops the right glasses: performance class 0

k) Robot finds the right glasses but is not able to grasp them: class 3 (assumed: command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses)

l) Robot stops working after finding the glasses: class 3 (assumed: command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses)

m) Robot cannot detect the glasses: performance class 2 (assumed: command by Annie understood, robot reaches the location suggested by Annie)

n) Robot cannot find the location where the glasses are located: performance class 1 (assumed: command by Annie understood)

o) Robot stops working during the search: performance class 1 (assumed: command by Annie understood)

We may have also cases where the task was completed, but in an unsatisfactory way:

p) Robot brings several glasses. Among them are the right glasses as well: performance class 5 (assumed: command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses, robot holds the right glasses, robot handles the right glasses to Annie) + 1 penalty because of dealing with wrong objects

q) Robot brings the right glasses on the second trial: performance class 5 (command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses, robot holds the right glasses, robot handles the right glasses to Annie) + 1 penalty because of dealing with wrong item + more time is needed

r) Robot needs 2× time than the fastest robot to do the same task in the competition: performance class 5 (command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses, robot holds the right glasses, robot handles the right glasses to Annie) + more time is needed

s) Robot reports the right position of the glasses but cannot bring them to Annie: class 3, (command by Annie understood, robot reaches the location suggested by Annie, robot reports the position of the right glasses) but less time is needed to perform the task

3.3 Combining task rankings

The RoCKIn Competitions are organized in a number of Task Benchmarks[5] that are evaluated and scored as explained in the previous sections. So far we have discussed the framework for the ranking of one task, now let assume we have three possibly different rankings of teams out of three Task Benchmarks. In this case we are interested in deciding which is the winning team of the competition. Provided a team has to participate, and get class 1 at least, we will decide the winning team by a social welfare function, i.e., rank combination.

In general, social welfare functions can be analyzed according to the properties they satisfy (see `http://en.wikipedia.org/wiki/Voting_system`). More specifically, the *Condorcet criterion* of social welfare functions states that a Condorcet winner must be selected when one exists. The Condorcet winner is the team that, when compared with every other team, is preferred in all the three rankings. The *Independence of Irrelevant Alternatives* (IIA) says that if team A is better than team B out of the choice set {A,B} by a social welfare function combining the three rankings that include A, B, and a third team X, then if only the position of X changes in the rankings, the social welfare function must not lead to B's being preferred over A. In other words, whether team A or team B is better should not be affected by the position of team X, which is irrelevant to the choice between A and B.

A possible rank combination is based on the social welfare function called *Borda count*. According to Borda count, a team A receives k points if in a Task Benchmark ranking, team A has performed better than k other teams. All the points for all teams are calculated and a global ranking is determined. For example, consider the rankings for the three Task Benchmarks of Table 3.1. In this case, team A gets $3+3+0=6$ points, team B gets $2+2+3=7$ points, team C gets $1+0+2=3$ points, and team D gets $0+1+1=2$ points. Hence, the global ranking is B (best), A, C, D (worst).

Table 3.1: An example of rankings with three tasks and four teams (A to D)

Rank	Task 1	Task 2	Task 3
1	A	A	B
2	B	B	C
3	C	D	D
4	D	C	A

The Borda count does not satisfy, in general, the Condorcet criterion nor the IIA. An alternative social welfare function for combining task rankings is based on the *Copeland's rule*. A team gets a point for every pairwise win (according to the majority of the rankings), one half point for pairwise ties, and zero points for pairwise loses. After all the possible pairwise matches between teams, the points of each team determine the global ranking. In the example of Table 3.1, team A gets 3 points (it wins against team B, because it is placed before B in two out of three rankings, and against teams C and D), team B gets 2 points (it wins against teams C and D, but it looses against team A), team C gets 1 point (it wins against team D, but it looses against teams A and B), team D gets 0 points (it looses against all other teams). Hence, the global ranking is A (best), B,

[5]Specifically: 3 @Home Task Benchmarks and 3 @Work Task Benchmark for the RoCKIn 2014 edition.

C, D (worst). The Copeland's rule satisfies the Condorcet criterion but, in general, not the the IIA.

There are dozens of other social welfare functions, some of them satisfying Condorcet criterion or IIA. However, most of these alternative social welfare functions are not intuitively easy to calculate and, as such, do not appear to be suitable for employment in the RoCKIn Competitions. In the 2014 RoCKIn Competition, the Borda count will be used because of its easy formulation. Other alternative social welfare functions for combining task rankings, such as the Copeland's rule, will be evaluated to be applied in the RoCKIn 2015 Competition, only after the 2014 RoCKIn Competitions in case the need to compensate for possible distortions will arise.

Chapter 4

Functionality Benchmarks

This chapter, as introduced in Section 2.4, is concerned with the scoring methodologies and metrics for Functionality Benchmarks.

4.1 Scoring framework for functionality benchmarks

As anticipated by Section 2.4, it is not possible to define a single scoring framework for all Functionality Benchmarks as it has been done for Task Benchmarks in the previous chapter. These, in fact, are specialized benchmarks, tightly focused on a single functionality, assessing how it operates and not (or not only) the final result of its operation. As a consequence, scoring mechanisms for Functionality Benchmarks cannot ignore how the functionality operates, and metrics are strictly connected to the features of the functionality. For this reason, differently from what has been done for Task Benchmarks in Chapter 3, this chapter will define scoring methodologies and metrics separately for each Functionality Benchmark of the 2014 RoCKIn Competition.

In RoCKIn, Functionality Benchmarks are defined by four elements:

- Description: a high level, general, description of the functionality.
- Input/Output: the information available to the module implementing the functionality when executed, and the expected outcome.
- Benchmarking data: the data needed to perform the evaluation of the performance of the functional module.
- Metrics: algorithms to process benchmarking data in an objective way.

The following of this section describes these four elements for each of the functionalities that will be benchmarked during the 2014 RoCKIn Competition ether through Functionality Benchmarks or, indirectly, through Task Benchmarks. Additional operative details may be found in the Rulebooks for the Competition; in particular, whenever multiple metrics to evaluate a Functional Benchmarks exist, we provide a procedure to come at a unique unambiguous ranking among the participants through their aggregation or prioritization.

4.1.1 Functionality benchmark: task planning

Description: Starting from a high level description of a problem domain (initial state and applicable actions) and a desired goal, find a set of actions which will lead the robot to the desired goal.

Input/Output: Input is a description of the current state and the goal expressed as statements in PDDL[1]. Also possible actions should be specified in PDDL with their preconditions and effects. Output is a plan, expressed as a sequence of actions, a partial order of plan steps, a task network, or a policy function mapping states into actions.

Benchmarking data: Some data has to be supplied by the robot, including: the input given to the planner, the output (i.e., the plan) delivered by the planner, and the performance data for the planner (e.g., memory and time needed). According to the planning approach, other data might be required, e.g., the formulation of the domain used by the planner; if SAT-based planners are compared, this requires the number of clauses in 3CNF generated by the transformation of the planning problem into the propositional representation; for graph-based planners the depth of the generated planning graph assumes a similar role; for policy based planners the generated policy function. Other data will be possibly collected when a plan is actually executed, including: time needed to execute each action and time needed to execute the whole plan.

Metrics: There is a long tradition of measuring plan performance in the International Planning Competition[2], where metrics are mainly related to good plan quality and to solving time. RoCKIn builds on this experience and defines the metrics listed below. The target of benchmarking task planning is twofold: to asses the quality of the *plan* produced (e.g., in terms of the cost for executing it and its likelihood to succeed) and to assess the quality of the *planning process* (i.e., how fast is the plan generated and how much resources are needed to do so).

- Quality of resulting plans is measured according to the number of correctly planned actions, considering also their order. Specifically, given the optimal plan, represented as an ordered list of actions, and a plan to be evaluated, also represented as an ordered list of actions, the Levenshtein distance between the two lists is calculated to measure the quality of the plan. *Levenshtein distance* $L_{A,B}(|A|, |B|)$ between two lists A and B is calculated as:

$$
\begin{aligned}
L_{A,B}(i,j) = &\quad \max(i,j) && \text{if } \min(i,j) = 0 \\
L_{A,B}(i,j) = &\quad \min(L_{A,B}(i-1,j)+1, \\
&\qquad L_{A,B}(i,j-1)+1, \\
&\qquad L_{A,B}(i-1,j-1)+1_{a_i \neq b_j}) && \text{otherwise}
\end{aligned}
$$

  where $|A|$ is the length of list A (same for B) and $1_{a_i \neq b_j}$ is 0 is $a_i = b_j$ and 1 otherwise. In case of plan defined in terms of policy function, the policy is first

[1] http://en.wikipedia.org/wiki/Planning_Domain_Definition_Language
[2] http://ipc.icaps-conference.org/

executed from the start state to the goal state in order to retrieve the list of actions that has to be evaluated. This is a integer-valued metric; the smaller the better.

- Number of actions correctly performed in the correct order (e.g., number of objects delivered to their correct destination) within a given time frame (e.g., 10 minutes) when executing the plan. In case of plan defined in terms of policy function, the policy is first executed from the start state to the goal state in order to retrieve the list of actions that has to be evaluated. This is a integer-valued metric; the larger the better.
- Time required to construct a plan. This is a real-valued metric; the smaller the better.
- Time required to execute the single actions of a plan and the whole plan. This is a real-valued metric; the smaller the better.

4.1.2 Functionality benchmark: 2D geometric mapping

Description: Starting from a given pose in the testbed, build a map of it. In this context, a map is defined as "any digital representation of the environment suitable for performing other functionalities (e.g., self-localization, path planning, etc.)". Depending on the specific robot platform under test, mapping requires a more or less extended exploration of the testbed.

Input/Output: Sensor data provided by the devices of the robot system under test. Optionally (depending on the specific benchmark) additional data (e.g., video) may be provided by devices that are part of the testbed together with the set of poses traversed by the robot. The expected output is the representation of the environment which is most suitable for the intended use.

Benchmarking data: Given a set of physical points on the testbed (e.g., extreme points of intersection lines between surfaces like the top and bottom corners of the edge connecting two walls or the end points of the edge of a table) identified by RoCKIn, benchmarking data are the Euclidean distances between each pair of such points, as evaluated on the map, on the horizontal (x, y) plane[3] and the true distances in the environment (i.e., the ground truth). The team responsible for the robot can choose one of the following ways to compute such distances:

- provide RoCKIn personnel with the map built by the robot and with readily usable software tools (e.g., Matlab scripts) to process it in order to perform point selection and distance computation;
- perform point selection and distance computation manually under supervision of the RoCKIn personnel; in this case, the team is also required to provide a

[3]The use of *derived quantities* (such as distances) instead of direct analysis of the map makes the benchmark largely independent from the representation of the map. This is a key issue, as it ensures that robot systems are not forced to use any specific representation for their maps. The choice of distances as derived quantities is due to (i) their clear physical significance and (ii) their strong connection to the possibility of using the map to plan accurate and feasible paths through cluttered environments.

description of the algorithms used for these operations, and to accept scrutiny and/or recomputation by competing teams[4].

Metrics: The following metrics explore two key aspects of the mapping process:

- Time required to provide a map complete with a representation of all the physical points composing the set used for benchmarking. This is a real-valued metric; the smaller the better.
- Average and maximum error of the reconstructed distances against the ground truth distances[5]. These are real valued metrics; the smaller the better.
- Percentage of mapped area (as measured by the percentage of mapped RoCKIn reference points) in a given time interval; the higher the better.

4.1.3 Functionality benchmark: 2D self-localization

Description: Being able to estimate the robot's own pose (i.e., x and y position + orientation θ) with respect to a known fixed reference frame in a map[6] while moving through it.

Input/Output: Having a map (either previously acquired by the robot or provided by the organizers) and some sensor data, the robot provides a (sequence of) 3DoF pose estimate(s) with respect to a known fixed reference frame in the map (an initial estimation of the relative pose of this reference frame and of the robot reference frame could be needed). The robot could move to perform self-localization especially in the case of "kidnapping". The pose estimate is (optionally) accompanied by an estimate of its uncertainty (e.g., through the use of the first K moments of its distribution).

Benchmarking data: the sequence of poses estimated by the robot during a path (the path could be provided[7], or it could be "random"), possibly with their uncertainty estimation. A ground truth measurement of the sequence of poses of the robot during its movement; the two sequences should be synchronized and should be referred to a common reference system[8].

Metrics: Several metrics are possible, each of which measuring a different aspect of this functionality.

[4]Complexity of application is the main drawback of the representation-independent nature of this benchmark. Practical difficulties of execution could easily prove to be excessive for the quick-paced timing of a competition event.

[5]These have to be evaluated with high precision. One way to do this is by measuring point locations with the same ground truth localization system used to measure robot position, then computing distances using such measurements.

[6]In self-localization, we stress the idea of being able to use sensors looking at the environment features, possibly represented in a map, and we do not assume the robot to have absolute localization sensors, e.g., GPS. The case where the robot has a global sensor could be named as *localization*.

[7]The need for a given path is related to the repeatability of the benchmark; an example of this could be the UMBmark for odometry evaluation.

[8]Or it should be possible to refer the two sequences to a common time base and reference frame by post processing.

- Time required to self-localize from an unknown pose ("kidnapped robot") with a position error below a given threshold. If the self-localization algorithm is not able to obtain an adequate accuracy in positioning, we consider the time required to reach a position for which the error does not change for a given amount of time. This is a real valued metric; the smaller the better.
- Relative Pose Error (RPE) [3] measures the local accuracy of the trajectory over a fixed time interval. Therefore, the relative pose error corresponds to the drift of the trajectory which is in particular useful for the evaluation of visual odometry systems. This is a real-valued metrics; the smaller the better.
- Average and maximum relative pose error on a given path. These are real valued metrics; the smaller the better.
- Average and maximum relative pose error on a path of a given length. These are normalized real valued metrics between 0 and 1; the smaller the better.

4.1.4 Functionality benchmark: robot path planning

Description: Determine the sequence of poses the robot should take for moving from a starting position to a goal position. This sequence should be executable by the robot and should keep the robot away from obstacles (i.e., not incurring in a collision).

Input/Output: A starting pose and a goal pose are given to the robot together with a map of the environment with known static obstacles. Path planning can produce either (i) a continuous geometric path described in terms of geometric primitives and (optionally) speed profile along it or (ii) a sequence of poses from the starting one to the goal one which the robot should reach during time, and (optionally) a speed profile along the path or (iii) a sequence of actions (commands to the base) to be executed from the starting pose to the goal pose (commands could be in terms of wheels speed or angular/tangential robot speed)[9]. As part of the benchmark (optional) requirements could be set on the kinematics of the robot, dynamic constraints on the path to be generated. A map with static objects to be avoided while moving and the footprint of the robot are available for path planning.

Benchmarking data: A sequence of poses to be reached during time and (optionally) the speed of the robot when reaching them (absolute speed in world reference frame) or a sequence of commands to be executed by the robot from the starting pose to the goal pose. The time required to perform this planning should be provided too. A method to interpolate consecutive poses could be necessary, for example to calculate the length of a path (see below).

Metrics: Several metrics are possible, each of which measures a different aspect of this functionality. For each of them, we have a *theoretical* value when we assume the real robot to obey its kinematic/dynamic model, or *empirical* value if we measure it during an empirical run (or several runs). In case it is measured empirically we should keep in mind it is "affected" by the quality of the execution of the run and we need to acquire ground truth data as well. In the following we refer to the

[9] In this case, in principle, any evaluation of the plan should consider also kinematics and dynamics properties of the robot, so that commands could really be executed, otherwise measures such as length and time required to execute a plan can be easily optimized with unrealistic assumptions.

theoretical case so to keep this Functionality Benchmarking as much independent from path following as possible; all following metrics can be applied to the empirical evaluation as well:

- Feasibility of the path, i.e., it starts from the starting pose + it ends in the goal pose + it does not hit obstacles + it respects (optional) kinematic and dynamic constraints. This is a Boolean metric; `true` is better.
- Time required to obtain the path from its request to the result, provided that the path is feasible. This is a real valued metric; the smaller the better.
- Time required by the robot to execute the planned path from the starting pose to the goal pose, provided the path is feasible. This is a real valued metric; the smaller the better.
- Length of the planned path from the starting pose to the goal pose, provided that the path is feasible. This is a real valued metric; the smaller the better.
- Linear quadratic (LQ) cost function (derived from linear quadratic optimal control theory)

$$J = \frac{1}{2} \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)] \cdot dt \tag{4.1}$$

  where the cost function can be thought of physically representing the control energy (measured as a quadratic form)[10]. This is applicable only when the sequence of commands is provided as output by path planning, and it is assumed to be valid if the path is feasible. This is a real valued metric; the smaller the better.
- Average/minimal/maximal distance from obstacles; depending on the final aim of the plan we could have both the higher and the lower the better (provided we do not collide).

4.1.5 Functionality benchmark: robot path following

Description: A given point of the robot (e.g., the center of its footprint) should follow a path defined on a map or according to some known reference system, eventually with a pre-defined timing.

Input/Output: the input is a path described either by a generic curve in space (eventually annotated with time and speed), or by a sequence of poses (eventually annotated with time and speed), or by a sequence of actions for each of which the expected effect is known and measurable. On the path there might be obstacles, i.e., objects that may prevent the robot to follow the path (in this case the robot is expected to stop and signal the problem). The position, and footprint at different heights, of each obstacle can be either communicated to the robot or perceived through sensors. Obstacles can be either fixed or mobile. The movement law of the obstacles can be either communicated to the robot or learned from data perceived through sensors.

[10] All terms in this formula need to be specified according to the specific instance of this Functionality Benchmark.

Benchmarking data: A sequence of real positions in space and time of the defined point of the robot, or of points of the robot that can be tracked, if their position relative to the defined point is provided. Resolution in space or time of this sequence is predefined according to the benchmarking goals and the quality of the tracking device. Optionally the sequence of commands executed by the robot can be provided to compute the control energy.

Metrics: Several metrics are possible, each of which measuring a different aspect of this functionality.

- Distance from the path, computed as sum of the distance from real poses in space and time to the planned poses normalized by the number of poses. Also average and max of distances might be considered. To be minimized. Note that this also provides an evaluation of obstacle avoidance if the same obstacles are presented in a repeatable way. Real-valued metric.
- Linear quadratic (LQ) cost function (derived from linear quadratic optimal control theory) in the empirical case (see Equation 4.1). This is applicable only when the sequence of commands is provided as output as well. This is a real valued metric; the smaller the better.
- Percentage of hit obstacles wrt the total number of the ones faced. This might also be split according to the different nature of obstacles: fixed, mobile, known a priori, unknown, etc. This is a real-valued metric, the smaller the better.
- Having specified a minimal distance from obstacles, compute the number of times the robot is closer than a threshold to obstacles during the path or the expected value of such distance. This metric is related to robot safety, the smaller the better in case of number of times closer, the higher the better in the case of expected value of the distance.

4.1.6 Functionality benchmark: object/face detection, recognition, and localization

Description: [11] By using proprioceptive sensors the robot perceives the presence of specific items in the environment. Objects belong to a given class or category (e.g., kitchenware, bolts, faces, glasses, boxes, pets, cans, etc.); we refer to *object detection* when the functionality provides only information about the class of the perceived objects. For each class several instances of objects exists; *object recognition* functionality provides information about the specific instance of an object within a given class (e.g., Paul's face, Mountain Dew can, Hex Cap 10mm Anodized bolt, etc.).

Input/Output: Sensor data by the devices of the robot system under test[12]. In the case of *object detection*, the set of categories the object belong to is know to the robot together with a description[13] for each of them and the expected output is the

[11] Since the functionality of Face Detection is a specific case of the more general Object Detection we decide to merge those together is a single description.

[12] Although the most used sensors for object detection and recognition are cameras, either 2D or 3D, here we left the perception system intentionally unspecified since the functionality (not its performance) is independent from that.

[13] An object description can be at different levels; as general as an ontology description or as specific as a detailed metrical or appearance based (set of) model(s).

notification of the presence or absence of an object of a given category in the scene. In the case of *object recognition*, a label for each instance of each class is also know to the robot together with its description and the expected output is the label of the recognized object. In case of localization 3D models of the objects to be localized, together with a reference frame for the objects, can be optionally provided to the robot, and the expected output is the object pose, i.e., position and orientation, of the object reference frame with respect to a world, or robot, reference frame.

Benchmarking data: Given a set of objects from a list of possible instances the robot is shown an object and it has to provide a notification with the label of the class, in case of object detection, the label of the specific instance of the shown object, in case of object recognition, and the 3D pose of the object, i.e., position and orientation, in case of localization. This benchmark can be run also "in the wild" that is the robot is guided in an environment, or is provided a pre-recorded stream of sensory input, and it has to provide notifications and poses of all recognized objects (or from a specific category).

Metrics: Several metrics are possible, most of them being inspired by object retrieval benchmarking best practices:

- Accuracy of recognition measured as percentage of correctly classified/recognized objects among the ones presented to the robot. This applies also to the "in the wild" modality; this is a real-valued metrics, the higher the better.
- Precision and Recall might be better metrics when multiple objects, possibly in multiple instances, are simultaneously presented to the robot, or are present "in the wild"; the former counts the number of correctly detected/identified items among those that have been detected/identified (it is similar to the accuracy, but applies only to the object which have been notified), the latter counts the number of items detected/identified among those which are present in the observed scene. In both cases the higher the better although a high recall might be quite misleading and should be associated to a high precision as well. To deal with the precision-recall trade-off the two metrics are often combined in the so called F-measure which is the harmonic mean of the two; also the F-measure is a real-valued metrics, so the higher the better.
- Pose error for all correctly identified objects can be used to evaluate the performance of object localization. Different importance can be done in evaluation translational part and rotational part of the object pose; if needed, a distance comprising the two aspects could be devised by using the SE(3) algebra. It is a real-valued metric; the smaller the better.
- Time required to execute the recognition/identification/localization, including the time to process the data input. It is a real-valued metric; the smaller the better.

4.1.7 Functionality benchmark: arm path planning

Description: Determine the sequence of poses the robot end effector (or arm) should take for moving from a starting position to a goal position. This plan could be done also in the joint space thus the sequence of poses is implicitly determined

by a sequence of joint positions/velocities/accelerations. This sequence should be executable by the robot and should keep the robot away from obstacles (i.e., not incurring in a collision with an obstacle or with the robot body itself).

Input/Output: A starting configuration and a goal configuration are given for the robot end effector together with a representation of the environment containing known static obstacles. Arm path planning can produce either (i) a continuous set of trajectories for the robot arm joints described in terms of joint positions and (optionally) speed / acceleration profiles or (ii) a sequence of poses for the end effector from the starting pose to the goal pose which the robot should reach during time (point-to-point motion), and (optionally) a speed profile along the path or (iii) a sequence of motion primitives for the end effector to be executed from the starting pose to the goal pose (commands could be in terms of linear motion, circular motion, etc.). As part of the benchmark (optional) requirements could be set on the kinematics of the robot and/or dynamic constraints on the path to be generated. A map with static objects to be avoided while moving and the shape of the robot body are available for path planning.

Benchmarking data: A sequence of poses for the end effector to be reached during time and (optionally) the speed of the robot when reaching them (absolute speed in world reference frame) or a set of joint trajectories (described by possibly different kinematic quantities) to be executed by the robot controller from the starting position to the goal position or a set of motion primitives to be executed in sequence by the robot arm controller. The time required to the algorithm to perform this planning should be provided too. A method to interpolate consecutive poses could be necessary, for example to calculate the exact length of a path (see below).

Metrics: Several metrics are possible, each of which measures a different aspect of this functionality. For each of them, we have a *theoretical* value when we assume the real robot to obey its kinematic/dynamic model, or *empirical* value if we measure it during an empirical run (or several runs). In case it is measured empirically we should keep in mind it is "affected" by the quality of the execution of the run and we need to acquire ground truth data as well. In the following we refer to the theoretical case so to keep this Functionality Benchmarking as much independent from path following as possible; all following metrics can be applied to the empirical evaluation as well:

- Feasibility of the path, i.e., it starts from the starting pose + it ends in the goal pose + it does not hit obstacles + it respects (optional) kinematic and dynamic constraints. This is a Boolean metric; `true` is better.
- Time required to obtain the path from its request to the result, provided that the path is feasible. This is a real valued metric; the smaller the better.
- Time required by the robot to execute the planned path from the starting pose to the goal pose, provided the path is feasible. This is a real valued metric; the smaller the better.
- Length of the planned path from the starting pose to the goal pose, provided that the path is feasible. This is a real valued metric; the smaller the better.
- Linear quadratic (LQ) cost function (derived from linear quadratic optimal control theory) in the empirical case (see Equation 4.1). This is applicable

only when the sequence of commands is provided as output as well. This is a real valued metric; the smaller the better.

- Average/minimal/maximal distance from obstacles; depending on the final aim of the plan we could have both the higher and the lower the better (provided we do not collide).

4.1.8 Functionality benchmark: arm path following

Description: The robot end effector should follow a path defined according to some known reference system either in a cartesian space or in the joint space, with a pre-defined timing, and optionally with prescribed speeds and accelerations.

Input/Output: the input is a path for the end effector described either by a generic curve in space (eventually annotated with time and speed), or by a sequence of poses (eventually annotated with time and speed), or by a set of joint trajectories (eventually annotatoed with speed and acceleration), or by a sequence of predefined known motion primitives. On the path there might be obstacles, i.e., objects that may prevent the robot to follow the path (in this case the robot is expected to stop and signal the problem). The position, and footprint at different heights, of each obstacle can be either communicated to the robot or perceived through sensors. Obstacles can be either fixed or mobile. The movement law of the obstacles can be either communicated to the robot or learned from data perceived through sensors.

Benchmarking data: A sequence of real positions in space assumed by the end effector while moving, or of points of the robot that can be tracked, if their position relative to the end effector is provided, with corresponding timing. Resolution in space or time of this sequence is pre-defined according to the benchmarking goals and the quality of the tracking device. Optionally the sequence of commands executed by the robot can be provided to compute the control energy. As ground truth for joint space trajectory following, joint angles during time are acquired.

Metrics: Several metrics are possible, each of which measuring a different aspect of this functionality.

- Distance from the path, computed as sum of the distance from real poses in space and time to the planned poses normalized by the number of poses. Also average and max of distances might be considered. To be minimized. Note that this also provides an evaluation of obstacle avoidance if the same obstacles are presented in a repeatable way. Real-valued metric, the lower the better.
- Linear quadratic (LQ) cost function (derived from linear quadratic optimal control theory) in the empirical case (see Equation 4.1). This is applicable only when the sequence of commands is provided as output as well. This is a real valued metric; the smaller the better.
- Percentage of hit obstacles wrt the total number of the ones faced. This might also be split according to the different nature of obstacles: fixed, mobile, known a priori, unknown, etc. This is a real-valued metric, the smaller the better.
- Having specified a minimal distance from obstacles, compute the number of times the robot is closer than a threshold to obstacles during the path or the

expected value of such distance. This metric is related to robot safety, the smaller the better in case of number of times closer, the higher the better in the case of expected value of the distance.

4.1.9 Functionality benchmark: grasp planning

Description: Determine the grasping approach and grasping position according to the object to be grasped and the robot gripper. By grasping we intend the constrain of one or more degrees of freedom for the grasped object. In some cases the grasping force should be controlled when dealing with soft, fragile, or non-rigid objects[14].

Input/Output: The robot is provided with a model of the object to be grasped and (optionally) a task for which the grasping is required (e.g., pouring a bottle vs. handling a bottle vs. stacking a bottle on a pile). The robot is assumed to know the kinematic and (optionally) the dynamics of the gripper and it has to plan the grasping approach position and (optionally) the grasping force so to limit the target degrees of freedom of the object and thus execute the desired task.

Benchmarking data: The data required to evaluate grasp planning are the grasping approach and the graps position with respect to the object as produced by the planning algorithm. Also the time required to produce the grasp plan is required to evaluate the effectiveness of the algorithm under evaluation.

Metrics: Different metrics could be defined for grasp planning. One of the most delicate aspects, in evaluation grasp planning, is to evaluate a grasping solution in a completely independent way with respect to its actual execution. In the following we provide some metrics which aim at evaluation the outcome of this functionality independently from the actual execution of its result.

- Planning success, i.e., the fact that the planning algorithm has achieved its goal in terms of limiting the prescribed degrees of freedom taking into account the task for which these degrees of freedom are required to be constrained. This is a boolean score, true is better.
- Time required to generate the grasping plan; this is a real-valued metrics the lower the better.

4.1.10 Functionality benchmark: visual servoing

Description: Visual servoing functionality, also known as Vision-Based Robot Control, uses feedback information extracted from a vision sensor to control the motion of a robot. Although it might be implemented as a combination of other basic functionalities, e.g., object localization + arm path planning + arm path execution, it might be occur also as a monolithic functionality, i.e., Image Based Visual Servoing, for which novel metrics need to be devised[15]

[14]Here we focus in the planning of the grasp and we do not specify a functional benchmark for the grasping execution which could be eventually re-casted in terms of control problem with (optionally) force feedback. In this case classical metrics for control system evaluation could be applied, e.g., repeatability, accuracy, bandwidth of the control loop, disturbance rejection, etc.

[15]This is the rationale of having this functionality benchmark and not just referring to the combination of existing ones.

Input/Output: The robot, or its end effector, is required to reach a given pose; as input, beside normal proprioceptive perception, it has a sequence of frames captured by a camera which is fixed to the moving body and observes the scene. A set of features from the first image need to reach a specified position in the image which refer to the target pose.

Benchmarking data: The sequence of poses reached by the robot or by its end effector from the initial configuration to the final configuration; the sequence of images acquired and processed by the robot during its motion; the sequences of positions in the acquired frames of the features for which an initial and a goal configuration have been defined.

Metrics: Depending on the aspect to be evaluated some metrics can be defined:

- Absolute position accuracy and repeatibility in reaching the target pose/configuration with respect to a predefined 3D reference frame; this is a real-valued metrics, the higher the better.
- Absolute position accuracy and repeatibility in reaching the target pose/configuration in terms of positions of image feature, thus with respect to the origin of the image frame; this is a real-valued metrics, the higher the better.
- time required to perform the control loop including the time required to perform image processing; this is a real valued metrics, the lower the better.

4.1.11 Functionality benchmark: input from humans through speech

Description: The robot should recognize a command issued by a person (or played from a file) through a sound reproduction system, by interpreting the audio signal coming from a loudspeaker to its own audio sensors. A version of the benchmark testing only the interpretation process and not the input audio system can use audio files instead of actual input from on-board microphone(s)

Input/Output: A priori, robots are provided with: the lexicon used in the benchmark (full list of verbs and nouns of objects used in the spoken sentences to be recognized), sample audio files, a knowledge base (Frame Knowledge Base, FKB) containing a set of *semantic frames* as described in the rulebook. In the reduced version of the benchmark, audio files with commands to be interpreted are provided as input. In the complete version, commands are provided in natural language as voice sound produced by a sound reproduction system. The output is the CFR (Command Frame Representation - as described in the rulebook) for each command that has been provided.

Benchmarking data: data acuired for benchmarking are:

- sensor data (in the form of audio files) used by the robot to perform speech recognition[16];

[16]Speech files from all teams and all benchmarks (both Task benchmarks and Functional benchmarks) will be collected and used to build a public dataset. The audio files in the dataset will therefore include all the defects of real-world audio capture using robot hardware (e.g., electrical and mechanical noise, limited bandwidth, harmonic distortion). Such files will be usable to test speech recognition software, or (possibly) to act as input during the execution of speech recognition benchmarks.

- the set of all possible transcriptions, in the appropriate natural language, for each user utterance;
- the final command produced during the natural language analysis process;
- intermediate information produced or used by the natural language understanding system during the analysis as, for example, syntactic information.

Metrics: Different aspects of the speech understanding process can be assessed:

1. The Word Error Rate on the transcription of the user utterances, in order to evaluate the performance of the speech recognition process.
2. For the generated CFR, the performance of the system will be evaluated against the provided *gold standard* version of the CFR, that is conveniently paired with the analyzed audio file and transcription. Two different performances will be evaluated at this step. One measuring the ability of the system in recognizing the main action, called *Action Classification* (*AcC*), and one related to the classification of the action arguments, called *Argument Classification* (*AgC*). In both cases the evaluations will be carried out in term of Precision, Recall and F-Measure. This process is inspired to the *Semantic Role Labeling* evaluation scheme proposed in [5]. For the *AcC* this measures will be defined as follow:
   - Precision: the percentage of correctly tagged frames among all the frames tagged by the system;
   - Recall: the percentage of correctly tagged frames with respect to all the *gold standard* frames;
   - F-Measure: the harmonic mean between Precision and Recall.

   Similarly, for the *AgC*, Precision, Recall and F-Measure will be evaluated, given an action f, as:
   - Precision: the percentage of correctly tagged arguments of f with respect to all the arguments tagged by the system for f.
   - Recall: the percentage of correctly tagged arguments of f with respect to all the *gold standard* arguments for f.
   - F-Measure: the harmonic mean between Precision and Recall.
3. Time utilized (if less than the maximum allowed for the benchmark).

4.2 Combining task and functionality rankings

The availability of both task and functionality rankings opens the way for the quantitative analysis of the importance of single functionalities in performing complex tasks. This is an innovative aspect triggered by the RoCKIn approach to competitions. While a full evaluation of the practical viability of the approach is deferred after the first 2014 RoCKIn Competitions, this section provides some details about the quantitative analysis of the importance of functionalities in performing tasks which is under consideration.

To state the importance of a functionality in performing a given task, RoCKIn will borrow the concept of *Shapley value* from Game theory. Let assume a coalition of players (Functionalities in the RoCKIn context) cooperates, and obtains a certain overall gain from that cooperation (the Task Benchmark scoring in the RoCKIn context). Since

Table 4.1: An example of task and functionality scores for a team

Benchmark	Score
F1	10
F2	5
F3	1
T1 ={F1,F2}	10
{F1,F3}	11
T2 ={F2,F3}	15
T3 ={F1,F2,F3}	20

some players may contribute more to the coalition than others or may possess different bargaining power (for example threatening to destroy the whole surplus), what final distribution of generated surplus among the players should arise in any particular game? Or phrased differently: how important is each player to the overall cooperation, and what payoff can he or she reasonably expect? Or in the RoCKIn jargon: how important is each Functionality to the reach a given performance in a Task Benchmark?

The Shapley value, named in honour of Lloyd Shapley who introduced it in 1953 [8], provides one possible answer to the aforementioned question by assigning, to each cooperative game, a unique distribution (among the players) of a total surplus generated by the coalition of all players. Re-phrased in RoCKIn jargon, the Shapley value could be used to assign, to each Task Benchmark, a unique distribution (among involved Functionalities) of the task score.

To clarify how Shapley values will be considered in RoCKIn to distribute task scoring among the involved functionalities, we consider the following example. Let three rankings relative to three tasks that involve three functionalities, which have been independently evaluated[17]. Task T1 involves functionalities F1 and F2, task T2 involves functionalities F2 and F3, while task T3 involve all three functionalities F1, F2, F3. Considering a single team, assume that its scores in the three Task Benchmarks are: score(T1)=10, score(T2)=15, score(T3)=20. The scores of the same team according to functionalities are score(F1)=10, score(F2)=5, score(F3)=1. The situation is depicted in Table 4.1. Note that a score for the combination of functions {F1,F3} is missing (no task involving only these two functionalities has been directly evaluated) and so the sum of the individual scores of F1 and F3 is used for scoring the combination {F1,F3}[18]. Assuming that all scores are expressed according to the same scale, the Shapley values of the single functionalities can be calculated as:

$$\phi_i = \frac{1}{n!} \sum_{\pi} [v(C_\pi(i) \cup \{i\}) - v(C_\pi(i))]$$

where i is a functionality, n is the total number of functionalities, π is a permutation of the n Functionality Benchmark scores, $C_\pi(i)$ is the set of functionalities that precede i is the permutation π, and $v()$ is the score of the set of functionalities specified as argument.

[17] In the following for the sake of easiness we will consider all the score to be integer values; the approach, when used to evaluate RoCKIn teams, will take into account the different metrics and scoring we have designed for the competition.

[18] Alternatives are possible, like calculating approximate Shapley values.

In the above example,

$$\phi_{F1} = \frac{1}{3!}\sum_{\pi\in\Pi}[v(C_\pi(F1)\cup\{F1\}) - v(C_\pi(i))]$$

having defined

$$\begin{aligned}\Pi &= \{\{F1,F2,F3\},\{F1,F3,F2\},\{F2,F1,F3\}, &(4.2)\\ &\quad \{F2,F3,F1\},\{F3,F1,F2\},\{F3,F2,F1\}\}. &(4.3)\end{aligned}$$

This turns into

$$\begin{aligned}\phi_{F1} = \frac{1}{3!}[&v(\{F1\}) - v(\{\}) + v(\{F1\}) - v(\{\}) + v(\{F1,F2\}) - v(\{F2\})\\ &+v(\{F1,F2,F3\}) - v(\{F2,F3\}) + v(\{F1,F3\}) - v(\{F3\})\\ &+v(\{F1,F2,F3\}) - v(\{F2,F3\})]\end{aligned}$$

$$\begin{aligned}\phi_{F1} = \frac{1}{3!}&[10-0+10-0+10-5+20-15+11-1+20-15]\\ &= \frac{1}{3!}45 = \frac{15}{2}.\end{aligned}$$

Similarly, $\phi_{F2} = \frac{42}{6} = 7$ and $\phi_{F3} = \frac{11}{2}$, showing that functionality $F1$ has a larger impact on the performance of the considered tasks.

We remark here that the use of Shapley values in RoCKIn is proposed as a post-competition analysis tool, for the time being it has not been validated on the field, and it will be used experimentally in RoCKIn 2014 for the first time. Other techniques, like the Banzhaf power index [2] or the Shapley-Shubik power index [7], could be used to perform the same kind of analysis and quantitatively evaluate the role of functionalities in tasks. After the validation of the Shapley value, in case we will devise the need for that, we will consider such alternatives too.

Bibliography

[1] M. Anderson, O. Jenkins, and S. Osentoski. Recasting robotics challenges as experiments. *IEEE Robotics Automation Magazine*, 18(2):10–11, 2011.

[2] John F. Banzhaf. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19(2):317–343, 1965.

[3] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kummerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J.D. Tardos. A comparison of slam algorithms based on a graph of relations. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2089–2095, Oct 2009.

[4] R. Cohn, A.G.and Dechter and G. Lakemeyer. The competition section: a new paper category. *Artificial Intelligence*, 175:iii, 2011.

[5] Danilo Croce, Giuseppe Castellucci, and Emanuele Bastianelli. Structured learning for semantic role labeling. *Intelligenza Artificiale*, 6(2):163–176, 2012.

[6] D. Holz, L. Iocchi, and T. van der Zant. Benchmarking intelligent service robots through scientific competitions: The RoboCup@Home approach. In *Proc. AAAI Spring Symposium on Designing Intelligent Robots: Reintegrating AI II*, pages 27–32, 2013.

[7] M. Shubik L.S. Shapley. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[8] Lloyd S. Shapley. A value for n-person games. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, volume 28 of *Annals of Mathematical Studies*, pages 307–317. Princeton University Press, 1953.

[9] B. Smart. Competitions, challenges, or journal papers. *IEEE Robotics Automation Magazine*, 19(1):14, 2012.

[10] M. Tedre and N. Moisseinen. Experiments in computing: A survey. *The Scientific World Journal*, Volume 2014:1–11, 2014.