Funded by the European Union

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

Description of Ground-truth system - v2

Grant Agreement Number: 601012
Funding Period: 01.01.2013 - 31.12.2015
Instrument: Coordination and Support Action

Deliverable:	D-2.1.8
Due Date:	June 30, 2015
Latest Update:	July 10, 2015
Revision:	1.0

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

IST-ID
Associação do Instituto Superior Técnico para a Investigação e Desenvolvimento

SAPIENZA
UNIVERSITÀ DI ROMA

Hochschule Bonn-Rhein-Sieg

KUKA

POLITECNICO DI MILANO

INNOCENTIVE®

Contents

List of Figures

Chapter 1

Introduction

1.1 About this document

The RoCKIn project aims at performing "benchmarking through competitions", which requires an approach as much rigorous as possible, without damaging the competitions' key elements of thrill and fun. A rigorous approach requires, in particular, that the observation of the performance of the participating robot systems is executed as precisely and objectively as possible. Whenever this is feasible, the observation should be based on reliable measurements of physical reality, with which the internal representation of the same reality provided by the robot systems can be compared. Such reference measurements, when available, are called **ground truth**.

As with Deliverable D2.1.7, the focus of Deliverable D2.1.8 is on data collection, with special attention devoted to ground truth data. Collected data are identified, collectively, by the term **data for benchmarking**. Up to the finalization of D2.1.7, the actual benchmarking activity of RoCKIn only included data collection, as no RoCKIn Competitions had yet taken place. Conversely, D2.1.8 comes after the first RoCKIn Competition. For this reason, D2.1.8 describes data collection in the wider context of a **RoCKIn Benchmarking System** that includes not only the elements dedicated to data collection, but also additional components dedicated to data processing (to evaluate robot performance) and to interaction with the infrastructure of the Competition (e.g., Referee Boxes).

1.2 How to read this document

This Deliverable is subdivided into two Parts and one Appendix. The idea behind this subdivision is that each reader can choose what part(s) of D2.1.8 to read according to what information about the RoCKIn Benchmarking System she is actually interested in. Precisely readers interested in:

- knowing about the **description of the System** and its configuration should read **Part I**;
- knowing about the **field tests** which led to the development of the System should read **Part II**;
- knowing the **technicalities** of how the System operates at a Competition should read **Appendix A**.

Readers who are familiar with the first version of this Deliverable (D2.1.7) will find that common content between it and D2.1.8 is limited to parts of Chapter 2 and to

the entire Chapter 3. Chapter 3 describes the experience at the RoCKIn Camp 2014 in Rome (Italy), and the lessons learned. These lessons led to a more advanced version of the RoCKIn Benchmarking System, which was successfully used at the first RoCKIn Competition 2014 in Toulouse (France).

Experience gained in Toulouse is described by new Chapter 4. Chapter 4 also describes the main remaining open issue for the RoCKIn Benchmarking System: namely, how to ensure that teams participating to the Competition correctly implement the procedures for the collection of data for benchmarking.

The benchmarking work of RoCKIn at the RoCKIn Camp 2015 in Peccioli (Italy) was focused on the ways to deal with the above issue. The results of this work are described in new Chapter 5.

Part I

Design and execution

Chapter 2

The RoCKIn Benchmarking System

This Chapter is dedicated to outlining the problems involved in data collection for benchmarking, and to descibing the RoCKIn solution to such problems. The focus will be, of course, on the system designed and built to acquire and process data for benchmarking at the RoCKIn Competitions: i.e., the **RoCKIn Benchmarking System** as defined in Section 1.1. With respect to the previous version of this Deliverable (i.e., D2.1.7), much experience has been gained and much work has been done. The result is that the RoCKIn Benchmarking System has now a well-defined structure and -importantly- has already been validated at the RoCKIn Competition 2014 (as well as at the RoCKIn Camp 2015). The three elements which are required for the benchmarking activities of RoCKIn are following:

1. hardware elements;
2. software elements;
3. competition procedures.

The first two items compose the RoCKIn Benchmarking System.

While the third item probably has a much less self-explanatory meaning than the previous two, competition procedures are a key element of the benchmarking process. At a robot competition, even a *benchmarking competition* such as RoCKIn Competition, participating teams are interested in getting a good ranking and showing off their abilities. They are not interested in providing good data for benchmarking, unless this is instrumental to those goals. For this reason, introducing objective benchmarking into the workings of a robot competition (as RoCKIn does) adds a whole additional layer of constraints to competition procedures. Part of the work of project RoCKIn consists of carefully designing the RoCKIn Competition in such a way that benchmarking is possible without impacting on the rhythm (for spectators) and "fun" (for both spectators and teams) elements. Among the aspects of such design are the the preparatory activities to be performed on robots before each benchmark is executed, the rules governing benchmark execution, and the evaluation metrics applied to benchmark results (which of course represent the ultimate indication to teams about what to do and what to avoid), and also the practical details of the organization and execution of the Competition.

This Deliverable is not concerned with most procedures. Their complete description can, instead, be found in Deliverables D2.1.3 and D2.1.6: i.e., the Rulebooks of the RoCKIn Competition. The task of this Deliverable is to describe the RoCKIn Benchmarking System, focusing on the collection of data for benchmarking.

Of course, data collection is not the only task performed by the RoCKIn Benchmarking System in the context of the RoCKIn Competitions. The System is also required to interact with the other elements of the Competition infrastructure, and (for some benchmarks) to process the collected data to provide an assessment of robot performance. Again, these tasks are not within the scope of this Deliverable. A brief description of how the RoCKIn Benchmarking System manages them can be found in Section 2.4, while the associated technical details are available in Appendix A.

2.1 External and internal data for benchmarking

The description of each RoCKIn benchmark includes the specification of the required data for benchmarking (also called "benchmarking data" in the following). Such data describe the behaviour of the robot system under test for what concerns the aspects that the benchmark is designed to assess, and comprise two types of data.

- **External benchmarking data** are collected by the testbed or by the referees: for instance, by tracking the trajectory of a physical point of the robot, or by verifying if a given effect has been obtained by the robot. Another type of external benchmarking data refers to events that are part of a benchmark, such as what specific object has been presented to the robot during an object recognition benchmark. No active participation by the robot (nor by the team running the robot) is required to collect these data, and no requirements are set on the structure and functions of the robot in order to enable the collection of such data. The only required change to the robot may be the installation of distinctive markers or patterns on it to allow external localization, if required.

- **Internal benchmarking data** are collected by the robot system. For benchmarking, such data are streamed by the robot to the testbed, or recorded and later transferred to it. Examples of internal benchmarking data are the estimate of the robot's own pose as provided by the robot's own self-localization system, or the identification by the robot of a specific object instance among a set of candidates.

Internal benchmarking data force the robots under test to have an internal representation and/or data interfaces that comply with RoCKIn's specifications[1]. Constraints such as these put an additional burden on participating teams, because they may require modifications to the robots; for some robots, such modifications may even prove to be unfeasible or too disruptive for their normal operation. For these reasons, in RoCKIn the use of internal benchmarking data is kept to a minimum by careful choice and design of the benchmarks. As it happens, practical experience at the RoCKIn events described in part II of this document has shown that this problem is largely non-existent. We experienced, in fact, a general convergence of the participating teams towards the same middleware (ROS[2]) and its associated methodology for system modularization. This simplified both the definition and the implementation of methodologies for the collection of internal benchmarking data. Additional information about this issue will be provided in Section 2.4.

[1]This does not usually force the robot to comply to a given internal representation. The only requirement is that the robot must be able to produce and record information according to a given representation. For instance, to benchmark self-localization it may be necessary for the robot to estimate its own location, which would rule out a robot which moves randomly through the environment.

[2]Robot Operating System, http://www.ros.org

2.2 Ground truth

The term **ground truth**, or **GT**, is used for reference data about the actual behaviour of a robot system, collected using highly accurate sensors and systems independent from those of the robot. For an ideal robot and environment, the robot's own perceptions and estimates would be in perfect accordance with ground truth; in a real setting, differences between what the robot perceives/estimates and ground truth data will occur, due to imprecisions in sensors and processing. Such imprecisions affect both the robot and the GT collection system, of course; however, if the GT system's imprecisions are much lower than the robot's, the data provided by the former can be used to estimate the quality of the data produced by the latter.

According to the classification of Section 2.1, in the context of RoCKIn, GT data correspond to *external benchmarking data*. For benchmarking, GT data can be used on their own or in conjunction with *internal benchmarking data* (provided by the robot). More precisely, RoCKIn benchmarks assess robot performance by using the following methods:

1. by processing GT data only (either by direct application of suitable metrics or by comparing GT data about the module under test with GT data about a reference module);

2. by jointly processing GT data and internal benchmarking data;

Only the second of these methods requires that the robot under test is compliant with RoCKIn-provided specifications about internal representations and/or interfaces: the other do not put any constraint on the robot, and is, therefore, preferred.

2.3 Elements of the RoCKIn Benchmarking System

This Section describes the problems associated to the process of acquiring the data necessary for benchmarking, and the solutions to such problems chosen for the RoCKIn Benchmarking System. Most of the contents of this Section are focused on the approaches chosen by RoCKIn. A description of the real- world implementation of the RoCKIn Benchmarking System can be found in Section 2.4, while Appendix A provides the technical details of such implementation.

2.3.1 Motion capture system

Among the ground truth data that RoCKIn acquires for the benchmarks are *trajectory data*, also including *pose data* for stationary objects. Ground truth about trajectory is required every time a benchmark requires an evaluation of position estimates (either of an element of the environment or of a part of the robot) provided by the robot system under test.

In RoCKIn, trajectory and pose data are collected by making use of a *motion capture system*. Systems belonging to this category are mostly used for cinematography, but their features are such that they are increasingly employed in robotics laboratories as well (e.g., to accurately track flying robots). They employ cameras to acquire video data about a volume of space where the objects to track move. Such objects are fitted with reflective *markers*; the video from the cameras is processed by the motion capture system in order

to estimate the position in space of the markers. By knowing (thanks to prior calibration) where the markers are placed on the objects, the system is therefore able to provide an estimate of the spatial pose of the objects.

The motion capture system chosen by RoCKIn is the OptiTrack system, manufactured by Natural Point, Inc. The choice of this system has first been narrowed down (using performance criteria) to choosing between similar products from Natural Point and its competitor Vicon; subsequently, the final choice was made based on cost-effectiveness. The OptiTrack setup used by RoCKIn includes: 12 infrared cameras (model S250e) with built-in IR LED illuminators[3], the Motive software package[4], as well as the necessary IR markers, mounting gear, network components, and data-acquisition PC. For RoCKIn, only the capability for tracking *rigid bodies* of the OptiTrack system is used: in fact, while the system can be also used to track deformable objects (such as the body parts of human actors), this capability is not sufficiently useful for RoCKIn to justify its considerable additional cost. The RoCKIn motion capture system is able to reliably track multiple rigid bodies at a high frame rate (250 fps) over a volume of up to a few tens of cubic metres (depending on obstacles and camera positioning), with millimetre-level accuracy in position tracking. This system has been chosen because it is a good trade-off, considering both the requirements and the budget of RoCKIn. As described in Part II of this document, for the 2014 RoCKIn Competition the motion capture system has been augmented with additional OptiTrack elements (cameras and PCs running motion capture software). As expected, such an augmentation improved the coverage and reliability (in terms of reduction of drop-outs in pose collection) of the system.

2.3.2 Installation of the motion capture system

The motion capture system chosen by RoCKIn is a commercial product, provided as a package. Except for some some quite basic configuration parameters, its operation cannot be finely tuned or adapted to the context of RoCKIn. However, this does not mean that the motion capture system is "plug-and-play": quite the contrary indeed, according to our actual experience with it. In fact, the capabilities of the system depend crucially on its installation, and installation choices have a strong impact on the quality of the system output in terms of

- coverage (i.e., actual volume where objects can be tracked);
- quality of tracking (including such elements as errors on reconstructed pose, smoothness of trajectory, absence of "dropout" time intervals);
- robustness of tracking (including resistance to occlusions, management of ambiguity between different marker configurations, and other elements).

In practice, installation of the motion capture system is at least as important for RoCKIn Benchmarking System as the motion capture hardware and software. For this reason, this Subsection is devoted to describing how the motion capture system has been installed and configured for RoCKIn.

[3] http://www.naturalpoint.com/optitrack/products/s250e/
[4] http://www.naturalpoint.com/optitrack/products/motive/

Reaching a satisfactory result is especially difficult when the system is installed in non-dedicated, unstructured (e.g., because of strong natural lighting) environments: precisely the case of RoCKIn. This section is dedicated to providing some guidelines for system installation. It provides information about how the installation of the RoCKIn motion capture system done at the 2014 Competition, which in turn is the result of significant accumulated installation experience. What follows is intended as a short guide for researchers interested in similar systems. Chapter 3 will provide additional information about both the setup of the system and its limitations.

Before the motion capture system can be used, it needs to be installed, pointed, and calibrated. Calibration is a specific procedure managed by the Motive software package. It makes use of a special "wand", which the operator is required to move over the observed volume, while the Motive acquires and processes calibration data[5]. Once familiar with such procedure, an operator can calibrate (or recalibrate) the system in a quarter of an hour. Since recalibration is required every time a camera is moved, unless cameras are installed in a very stable way it is advisable to recalibrate fairly frequently (e.g., once a day).

Installing and pointing the cameras requires considerably more time than calibrating the OptiTrack system. To give an idea about this to the reader, if the environment includes many obstacles (such as the RoCKIn testbeds) a motion capture system similar to the one used by RoCKIn requires several hours to be installed and optimally pointed. This is true even if the work is done by people experienced in the setup. An additional difficulty is related to the fact that getting the best coverage for each camera in environments where obstacles are present usually requires that cameras are mounted overhead, on supporting structures. This in turn makes installation more difficult and time-consuming. For all these reasons, it is important that the mountings are as stable as possible: repositioning is in fact very time-consuming. Some guidelines about what "optimally pointed" means will be provided by the remainder of this chapter, and especially by Section 3.3.

Each Natural Point S250e camera produces square images of 832 by 832 pixel; width of the field of view is 56 degrees. For reasons that will be explained shortly, it is absolutely necessary that significant superposition occurs between the fields of view of the cameras. Within limits, superposition can be increased by moving the cameras away from the observed volume. However, maximum distance from the markers is limited by the fairly low resolution of the image sensor: as the image of each marker becomes smaller on the image sensor, the system loses its capability to reliably recognize the marker. Practical experience shows that tracking distances up to 10 m are reasonable using the largest standard markers (spheres with a diameter of 19 mm). For the RoCKIn Competition, cameras for the acquisition of the trajectory ground truth data have been mounted at around 4 m from the ground, all around the two testbeds (one each for RoCKIn@Home and RoCKIn@Work), on truss-based support structures designed to be stable and vibration-free. Figure 2.3.2 shows a sketch of the testbeds with the trusses for the motion capture system is presented in and part of the actual installation in Toulouse.

After pointing and calibration, the required rigid bodies must be defined. This corresponds to the operation of mounting markers on the objects to track. Such objects must be rigid: no possibility of relative motion between markers associated to the same rigid body must be present, or the system will cease to recognize it. At least *3 markers per rigid body* are required; in practice, given that occlusions occur very often (because

[5]Additional information (including videos) about this calibration procedure is available through the Natural Point website.

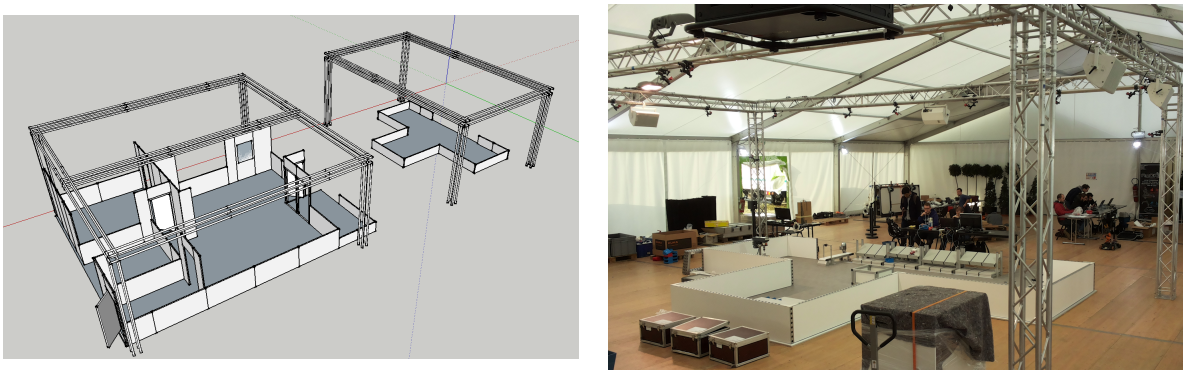


Figure 2.1: Left: CAD sketch of the testbeds, showing the trusses dedicated to supporting the motion capture cameras. Right: the actual truss built above the RoCKIn@Work testbed at the 2014 RoCKIn Competition in Toulouse. The little black objects underneath the truss are motion capture cameras on adjustable mountings.

the rigid body itself is not transparent or because of other obstacles, including the other markers) it is advisable to use more than the minimum allowed number of markers. The maximum allowed number of markers per rigid body is 7. Even with 7 markers it is not infrequent to lose track of a rigid body unless its shape, the positioning of the markers and the surroundings are favourable.

Markers of different types (adhesive pads or 3D spheres) and sizes are available. We found that adhesive pads lead to poor performance. The best performance for RoCKIn requires use of 3D markers: the larger spheres (diameter of 19 mm) are the best whenever their size does not prevent their installation.

To be able to estimate the location in space of each marker, at least *2 cameras must be able to see the marker* at any time. This is an absolute minimum, though; in practice, performance is very unstable unless at least 3 cameras see each marker. On the other hand, it is not necessary that such 3 cameras remain the same over time: handover from one camera to another occurs with little or no disruption of localization. To estimate the pose of a rigid body, at least 3 of its associated markers must be located by the system. So, to allow localization, *at least 3 of the markers of a rigid body must be seen by at least 2 cameras each.* Again, this is an absolute minimum: to achieve a robust localization it is best to exceed it significantly.

While the above constraints may be easily met in a specially structured environment (such as the studios where motion capture systems are used for cinematography), practical experience taught us that they are extremely challenging in unstructured and partially controllable settings such as a robot competition. Additional information can be found in Part II.

2.3.3 Time synchronization

For benchmarks that make use of both *external benchmarking data* and *internal benchmarking data* (see Section 2.1), it is important that external and internal data are time-synchronized, since only in this case data for benchmarking can be meaningfully compared. As external data are produced by the testbed while internal data are provided by the robot, time synchronization requires that the clock(s) of the robot are synchronized with the clock of the testbed.

In RoCKIn Competitions, time synchronization between robots and testbeds is ensured by using established network sync protocols. These are networking protocols for clock synchronization between computer systems over packet-switched, variable-latency data networks, and have a client-server structure. For RoCKIn, **Network Time Protocol** (NTP) has been chosen as the preferred protocol: being a "mainstream" protocol, NTP is very well supported over the widest range of systems and has been successfully employed at the 2014 RoCKIn Competition. Within a LAN (i.e., without relying on the internet), the typical time accuracy of NTP is below the millisecond. External benchmarking data deal with the macroscopic actions of the robot: for the type of actions required to participate to the RoCKIn Competitions, observable changes over 1 ms time intervals are negligible. In fact, the RoCKIn benchmarks are designed in such a way that extremely high-resolution over time in observing robot actions is not required. For this reason, NTP was expected to fulfill the needs of RoCKIn comfortably. As an example, a robot moving at a speed of 2 m/s (which corresponds to around 7 km/h, i.e., the speed of a human walking briskly) changes its position by 2 mm over 1 ms. Given that the tracking system has an accuracy of millimeter level, a time sincronization within 1 ms is considered fully acceptable.

Until NTP was tested at the first RoCKIn Competition, PTP (Precision Time Protocol) was considered a possible alternative in case the performance of NTP proved to be unsatisfactory. PTP is a time synchronization protocol similar to NTP in its structure, but with higher performance[6]. However, after the first RoCKIn Competition NTP has been definitely chosen, having shown more than sufficient precision.

Naturally, synchronization requires that all machines participating to a RoCKIn competition, including those on board of the robots, share a common clock. This requires the presence of an accessible NTP server and -more importantly- that all these machine run a NTP client connected to such server. Fortunately, well-established off-the-shelf NTP clients (with small computing power footprint) are already available for all major operating systems and were successfully employed at the RoCKIn 2015 Camp and Competition. In particular, (chrony - http://chrony.tuxfamily.org/) has been recommended to teams; additional information about its configuration is available in Appendix A.

2.3.4 Data transmission

For the RoCKIn Competitions, it is required that the metrics of the benchmarks can be applied to the data during the benchmarks or immediately after. This requires that suitable methods to transmit and/or store the data as they are produced must exist. Unfortunately, this problem is not as simple as it can appear: first of all, because the volume of data may be large (for instance, if video data are involved); secondly, because the chosen method to manage the data should be adaptable to the vast majority of the participating robots, whatever their hardware and software architectures.

The first, important technical decision to be taken is between *transmission* and *storage* of data for benchmarking. This is especially relevant for *internal* benchmarking data, i.e. those produced by the robot under test (see Section 2.1). The choice between transmission

[6]PTP (http://ieee1588.nist.gov/) significantly improves on the performance of NTP. Over a LAN, its typical accuracy is below the nanosecond, thus greatly exceeding the expected needs of RoCKIn. However, being much less widespread than NTP, PTP is not so widely supported. More importantly yet, PTP has a more aggressive approach towards manipulating the system clock with respect to NTP. This can be a drawback when fluctuating network links are involved: a typical situation when wireless networks are used, especially in the very noisy context of a robot competition where several wireless LANs are active over the same, small, area.

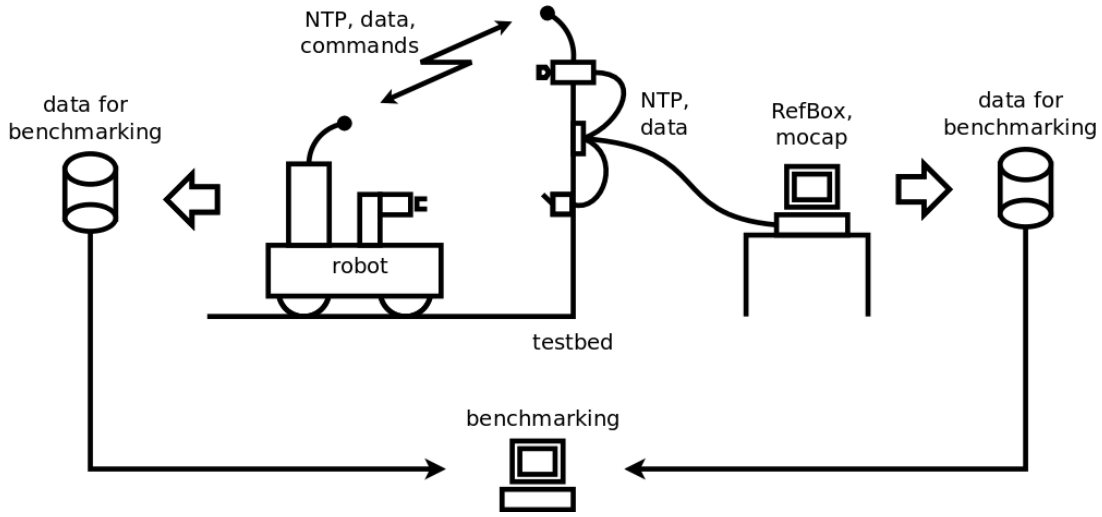


Figure 2.2: Simplified view of the architecture of the RoCKIn data collection and processing system for benchmarking.

and storage mainly depends on practical issues. In fact, storage can be managed locally (e.g., by writing on a USB stick) by the device that produces the data, e.g., by the robot system under test. On the other hand, transmission requires that the communication network between robot and testbed has sufficient bandwidth, it is not subject to drop-outs, and so on.

The problems related to data transmission do not occur, or can be solved before the benchmarking experiment occurs, for *external* benchmarking data (as defined by Section 2.1). External data for benchmarking include those produced by the motion capture system described by Section 2.3.1 and all data produced by the RoCKIn testbed itself (including human referees). For these data, every feature (such as format or bitrate) is known beforehand, and additionally wireless transmission can be avoided.

For the acquisition of internal benchmarking data, the RoCKIn Benchmarking System currently relies on data storage. Experience done at the RoCKIn events (Camps and Competitions) confirmed that in such environments the reliability of wireless networks is erratic at best; moreover, the volume of data to be transmitted is significant. For these reasons, robot data has been (and will be for the foreseeable future) collected by providing each team with a USB stick just before each benchmark, then retrieving the stick at the end of the benchmark.

Of course, having set up a methodology for the collection of robot data does not ensure that the data get produced, or that they correspond to what has been requested. At past RoCKIn events (Camps and Competitions) this has, in fact, often been a problem. Subsection 2.3.6 will deal with this problem and with the solutions to it chosen by RoCKIn.

2.3.5 System architecture

We can now describe the architecture of the RoCKIn system for the collection and processing of data for benchmarking. This system has first been used at the first RoCKIn Competition in November 2014, and its simplified outline is shown in Figure 2.2. The system has subsequently been further developed for the 2015 RoCKIn Competition, also building on the work done at the 2015 RoCKIn Camp.

Figure 2.2 shows the general outline of the system, and it is useful to understand where

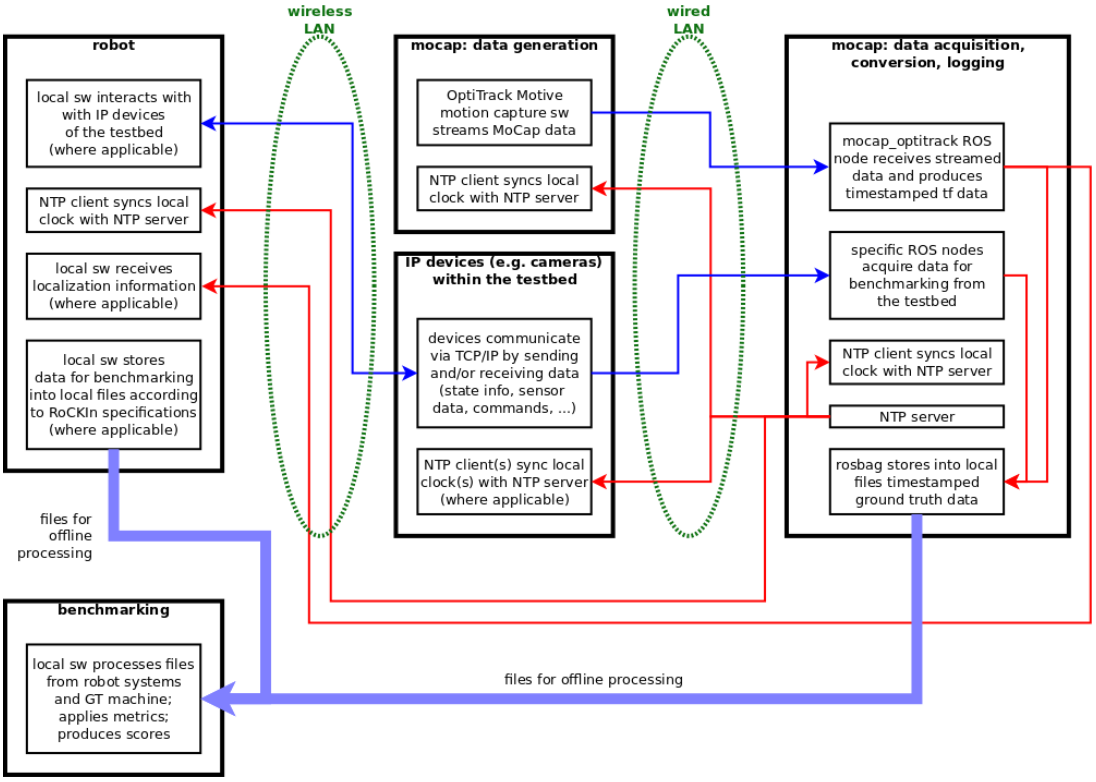


Figure 2.3: Architecture of the RoCKIn data collection and processing system for benchmarking. The modules in this Figure are the same of Figure 2.2.

data for benchmarking (both internal and external) are actually produced. Internal data come from the robot, while external data are generated by the Referee Box (i.e., the machine(s) that manage the competition, also called RefBox) and the machine(s) that produce motion capture data (mocap). The figure also shows how the testbed can include data-exchanging devices connected to the RefBox. Figure 2.3 provides additional detail about the modules involved in the process of generating and using benchmarking data in the system of Figure 2.2. The description of Figure 2.3 is still a high-level one focusing on exchanged information. Section 2.4 will describe how the diagram in Figure 2.3 is implemented using a networked system of computers.

The RoCKIn Benchmarking System (in the version used at the 2014 Competition) manages the exchanges of data shown by Figure 2.3 according to the following design choices:

- *Data formats* are chosen among the available message types provided by ROS. This decision has a number of reasons: first, that almost all the teams participating to the RoCKIn events (Competitions and Camps) relied on ROS and thus such choice made their work significantly easier; second, that such formats are suitable for the task and already fully documented; third, that a set of readily available tools for creation, inspection and storage of such data is already available as part of ROS. For the (very few) teams which did not use ROS, information and guidelines to use ROS libraries to use ROS-compliant data format from external software systems were provided.

- *Data transmission* is managed in two different ways. For low-bandwidth messages (e.g., notifications of events as detected by the robot) or interaction with the testbed

(e.g. interaction with IP cameras), a wireless nework is used. Logging of the internal data for benchmarking (as defined in Section 2.1) relies instead on providing teams with USB sticks where the robot is required to log the data. Immediately after each benchmark the USB key is retrieved and the data dumped to the benchmarking machine. The justification for this design choices is provided by Subsection 2.3.4. Subsection 2.3.6 is dedicated to the (not trivial) problem of getting teams to produce and store correctly the required data.

- *Data for benchmarking* required for the benchmarks, i.e. what data the teams are actually required to log, is specified in the Competition Rulebooks. Additionally, direct interaction between RoCKIn personnel and the teams at the Competitions is used to help teams actually follow such specifications and ensure that they are aware of the need to do so. Also this topic will be tackled by Section 2.3.6.

2.3.6 Collection of robot data

According to the definitions of Section 2.1, data used for benchmarking which are produced internally by a robot are called *internal benchmarking data*. Collection of internal data for benchmarking is especially complex because it requires the involvement, and active participation, of the teams. This creates a problem: as already explained, teams are required to perform additional activity for internal data collection. Such activity is often perceived by the teams as "wasted effort" with respect to their goal of optimizing their robot's ranking in the Competition. Therefore, some pressure must be applied in order to convince the teams to devote the necessary effort (not much, actually) to set up the acquisition of internal data. This is not a trivial issue, and requires careful consideration, as the quality of team participation is crucial in making the benchmarks actually feasible.

Notwithstanding the efforts of RoCKIn, at the 2014 Competition most of the teams did not put much effort into following the specifications and procedures related to data collection. The result of this was that internal data for benchmarking produced by the teams was often incomplete and/or non-compliant. This is understandable for two reasons: first, because the need for precise and comprehensive data collection procedures is, with respect to other existing robot competitions, a novelty introduced by RoCKIn, and some time will be needed before it becomes a "mainstream" activity; second, because in the frantic setting of a robot competition it is difficult to convince participants to devote part of their time and work to activities that they perceive as not improving to their chance of getting a good placement in the ranking. In general, RoCKIn manages its activities with full knowledge of the fact that benchmarking competitions (and the consequent need for high-quality benchmarking data) is new in the field of robotics. Therefore RoCKIn has to find a compromise between pushing teams towards correct data collection procedures and not discouraging them from participating to the RoCKIn events.

With the above contrasting needs in mind, two lines of action have been pursued to improve the situation in view of the forthcoming 2015 RoCKIn Competition. Precisely:

1. Providing teams with *support* in terms of instructions, software, templates, and direct human help. Additional tools to help teams understand what they are required to log and how to do the logging have been and will be prepared. In particular, at the 2015 RoCKIn Camp a specific "Benchmarking Kit" (also valid for the 2015 Competition) has been provided to the participating teams. Further information about this Kit will be provided in Chapter 5.

2. Making good-quality data collection *valuable* to teams. Data collection and compliance of such data to specifications have been included among the elements that define the ranking of a team in the RoCKIn Competitions[7]. Suitable procedures for data inspection during the Competition have been defined and tested at the 2015 Camp and will be applied at the RoCKIn Competition 2015.

As already noted, one trend that alleviates this problem is the increasing reliance of teams on ROS (Robot Operating System). While not without defects, ROS provides a wide set of ready-made tools for data publication, acquisition and logging that simplify the effort required from teams in order to collect internal data for benchmarking. Additional information about the tools provided by RoCKIn to promote correct data collection by the teams at the RoCKIn Competitions is provided in Subsection 2.4.3.

2.4 Implementation of RoCKIn Benchmarking System

This Section provides an overview of the current implementation of the RoCKIn Benchmarking System. Such implementation corresponds to the system used at the 2014 RoCKIn Competition in Toulouse and also (in a reduced version) at the 2015 Camp in Peccioli. Additional information about the benchmarking setup at the 2014 Competition is available in Chapter 4, while technical details of the RoCKIn Benchmarking System (including operating instructions) are provided by Appendix A. In the following of this Section, we will only focus on some aspects of the implementation, and particularly on the elements that are new with respect of the contents of Section 2.3.

2.4.1 System modules

The first point that must be highlighted when comparing the high-level description of Section 2.3 with the actual implementation of the RoCKIn Benchmarking System is that the second includes several additional modules. Importantly, among these there are some (described below) that have great importance even in an abstract view of the System.

The reason why these additional modules were not presented in Section 2.3 is that this Deliverable is concerned with data collection, while the additional modules are not involved in it. For the same reason, this Subsection will only provide basic information; all details are left to Appendix A.

The reason why the implementation of the RoCKIn Benchmarking System includes requires additional modules is that the System is not a stand-alone device. On the contrary, it is an integral part of the infrastructure of the RoCKIn Competitions. As such, while the core function of the System remains that of acquiring data for benchmarking (in the sense of Section 2.1), the RoCKIn Benchmarking System also includes several additional modules required to integrate it with the other elements of the Competition and to make actual benchmarking procedures feasible and practical. So, a first class of additional (with respect to those described by Section 2.3) modules that belong to the RoCKIn Benchmarking System are those that contribute to the management of the Competition (for instance by controlling the progress of a benchmark and/or by interacting

[7]More information about how this has been done are available in version 3 of the RoCKIn Rulebooks, i.e. Deliverables D2.1.3 and D2.1.6. For instance, correct data collection has been added to Task Benchmarks as one of the *achievements* that determine the score.

with the Referee Boxes), perform data management and storage, and execute monitoring and verification tasks.

Other very important modules of the RoCKIn Benchmarking System introduced by this Section are those concerned with **automatic performance evaluation**. Automatic (or machine-assisted) evaluation is based on computer analysis of the data for benchmarking (as defined by Section 2.1) and is not feasible for all benchmarks. In particular, the current version of all RoCKIn Task Benchmarks (TBMs) rely on human evaluation to apply the performance metrics: more precisely, metrics are based on objective criteria (a property that any benchmark must possess) but they require human capabilities to be applied. Therefore, currently TBMs are not susceptible to automatic evaluation[8]. This is mostly due to the system-level approach of the TBMs, where robots are asked to execute complex tasks and performance evaluation compounds specific aspects of such execution (please see the Rulebooks for details).

For automatic (or semiautomatic) performance evaluation it is necessary that performance metrics can be specified in machine-applicable terms (thus ruling out high level descriptions), and that their application only requires machine-available data (thus ruling out visual inspection by human referees). These are stringent limitations, which *de facto* restrict the field of application of automatic performance evaluation to Functionality Benchmarks (FBMs). FBMs focus on specific abilities of the robots: the functionalities. Thus, FBMs are much more closely specified than TBMs, and sometimes only involve machine-generated data: in some cases, then, FBMs allow automatic or assisted analysis. For example, FBM1@Home and FBM1@Work (Object Recognition) satisfy these constraints: performance metrics only consider the difference between ground truth pose of objects (provided by the motion capture system) and reconstructed pose, and comparison between actual (i.e. specified by the Benchmarking System) and reconstructed object class. For this reason, at the RoCKIn Competition 2014 performance analysis for FBM1@Work and FBM1@Home was performed automatically by the RoCKIn Benchmarking System. For contrast, let's consider now one of the elements determining the score in Task Benchmarks involving robot navigation. The robot receives a penalty every time it bumps against a piece of furniture or structural element, while bumping against people leads to disqualification. Automatizing the application of such criteria would be extremely complex, actually making it unfeasible.

2.4.2 Hardware

Together, Section 2.3 and Subsection 2.4.1 described all the elements present in the high-level model of the RoCKIn Benchmarking System. However, such elements require a implementation in terms of hardware and software components. Such components are physically realized by a collection of machines: some of these are dedicated devices (e.g., cameras of the motion capture system); other are general-purpose PCs running software. Except for the PCs that are part of the motion capture systems (running the Natural Point Motive [9] software), all other PCs run custom software systems written for RoCKIn.

Figure 2.4 illustrates the hardware composing the RoCKIn Benchmarking System (including the robot, which is part of the system as it collects internal benchmarking

[8]More precisely: given the rules for the benchmark and the current state of the art, the amount of effort required to automatically assess robot performance in the Task Benchmark would far outweigh the benefits.

[9]http://www.optitrack.com/products/motive/

data). The elements are by wired and wireless TCP/IP networks. The wireless network is only used to exchange low-bandwidth data between robot and Referee Box, for the reasons illustrated by Subsection 2.3.4. The wired network is split into two subnets, one for the RoCKIn Benchmarking System and the other for the Referee Box and the network devices integrated into the testbed.

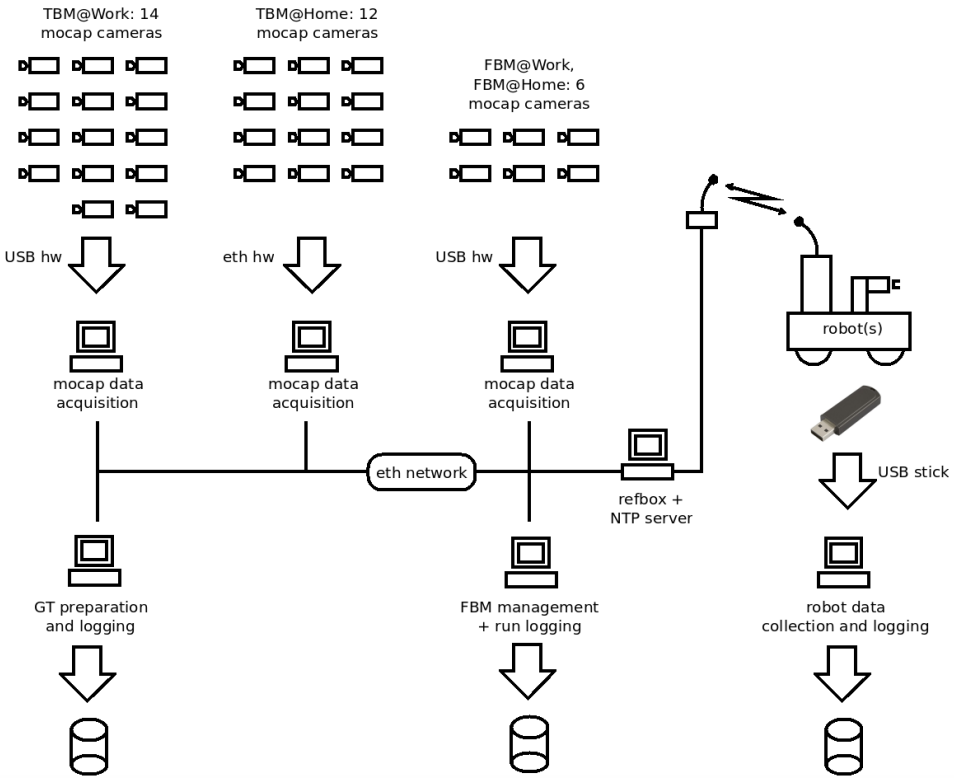


Figure 2.4: Structure of the RoCKIn Benchmarking System used at the 2014 RoCKIn Competition in Toulouse, France.

Figure 2.4 shows that the current version of the RoCKIn Benchmarking System includes three separate OptiTrack motion capture systems. At the 2014 Competition in Toulouse, these covered three distinct areas: FBM1@Home/@Work, TBM@Home and TBM@Work (see Chapter 4 for further information). Each of the OptiTrack setups includes a Windows PC executing the Motive software, which is required to interface with the cameras and track the marker sets. A *marker set* is a 3D object fitted with IR-reflective markers, that the motion capture system is configured to identify as a rigid body in order to reconstruct its 6DOF pose. Processed data is streamed over UDP to the client PCs described below. For ground thruth (GT) logging, a Linux PC running ROS is used. On this PC, a ROS node called *mocap_optitrack*[10] receives the UDP packets broadcast by Motive and publishes, for each tracked marker set, the corresponding *tf* frame, the 3D pose and the 2D pose. The GT is then logged by a set of other ROS nodes. To manage the FBM1@Home and FBM1@Work Functional Benchmarks and to perform automatic performance evaluation (as described in Subsection 2.4.1), a second Linux PC running ROS is used. This PC receives data from the motion capture too, when needed by the benchmark, and runs a set of ROS nodes that interact with the Referee Box, to control the execution phases of the benchmark and to compute the final score.

[10] http://wiki.ros.org/mocap_optitrack

As explained in Subsection 2.3.3, clock synchronization among all the PCs is provided by NTP. The GT logging PC and the FBM1 PC run the Chrony NTP client, while the NTP server runs on the @Home Referee Box.

Finally, an additional PC is used to save all the data logged by the robots and collected on USB sticks.

2.4.3 Collection of robot data

In RoCKIn Competitions, data collection is very important: firstly, because such data are necessary for benchmarking and benchmarking is an integral part of the Competitions; secondly, because acquisition and publications of datasets is one of the goals of RoCKIn. As explained in Subsection 2.3.6, obtaining data from participating teams is not as easy as it may seem; on the contrary, it is necessary to *convince* teams to spend the necessary effort. Subsection 2.3.6 also provided a short analysis of why teams are (relatively) insensitive to the issue, and described the strategies used by RoCKIn to change the situation.

A key element of the above strategies, and the most complex to set up, is support. Experience at the first RoCKIn Competition in 2014 showed that a good way to encourage teams to devote more effort to the collection of robot data is to provide them with complete, comprehensive and directly applicable documentation and software. Therefore, benchmarking work before and during the RoCKIn Camp 2015 in Peccioli was focused on understanding the needs of teams, preparing suitable material, personally helping team members to use it, collecting feedback from them and revising the material until it was satisfactory. While not a competition environment, the Camp is nonetheless an occasion where real teams work, and compete, to obtain the best possible results over a short period of time: therefore, a convenient testing ground in view of the 2015 RoCKIn Competition. The work on support material continued after the end of the Camp; its final result has directly been included in the Rulebooks for the RoCKIn Competition 2015 in order to make it maximally visible to teams. Chapter 5, which is dedicated to the RoCKIn Camp 2015, includes an example of this material (Section 5.3).

Data collected (both from robots and from the RoCKIn Benchmarking System) at the RoCKIn events are post-processed, published on the internet and freely downloadable. Access to the data is provided through the section of RoCKIn's website dedicated to benchmarking[11]. Details about what data has been published are provided in the relevant Chapters of Part II. Specifically, Chapter 4 will describe the data collected at the 2014 Competition, while Chapter 5 will do the same for the data collected at the 2015 Camp.

[11] http://rockinrobotchallenge.eu/benchmarking.php

Part II

Practical Experiences

Chapter 3

RoCKIn Camp 2014 (Rome, Italy)

This Chapter has been written after the end of the 2014 RoCKIn Camp[1] (Rome, January 26th-30th, 2014). One of the objectives of the Camp was to investigate in a real-world setting the feasibility and effectiveness of the benchmarking procedures envisioned for the RoCKIn Competitions. This chapter is concerned with the results of such investigation. Additional information about the activities of the 2014 RoCKIn Camp can be found in Deliverable D5.2 (RoCKIn Camp 2014 Report).

Note. This Chapter of Deliverable D2.1.8 is unchanged from Deliverable D2.1.7. The recommendations provided by Section 3.3 have been taken into consideration in the design and setup of the 2014 RoCKIn Competition.

3.1 Camp and Competitions

The 2014 Camp was an excellent setting to test some aspects of the benchmarking procedures: in particular, those concerning set up, calibration and data acquisition. While not fully possessing the frantic pace of a competition (where every procedure must be painstakingly prepared beforehand in order not to disrupt the flow of the event), under many points of view the Camp provided a similar environment and similar challenges. In particular, the Camp included:

- competition-like testbeds;
- competition-like interference problems due to the concurrent access to the testbeds by the teams and the benchmarking personnel (in fact, the testbeds also acted as testing areas for the teams);
- competition-like resource allocation problems due to the need to perform simultaneous benchmarking activities on two separate testbeds (RoCKIn@Work and RoCKIn@Home);
- competition-like data acquisition problems (e.g., set up and calibration in real-world conditions, interference from objects different from the robot under test, unexpected disruptions);

[1] http://rockinrobotchallenge.eu/camp2014.php

- competition-like limitations to the quantity and quality of ground truth data that could actually be collected (e.g., due to limitations in the number and/or positioning of motion capture cameras).

What could not be tested at the Camp is the *processing* of ground truth data, as this was neither defined nor implemented at the time of the Camp. Nonetheless, the Camp offered valuable lessons regarding the way such processing should be organized in order to proceed smoothly without interfering with the other activities.

It is important to stress that, while the robotics community (and the RoCKIn Consortium in particular) has collected a large experience in running robot competitions, experience in performing objective benchmarking of robot performance in the context of competitions has to be built from scratch. RoCKIn hopes to provide a valuable starting point for that.

3.2 Experience at the Camp

As explained in Chapter 2, in principle, RoCKIn benchmarking is based on the processing of data collected in two ways:

- external benchmarking data, collected by the testbed;
- internal benchmarking data, collected by the robot system under test.

During the Camp, both these types of data have been collected. The following of this section describes the aspects of this data collection activity that are relevant to describe the experience of the Camp in terms of data collection for benchmarking.

3.2.1 Data acquisition and logging

During the Camp, external benchmarking data consisted of the trajectories of rigid elements of the robots under test (such as the base or the wrist) and/or of objects that the robots interacted with. For this, the elements to be tracked had to be fitted with IR-reflecting markers which are part of the motion capture system. Figure 3.1 shows some examples of marker installation at the Camp.

Trajectory data have been generated using the camera-based commercial motion capture system already described in Chapter 2 (i.e., Natural Point OptiTrack). Each trajectory had the form of a time series of poses of the relevant robot element. Pose data generated by the OptiTrack system have been acquired and logged by a customized external software system based on ROS (Robot Operating System: the same middleware used by all participating robot systems). More precisely, logged data were saved as *bagfiles* created with the *rosbag* utility provided by ROS (`http://wiki.ros.org/rosbag`). Figure 3.2 shows the architecture of the logging system.

The choice of using a ROS-based system allowed translation of OptiTrack data into geometric transforms published on the `tf` topic of ROS (`http://wiki.ros.org/tf`): i.e., precisely the same form taken by the robot's estimate of its own pose. This enabled (subsequent) direct comparison between ground truth trajectories and trajectories estimated by the robots.

Internal benchmarking data have been collected as ROS *bagfiles* as well. In this case the files had to be created by running *rosbag* on the robot. This, as will be better explained

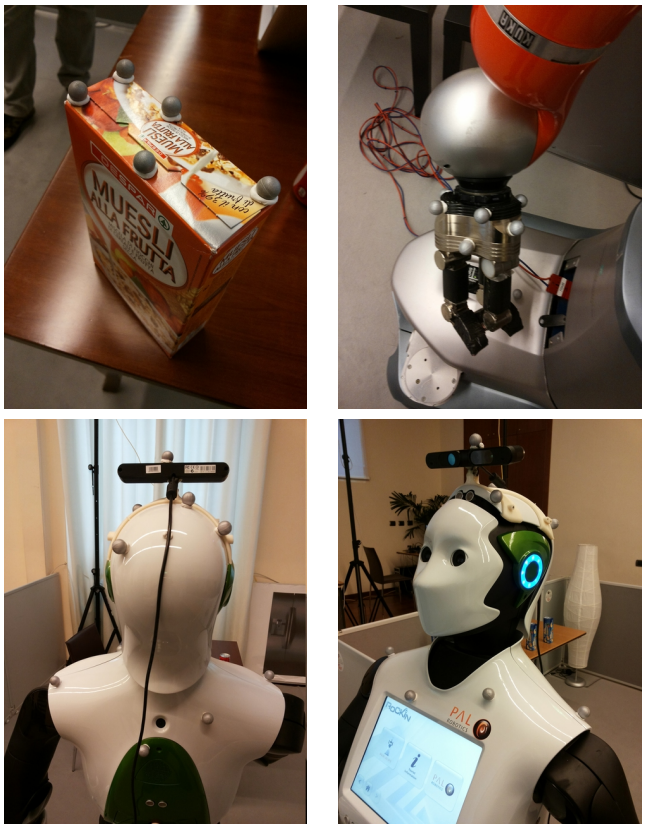


Figure 3.1: Examples of installation of markers (gray spheres) at the 2014 RoCKIn Camp. Clockwise from top left: on a box (the box had to be grasped by robots); on the wrist of a robot; on the back and front of the head and torso of a robot.

later in this chapter, did not prove to be a viable solution for the competition. Firstly, because many teams forgot to perform the manual running of *rosbag*; secondly, because (differently from what happened at the Camp, where all robots were ROS-based) at the Competition it is not possible to take for granted that *rosbag* will be available on all robots.

Subsequent comparison between OptiTrack-generated (i.e., ground truth) and robot-generated poses requires that they are aligned in time, as already explained in Chapter 2. For this reason, a special ROS node (i.e., a software module compatible with the ROS middleware) was developed and distributed to teams. Such node, which was required to be running both on the robots and on the machine that acquired ground truth data, simply generated (on a specific ROS topic) periodic messages including both machine time ("wall clock") and ROS time, thus enabling subsequent synchronization of the data in the bagfiles with ground truth data. Participating teams were required, before running their Camp demos, to perform manual alignment of their robot's clock with an external NTP time server[2]. The latter had been set up as part of the ground truth acquisition system. In practice, this manual alignment procedure proved to be unreliable, since in many cases teams forgot to apply it.

[2]For Linux-based computers such alignment was done by running the *ntpdate* command; on Windows-based machines a graphical tool is available as part of the clock system; while not tested at the Camp, *ntpdate* should be available in MacOS as well.

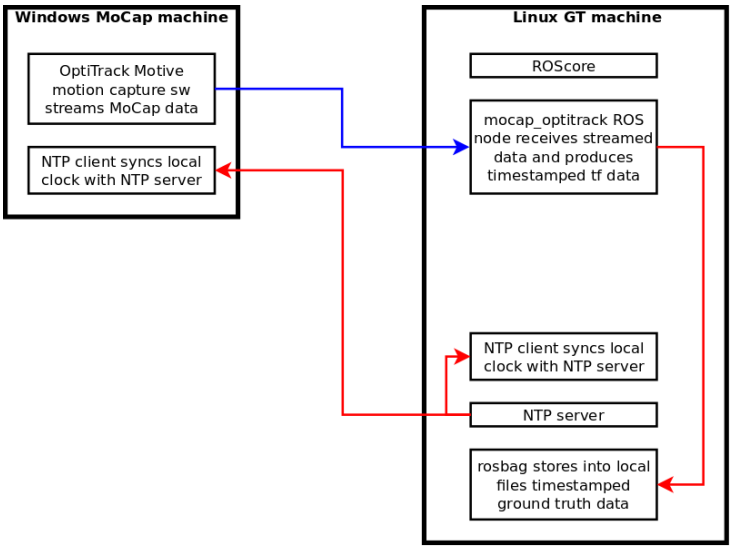


Figure 3.2: System used at the 2014 RoCKIn Camp to acquire and log trajectory data from the motion capture system. This is a subset of the system shown in Figure 2.3.

3.2.2 Physical setup of the motion capture system

For the 2014 RoCKIn Camp the RoCKIn@Home and RoCKIn@Work testbeds were installed in two separate halls. This led to the decision to split the 12-camera OptiTrack system into two 6-camera systems, one for each testbed. Figure 3.3 shows the two motion capture systems as installed at the Camp.

Considering the dimensions of the testbeds, using 6 cameras per testbed lies at the very lowest limit of what -in our practical experience- is feasible with the OptiTrack system. Below we will describe the issues that arose from that, and what compromises had to be made.

The first issue encountered is related to physical camera installation. The positioning of the testbeds within their respective halls proved to be very challenging from this point of view. For the Camp, cameras were expected to be installed on (3m-high, and therefore having a large footprint) tripods around the capture volume; however, in practice the available space around the testbeds proved to be insufficient for a good installation. The worst situation occurred with the RoCKIn@Work testbed, which -being adjacent to a wall and a passage area- only allowed cameras on 2 of its 4 sides (no space suitable for tripods was available inside the testbed). For RoCKIn@Home things were better (all four sides of the testbed were usable, with some limitation, thanks to the possibility of installing tripods inside the testbed). However, the presence of working areas directly adjacent to the testbed (and thus to the tripods) led to accidental motions of the tripods, and consequent need for recalibration.

Another issue related to camera positioning is the difficulty in ensuring a satisfactory capture volume. Capture volume is the region of space where reliable motion capture is possible, which strongly depends on the number and positions of cameras. As explained in Chapter 2, the OptiTrack system uses markers on the objects (rigid bodies) to be tracked. Figure 3.4 shows two of the rigid bodies fitted with markers which were built at the 2014 Camp.

In order to provide pose data, a minimum set of requirements must be fulfilled by each tracked object, in terms of number of separate markers perceived by the system

Figure 3.3: Motion capture cameras as installed at the 2014 RoCKIn Camp. On the left, camera setup around the RoCKIn@Home (up) and RoCKIn@Work (down) testbed. On the right, closer views of a single RoCKIn@Home tripod with camera (up) and of part of the RoCKIn@Work setup (down).

and number of cameras that perceive each marker. These requirements proved to be problematic for a real-world setting (such as the Camp) where the tracked objects are not optimized for motion capture and the number of cameras is low. This resulted in reduced capture volumes and occasional loss of tracking, especially close to the edges of such volumes.

One interesting thing that has been noted at the Camp is that small movements of the motion capture cameras after calibration (typically due to people accidentally bumping into the tripods) do not severely impair the precision of the OptiTrack system in localizing objects within the capture volume; conversely, such movements have a strong impact on the capture volume, reducing it. Both effects can be verified by placing the calibration "wand" in the location where the check has to be performed. For what concerns precision such verification is easy: the software provides a specific function, and (most importantly) only spot checks are needed, as precision is quite constant over the capture volume. On the other hand, the only effective way to check if the capture volume has reduced is to explore its boundaries with the "wand": an error-prone, time-consuming process. The final consequence of this situation is that it is necessary to make specific efforts to carefully prevent camera movements. Alternatively, a tight schedule of periodic recalibrations need to be established.

Figure 3.4: Two examples of rigid bodies fitted with markers, and a robot equipped with one of them.

The Camp offered an excellent occasion for building up experience in the optimal positioning both of the motion capture cameras and of the markers on the robots. With careful set up, dropouts in pose data were reduced to a minimum. However, to reach this result it was necessary to reduce the capture volume with respect to the overall volume of the testbeds. With the best available setup, covered testbed volume was around one half both for RoCKIn@Work and RoCKIn@Home. The capture volume for RoCKIn@Home was actually much larger than for RoCKIn@Work, as the overall testbed volume for @Home is around 6 times larger than that for @Work. This better result is due to a larger base area and, mostly, to the possibility to install the cameras all around the capture volume. This was not possible for RoCKIn@Work, as two out of four borders of the RoCKIn@Work testbed were not usable (one was a passage area, the other a wall.

A final issue related to the physical setup of the motion capture system was the preparation of the rigid bodies that the system had to track. When possible, special 3D "marker sets" were built and fitted to the robots. However, no standardized marker set was used: indeed, in most cases changes to the sets and/or their construction was done on the spot, during the demos. In some cases, and particularly with larger robots, it was found that the only possible approach was to fit the markers (with removable putty) directly on the robot's body. In all cases, the OptiTrack system had to be used during the demos to define the just-realized marker sets as rigid bodies, in order to track them. A side effect of this approach has been that the roto-translation of the intrinsic reference system of each rigid body w.r.t. the reference system(s) of the robot was unknown, and had to be deduced indirectly from acquired data. Such an *ad hoc* approach, while acceptable in a Camp, is not feasible in a Competition where the time schedule is tight and where it is not acceptable that the performance of the benchmarking system changes from robot to robot. Therefore, a general and standardized approach to marker positioning will be used for the Competition.

All in all, the 2014 RoCKIn Camp allowed experimental verification of a wide range of aspects and procedures related to the collection and logging of benchmarking data. Section 3.3 will translate such experience into a list of practical recommendations. These have been already taken into account in the design of the 2014 RoCKIn Competition, and will be in the design of the subsequent 2015 Competition. Moreover, some of the recommendations are not relevant to the benchmarks actually used for the 2014 Competi-

tions. In any case, all recommendations are provided here for the benefit of other research groups planning similar efforts.

3.3 Lessons learned

This Section is dedicated to outlining the lessons that the benchmarking experience done at the 2014 Camp has provided. What follows is strongly focused on the necessity of performing benchmarking in the context of the RoCKIn Competition: therefore it will be expressed in the form of a list of recommendations for the final design and setup of the Competition. Of course, such recommendations have already been taken into consideration in the design of the forthcoming 2014 RoCKIn Competition.

3.3.1 General recommendations

- It is infeasible to make benchmarking rely on explicit actions that a participating team must take when their robot is subject to test. Any activity required by the test must be automatically executed by the robot, without any intervention from the team. In other words, benchmarking aspects of the Competition should be made as *transparent* as possible to the teams.

Such "transparent benchmarking" can be supported by providing each team with a single piece of software (e.g., a ROS node) and by requiring that such software is running on the robot during the Competition. The testbed should be capable of detecting if the software is actually running: in this way each benchmarking experiment will be started only if and when the software is active, i.e., if it is certain that all the activities required to the robot to participate to the experiment will actually be executed.

For instance, the piece of software described above may: (i) connect to an NTP server on the testbed's wireless network to synchronize the robot's clock with that of the server; (ii) notify the testbed (again through the wireless network) that the node is actually running and synchronization has been achieved; (iii) start the logging of the data required for benchmarking.

The software provided by RoCKIn should be robust, lightweight in terms of required computing resources and possibly also provided well in advance w.r.t. the Competition, in order to let teams check that the performance of their robots is not affected when the software is running on the robot.

Issues raised by the above recommendation:

1. Forcing a robot to run a piece of software means forcing the internal architecture of the robot to be compatible with such software: therefore, it is a decision that cannot be taken lightly.

2. It is likely that the requirement of being lightweight in terms of resources requested to the RoCKIn software will set constraints on the benchmarking process. For instance, logging large quantities of data (e.g., video) may be useful for benchmarking, but incompatible with such requirement. A careful balance will have to be sought.

3. One issue to be managed is how logged data should be stored and/or transmitted. Data streaming over the testbed's wireless network is probably best avoided to avoid reliability issues interfering with the logging (and therefore with benchmarking):

therefore storage should be done onboard. To reduce operative problems and the risk of cheating, the best solution is that RoCKIn provides the teams with an external storage device (e.g., USB key): however, this introduces new requirements on the robots (such as the availability of a suitable free socket), which should be carefully considered.

3.3.2 Recommendations on testbeds

- A lightweight overhead truss (such as those used for theatre or concerts) should be available above the whole perimeter of each testbed area where motion capture has to be performed. The truss should be at 4m from the ground, and should be affixed to the ground (*not* to the testbed or its boundaries) in order to make it impossible to move it even in the occurrence of serious collisions with people, robots or other heavy objects. The horizontal bars of the truss will support the motion capture cameras, and should therefore not be subject to vibrations; moreover, sufficient space should be present around such bars to allow free positioning, moving and aiming of cameras and associated mounting systems.

3.3.3 Recommendations on the motion capture system

- Objects contained in the testbeds, as well as testbed features, that can interfere with motion capture (e.g. bright metal surfaces, reflecting surfaces, light sources) should be avoided whenever possible. As it is impossible to guarantee that such interfering objects will not be present in the vicinity of the testbeds, positioning and pointing of motion capture cameras should ensure that out-of-testbed objects cannot be perceived[3].

- During the Competition, only the robot subjected to a test should be allowed in the testbed for the duration of the test and its preparation. No people or extraneous material (which can create occlusions and/or otherwise interfere with the collection of motion capture data) should be present.

- Due to the tight time schedule of the Competition, it is necessary to have a means to immediately identify what was happening in each testbed at any arbitrary time (e.g., what robot was in there). Therefore, (conventional) video coverage of the whole volume of the testbeds should be acquired and saved for the whole duration of their opening hours. Time (synchronized with the NTP server of the relevant testbed) should appear superimposed on video frames.

- Any processing operation on benchmarking data to be done during the Competition should be implemented as a highly-automated script, written and tested well in advance of the Competition. In a competition setting, manually performing data processing is not feasible in terms of time and leads to errors.

[3]The OptiTrack camera management software allows masking of stationary video disturbances. This is done by defining masking shapes covering (small) areas of the field of view of single cameras. However, though possible, masking should be avoided. As cameras subjected to masking become partially "blind", masking introduces "blind spots" in the volume of space observed by the motion capture system, in the form of subvolumes that are perceived only by part of the cameras or, in the worst case, by none of them.

- As already observed, the number of cameras required to observe the testbeds strongly depends on the size and shape of the testbeds. For this reason, testbed dimensions and shape should carefully matched to the available motion capture hardware.
- For testbeds similar to those used for the 2014 RoCKIn Camp, the 12 cameras available to the POLIMI partner proved to be insufficient (good coverage over the whole volume of the testbeds would have required at least 20 cameras). If similar testbeds will be used and full coverage of them will be required, it will be necessary to restrict the capture volume (as done at the Camp). Otherwise, a higher number of cameras should be acquired (possibly only temporarily).
- As already explained, fitting motion capture markers to robots in a per-robot way is infeasible under the conditions of the Competition. Previously prepared *3D marker sets* (e.g., suitable non-deformable lattice-like objects provided with markers on vertexes) should be used instead.

Issues raised by the above recommendations:

1. 3D marker sets are a good solution to track mobile bases, but their use is infeasible for most end effectors. This strongly restricts the feasibility of collecting motion capture data for manipulation. Not wanting to return to customized marker positioning, two approaches to this problem are possible: (i) affix the marker set on the manipulated object instead (which should therefore be designed right from the start with this requirement in mind); (ii) restricting the collection of motion capture data to the tasks and cases where the marker sets can be used.
2. Robots participating to the Competition are expected to be very heterogeneous in their shape and structure. Devising 3D marker sets that can be successfully fitted to all participating robots is not a trivial task. In addition to mechanical problems and occlusions, issues of other type could arise: for instance, some teams at the Camp were unwilling to accept any mounting that could pose a risk of aestethic damage to their robot.

3.3.4 Recommendation on robots

The recommendations provided in the preceding part of this section lead to requirements that each robot system participating to the RoCKIn Competitions should comply with. Such requirements are listed here.

1. Each robot should possess the hardware and software necessary to interact with the testbed's wireless network. Additionally, the robot should be capable of keeping such network connection active for the whole duration of the benchmarking experiment (which prevents using the same hardware or software for other connections).
2. Each robot should be mechanically compatible with the *3D marker sets* prepared by RoCKIn, both in terms of providing suitable fixtures for their attachment to the robot and in terms of ensuring visibility of the markers by the motion capture cameras (i.e., avoiding that they get occluded by elements of the robot).
3. Each robot should provide a suitable free USB socket (possibly USB 3.0) to connect an external USB disk or stick provided by RoCKIn, for logging data.

4. Each robot should be endowed with sufficient processing power to manage task execution and logging at the same time.

5. Each robot should be capable to run any piece of software that RoCKIn provides to the teams, both in terms of processing resources and in terms of onboard software environment. As the RoCKIn project desires to be as inclusive as possible, this turns into a recommendation on RoCKIn software, which has to be multi-platform and require few resources.

Chapter 4

RoCKIn Competition 2014 (Toulouse, France)

This Chapter describes how the RoCKIn Benchmarking System has been set up and used at the first RoCKIn Competition[1] (Toulouse, November 26-30, 2014). This has been the very first RoCKIn Competition to occur, and the first public test both for the Benchmarking System and, more generally for the Competition procedures (including those related to benchmarking). Additional information about the activities of the 2014 RoCKIn Camp can be found in Deliverable D3.1 (Report on Progress of the Competition and Benchmark Activities).

The work of project RoCKIn at Toulouse was based on the experience gained by previous events, and especially by the 2014 RoCKIn Camp. As the activities and lessons of the 2014 Camp are already described by Chapter 3, Chapter 4 we will focus exclusively on the additional activities and insight related to the 2014 Competition. For the same reason, we will omit all information about the RoCKIn Benchmarking Systsem setup that was already provided in Chapter 2.

4.1 Benchmarking activities

As already said, the 2014 RoCKIn Competition was the very first one. As such, it had two main objectives: first, of course, that of being a successful and full-fledged robot competition with tests, scores and winners; but also, importantly, that of acting as a first real-world testing ground for many aspects of project RoCKIn. Such aspects include: the rulebooks; the organization and management of the Competition; the supporting infrastructure; the setup procedures (in particular for what concerned the construction of the RoCKIn@Home and RoCKIn@Work testbeds); and finally, the benchmarking activities. About last item, experience at the 2014 Competition confirmed the soundness of the design choices done for the RoCKIn Benchmarking System, as described by Deliverable D2.1.7 for what concerns data acquisition, and by Deliverables D2.1.2 and D2.1.5 (2014 version of the RoCKIn Rulebooks) for what concerns procedures and metrics.

Benchmarking activities at the Toulouse Competition belonged to two categories:

1. Collecting *data for benchmarking*. These data, according to the terminology introduced in Section 2.1, include internal data (from robots) and external data (from

[1] http://rockinrobotchallenge.eu/rockin2014.php

independent sources, most notably from the motion capture system described in Chapter 2).

2. Assessing the performance of the robot systems participating to the Competition.

For what concerns data collection, the system described by Chapter 2 was used. While we will not go again into the details of how such system is build, details about the specific setup for the 2014 Camp will be provided in the following of this Chapter. Some of the data was collected for subsequent publication by RoCKIn as datasets, not for immediate use as a benchmarking tool: dataset publication is one of the activities of RoCKIn that aim at supporting and encouraging research and, in particular, current and prospective participants. The data, along with those collected at the subsequent 2015 Camp in Peccioli, are available at `http://thewiki.rockinrobotchallenge.eu/index.php?title=Datasets`.

Teams participating to the Competition were provided with a support *wiki*. Part of this *wiki* was dedicated to benchmarking, and included containing instructions and tools to help teams in successfully setting up their robots for the benchmarks and collecting data. Figure 4.1 is a screenshot showing the beginning of this section of the Competition *wiki*.

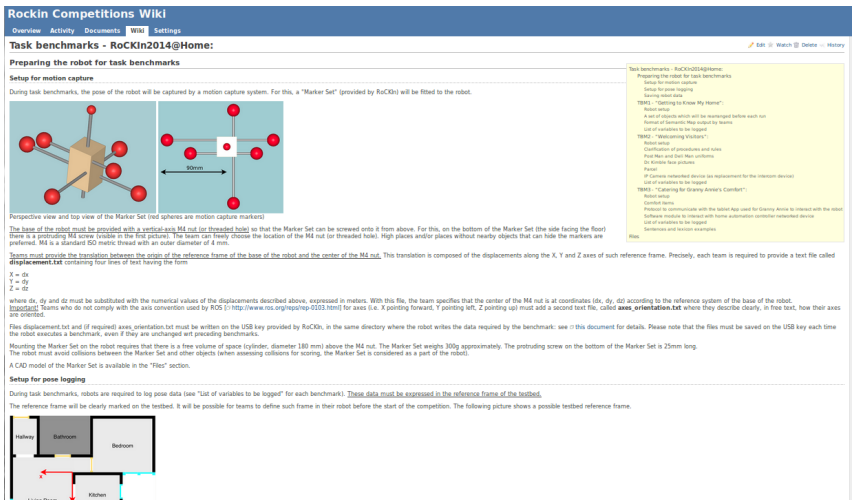


Figure 4.1: Fragment of the section of the RoCKIn Competition 2014 wiki dedicated to benchmarking.

For the assessment of robot performance, the evaluation criteria established by the 2014 version of the RoCKIn Rulebooks (Deliverables D2.1.2 and D2.1.5) were applied. These were based on a combination of human assessment (according to well-specified and objective criteria) and automatic assessment. In particular, for Functional Benchmarks 1 for RoCKIn@Home and RoCKIn@Work (object recognition) the RoCKIn Benchmarking System was used both to collect data and to assess robot performance automatically. While outside the scope of this Deliverable, nonetheless this assessment was part of what the RoCKIn Benchmarking System was used for at Toulouse. For this reason, a technical description of how the system was configured and used for that is available in Appendix A.

4.2 Setup

The general structure of the RoCKIn Benchmarking System at Toulouse was fairly complex: it comprised three separate motion capture systems, several computers and a network infrastructure (shared with the RoCKIn@Home Referee Box). Figure 4.2 shows such structure.

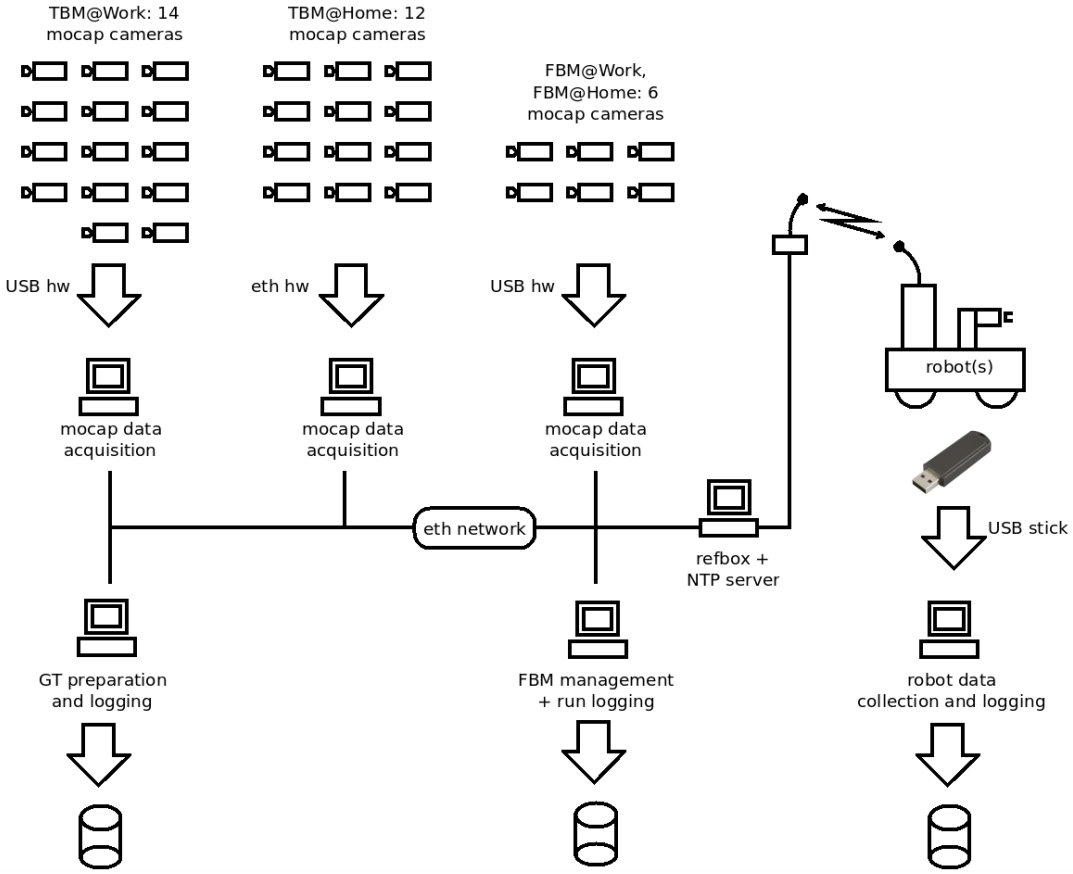


Figure 4.2: Structure of the RoCKIn Benchmarking System used at the 2014 RoCKIn Competition in Toulouse, France.

In the following of this Section we will provide additional information about specific elements of the setup for benchmarking at the RoCKIn Competition 2014.

4.2.1 Motion capture setup

The first RoCKIn Competition took place in Toulouse, France. More precisely, the venue was a large tent-like structure sited within "La Cité de L'Espace" (http://www.cite-espace.com/en#accueil). Here, three different benchmarking areas were built:

1. the testbed for the Task Benchmarks of RoCKIn@Home (TBM@H);
2. the testbed for the Task Benchmarks of RoCKIn@Work (TBM@W);
3. the special table where Functional Benchmark 1 (FBM1), Object Recognition, took place both for RoCKIn@Home and RoCKIn@Work.

Though the FBM1 area was set up on the periphery of the TBM area for RoCKIn@Home, the overall area to be covered by the motion capture system(s) was very large: approximately, one 100 square metre area for TBM@H and FBM1, and a slightly smaller one for TBM@W. An additional difficulty was posed by the fact that the TBM@H and TBM@W testbeds were not adjacent but separated by about 10 metres.

All in all, there was no possibility to obtain good coverage of all three areas with a single motion capture system. Additionally, it was considered convenient to dedicate a separate system to FBM1. Thanks to two kind loans, for which we thank RoCKIn partner BRSU and the Italian distributor of Natural Point products, FVR[2], we were able to set up three separate OptiTrack motion capture systems, as shown by Figure 4.2: a 12-camera system for the TBM@H, a system comprising 14 (less capable) cameras for the TBM@W, and a 6-camera system dedicated to FBM1.

Figure 4.3: Four snapshots of the benchmarking system at Toulouse. Clockwise from top left: trusswork with cameras (and spotlights) above the RoCKIn@Home testbed; trusswork above the RoCKIn@Work testbed; some of the PCs used by the Benchmarking System; motion capture cameras used for FBM1.

Motion capture cameras were installed on a strong truss 4 metres from the ground (see also Figure 2.3.2). The truss was specially designed for the motion capture system with two main aspects in mind: first, to provide a stable and non-movable platform for the cameras (at Camp 2014 we learnt that tripods cannot be trusted not to get moved);

[2] http://fvr-cgi.it/

second, to optimize coverage by locating cameras at the optimal height. This significant height necessitated special arrangement in terms of installation and cabling; on the other hand, it proved especially important for the TBM@H, as the testbed included 2m-high partitioning walls. Only by mounting the cameras up above the ground we were able to get good camera coverage also *beyond* the partitions. Figure 4.3 shows parts of the camera arrangement of the three motion capture systems installed in Toulouse.

4.2.2 Robot physical setup

While most of the setup to enable collection of the data for benchmarking was done by RoCKIn, some setup activities were required from the participating teams. These activities were of two types: (i) fitting the robot with **marker sets** provided by RoCKIn in order to make them recognizable by the motion capture system; and (ii) logging robot data. These setup procedures were carefully described in the wiki supporting the Competition.

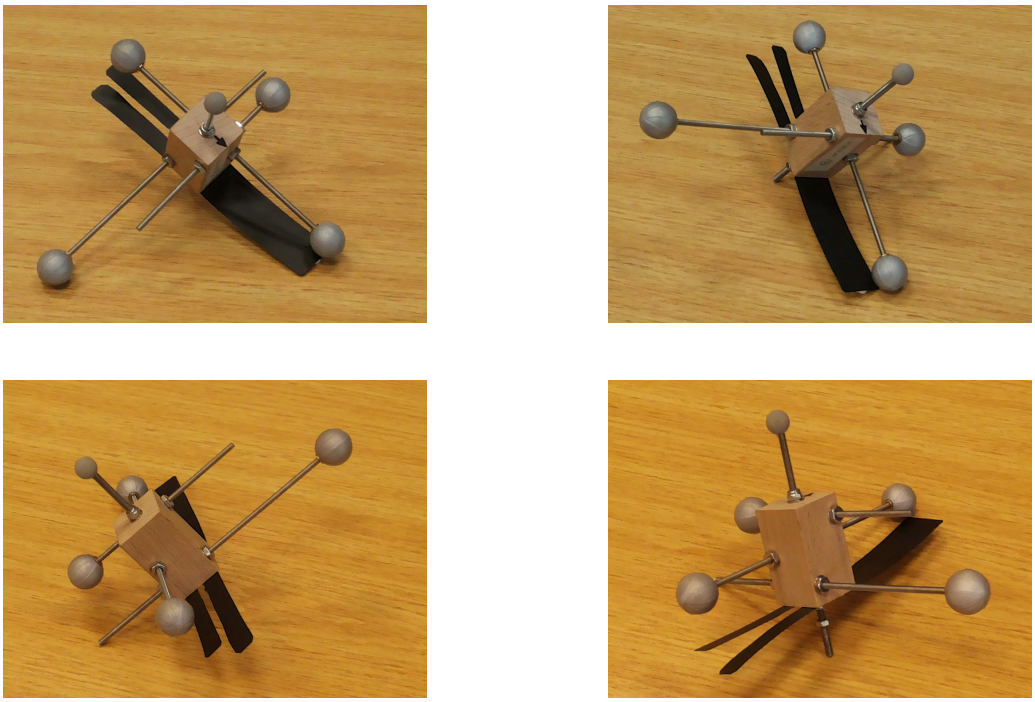


Figure 4.4: Four different views of one of the marker sets used at the 2014 RoCKIn Competition.

The marker sets had been prepared by RoCKIn before the Competition, and were given to the teams to be mounted on their robots before their entrance in the testbeds. Figure 4.4 shows one of these marker sets.

The maximum dimension of each marker set is about 15 cm. The spherical objects at the end of the rods are infrared reflective markers; relative positions of the markers have been chosen to help localization by the motion capture system. The threaded rod protruding from the bottom was used to engage a threaded hole in the robot. The black flexible sheets were used as a quickly-set and quickly-released system to stop the marker from rotating around the rod: removable putty was used, in fact, to temporarily glue one of these sheets to a part of the robot.

Well in advance of the starting date of the Competition, participating teams were provided with pictures, dimensions and a 3D CAD model of the marker set. All this information was also published on the RoCKIn Competition 2014 *wiki* (see Section 4.1). In particular, to prepare for the acquisition of robot pose data, teams were required to comply with the following setup instructions.

> The base of the robot must be provided with a vertical-axis M4 nut (or threaded hole) so that the Marker Set can be screwed onto it from above. For this, on the bottom of the Marker Set (the side facing the floor) there is a protruding M4 screw (visible in the first picture). The team can freely choose the location of the M4 nut (or threaded hole). High places and/or places without nearby objects that can hide the markers are preferred. M4 is a standard ISO metric thread with an outer diameter of 4 mm.
>
> Teams must provide the translation between the origin of the reference frame of the base of the robot and the center of the M4 nut. This translation is composed of the displacements along the X, Y and Z axes of such reference frame. Precisely, each team is required to provide a text file called displacement.txt containing four lines of text having the form
> X = dx
> Y = dy
> Z = dz
>
> where dx, dy and dz must be substituted with the numerical values of the displacements described above, expressed in meters. With this file, the team specifies that the center of the M4 nut is at coordinates (dx, dy, dz) according to the reference system of the base of the robot. Important! Teams who do not comply with the axis convention used by ROS [http://www.ros.org/reps/rep-0103.html] for axes (i.e. X pointing forward, Y pointing left, Z pointing up) must add a second text file, called axes_orientation.txt where they describe clearly, in free text, how their axes are oriented.
>
> Files displacement.txt and (if required) axes_orientation.txt must be written on the USB key provided by RoCKIn, in the same directory where the robot writes the data required by the benchmark: see this document for details. Please note that the files must be saved on the USB key each time the robot executes a benchmark, even if they are unchanged wrt preceding benchmarks.
>
> Mounting the Marker Set on the robot requires that there is a free volume of space (cylinder, diameter 180 mm) above the M4 nut. The Marker Set weighs 300 g approximately. The protruding screw on the bottom of the Marker Set is 25 mm long. The robot must avoid collisions between the Marker Set and other objects (when assessing collisions for scoring, the Marker Set is considered as a part of the robot).

All the teams participating to the RoCKIn Competition 2014 followed these instructions. Additionally, the supports for the marker set provided by some of the teams proved to be too flimsy or awkward for an easy and/or safe mounting; in these cases, the problem was solved with a bit of improvisation and the help of the RoCKIn staff. Figure 4.5 shows the marker set mounted on one of the RoCKIn@Work robots.

Figure 4.5: Marker set mounted on one of the robots participating to the 2014 RoCKIn Competition.

4.2.3 Robot logging setup

Beside physical setup for the mounting of the marker set, teams were required to log some data. While the RoCKIn Rulebooks specified what data had to be logged, the actual way to do the logging was defined by the following instructions provided to the teams.

> RoCKIn benchmarks require that the robot logs some of its own internal data. This document explains the modalities of this logging.
>
> ***WHAT DATA MUST BE SAVED The data that the robot must save are specified by the Competition wiki. They differ from benchmark to benchmark. Please note that some data streams (those with the highest bitrate) should 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 should be logged by the robot only in the time intervals when it is actually performing object recognition.
>
> ***WHAT WE DO WITH THE DATA These data are not used during the Competition. In particular, they are not used for scoring. The data are processed by RoCKIn after the end of the competition; they are used for in-depth analyses and/or to produce datasets to be published for the benefit of the robotics community.
>
> ***WHERE AND WHEN THE ROBOT MUST SAVE THE DATA Robots must save the data specified by the Competition wiki on a USB stick provided by RoCKIn. 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.
>
> ***DIRECTORY NAMES Each time a team's 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 Functionality Benchmarks) or
NameOfTheTeam_TBMx_DD_HH-MM (for Task Benchmarks)
where
x is the number of the Functionality or Task Benchmark
DD is the day of the month (28, 29 or 30 for the 2014 Competition)
HH, MM represent the time of the day (hours and minutes)

2. Configure the robot to save the data files in such directory.

All files produced by the robot that are associated to 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.

Additional information and guidelines necessary for specific benchmarks were provided, when needed, to the teams. For instance, the reference frames associated to the RoCKIn@Home and RoCKIn@Work testbeds, to be used to provide robot poses; or the format for semantic maps that the robot had to provide for TBM1@Home.

4.3 Results

The first and most important result of the RoCKIn Competition 2014 was its own overall success. This was not to be taken for granted, given the complexity of setting up a completely new robot *benchmarking competition* with significant novel elements (the whole benchmarking aspect).

The second result was the validation of the RoCKIn Benchmarking System, realized according to the indications of D2.1.7. Notwithstanding the practical difficulties (including multiple, disjointed and large areas to be covered by the motion capture systems; erratic lighting; strong presence of occlusions in RoCKIn@Home; large number of hardware and software components to coordinate) the System worked as expected and data collection was successful. This outcome confirmed the soundness of design of the System: for that reason, subsequent improvements (as described in this Deliverable) have been incremental instead of disruptive.

The last result of the RoCKIn Competition 2014 are the datasets. These include both ground truth data (originating from the motion capture system) and robot data (collected by the teams). The latter comprises raw sensor data (e.g., from laser scanners or cameras) and processed data (e.g., reconstructed robot pose). For example, for the TBM3@Home Catering for Granny Annie's Comfort, the following data were expected to be collected for each run of every team: the audio signals of the conversations between Annie and the robot (collected by the robot), the final commands produced after the natural language analysis process (collected by the robot), the ground truth pose of the robot while moving in the environment (collected using the motion capture system), the pose of the robot (as perceived by it) while moving in the environment, the sensorial data of the robot when recognizing the object to be operated, and the results of the robot's attempts to execute commands. For the FBM1@Home Object Perception, expected benchmarking data include: sensor data (images, point clouds, ...) used by the robot to perform classification; the class, instance, and pose of every object (as determined by the robot); the actual class, instance, and pose of every object (ground truth). For the FBM3@Home Speech

	Day 1	Day 2	Day 3
runs (TBMs and FBMs) with ...	23	31	22
complete data	10	22	20
incomplete data	2	1	1
no data	11	8	1

Table 4.1: Benchmarking data collected during the 2014 RoCKIn competition

Understanding, benchmarking data that were expected to be collected include: sensor data (audio files) used by the robot to perform speech recognition and the command (action and arguments) as recognized by the robot. Similar rich benchmarking data were expected to be collected for all other FBMs and TBMs.

For the participating teams, recording of sensor data and processed information is mandatory, although some flexibility has been allowed during the RoCKIn Competition 2014. Since it turned out that most teams were using ROS, we tried to limit the effort for onboard data collection by using the ROS rosbag recording tool (which can be used also by teams not using ROS, by exploiting the rosbag APIs). Recorded data depend on the hardware equipments of the robots; for example, data collected during the FBM1@Home Object Perception includes both images and images plus point clouds of the same objects, according to the different sensors mounted on different robots. In principle, also stereo images could be present.

The amount of benchmarking data that have been actually collected over all the runs of the TBMs and FBMs on the 3 days of the 2014 competition is summarized in Table4.1. It is evident the positive trend as the competition progressed, from 43% of runs (10 out of 23 runs) with complete benchmarking data on the first day, to 91% of runs (20 out of 22 runs) with complete benchmarking data on the last day, which has been a half-day competition. This is due to increased awareness of teams about data collection. Globally, 68% of runs (52 out of 76 runs) have complete benchmarking data. Incomplete benchmarking data are due to their incorrect format or to missing portions. Note that runs with no benchmarking data include also runs in which robots failed to start, which have been 4, 3, and 0, on the three days, respectively.

These benchmarking data are made available[3] to the research community, in order to ease the reproducibility of results and the comparison with the teams participating in the RoCKIn competitions. In particular, data relative to poses of robots as collected by the ground truth system can be used by teams to "replay" the runs of their robots, for example matching the actual pose of a robot with the expected one according to the robot perception. As some anecdotal evidence from the 2014 competition confirmed, this can have a positive feedback on fixing bugs and improving performance of teams. RoCKIn competitions play the role of tools for collecting a huge amount of data that can be later used for reproducing experiments and for benchmarking also by researchers not participating in the competitions, who can download the data sets and run their algorithms on them.

[3] http://thewiki.rockinrobotchallenge.eu/index.php?title=Datasets

Chapter 5

RoCKIn Camp 2015 (Peccioli, Italy)

This chapter has been written after the end of the 2015 RoCKIn Camp[1] (Peccioli, March 18-22, 2015). One of the objectives of Camp 2015 was to further advance the development of the RoCKIn Benchmarking System, after the successful test of the Competition 2014 and in view of the Competition 2015.

Benchmarking work at Camp 2015 focused on the only point of the benchmarking procedure where significant problems were detected at the Competition 2014: namely, collection of robot data by the teams (see also Subsection 2.3.6). Although at the RoCKIn Competition 2014 teams were fully aware of the need to collect data, and were provided with complete instructions explaining what to log and how, few of them actually committed the effort necessary to provide good-quality data. For this reason, Camp 2015 was used as a testing ground for the two-pronged strategy devised to solve problem. As explained in Subsection 2.3.6, this strategy is composed of two elements:

1. Providing teams with *support* in terms of instructions, software, templates, and direct human help. Additional tools to help teams understand what they are required to log and how to do the logging have been and will be prepared.

2. Actively pushing the teams towards good-quality data collection. At future Competitions this will be done by including data quality among the elements used for scoring. Of course, at the Camp this approach was not applicable, so the push was obtained by devoting RoCKIn personnel to provide the teams with full-time advice, suggestions, support and debugging help on data logging.

5.1 Setup

The 2015 RoCKIn Camp took place at the Service Robotics and Ambient Assisted Living Lab[2]. In particular, the environment included a "domotic home": an apartment equipped with several domotic devices, used as testing grounds for scientific experiments. RoCKIn did not use such devices, but benefited from the realistic home-like (but easily accessible for robots) environment by setting up the RoCKIn@Home testbed there. The RoCKIn@Work testbed was located in a nearby room in the same group of buildings.

The setup of the RoCKIn Benchmarking System used at the 2015 Camp was a subset of that used at the 2014 Competition, described in Chapter 4. For this reason, we will not

[1] http://rockinrobotchallenge.eu/camp2015.php
[2] http://www.echord.eu/facilities-rifs/the-peccioli-rif/

Figure 5.1: Left: two views of the motion capture system installed at the Camp 2015.

go again into setup details here. The only differences between the setup of the Peccioli Camp and that of the Toulouse Competition were the following:

- cameras were mounted on tripods instead of on a truss, which was feasible due to the much more controlled environment (no non- RoCKIn people roaming around without taking care not to move the tripods) and to the lower height (ceilings at the "domotic home" were 3 m high);
- we used only one OptiTrack motion capture rig, mounted around the RoCKIn@Home testbed, instead of three.

Figure 5.1 shows the motion capture setup at the 2015 Camp. Notwithstanding the use of only one motion capture system (instead of the three separate ones used at the Competition 2014), the different environment of Peccioli allowed to cover RoCKIn@Work Task Benchmarks and Functionality Benchmark 1 for both RoCKIn@Home and RoCKIn@Work with only one system instead of two. In fact, in Peccioli it was possible to use the RoCKIn@Home testbedarea also for Functionality Benchmark 1, as shown in Figure 5.2.

Motion capture coverage of the RoCKIn@Work testbed was avoided. In part this was due to logistic difficulties (the RoCKIn@Work room was quite far from the RoCKIn@Home area, thus staffing both would have been difficult), but mostly this was a decision taken to focus benchmarking efforts towards interaction with teams. In fact, the main issue outlined by the Competition 2014, as explained in Section 2.3.6, was that teams were not correctly implementing data collection and logging on board their robots. The benchmarking work at the Camp 2015 was mostly focused on this issue.

5.2 Results

The results of the RoCKIn Camp 2015 can be subdivided into three categories:

- support material (documentation and software) prepared to help teams to correctly log robot data;
- feedback from teams on such material, and fine-tuning of it based on interaction with teams;
- datasets collected by teams using the support material.

Figure 5.2: Example of a testbed reference frame for RoCKIn@Home. The z-axis points towards the reader.

Among these results, the most important are certainly the first two. They were the starting point for a work that went on also after its conclusion, and led to a new documentation for benchmarking of robot data that will be included into the Rulebooks for the RoCKIn Competition 2015. Section 5.3 is dedicated to this documentation.

For what concerns datasets collected at the RoCKIn Camp 2015, they were actually a useful byproduct of the work on supporting teams, in the sense that they were the proof that such work was producing an effect. The collected data were related to the ground truth poses in the @Home testbed during the whole duration of the Camp and we collected also several bags from the robots to test and verify the new specifications for internal data acquisition. One of the RoCKIn@Work teams distinguished for the development of a logging toolkit to be used by other teams and for that reason it was awarded during the Camp with a Benchmarking Award.

5.3 Support and documentation

Another key result of Camp 2015 was the new support material about robot data logging provided to the teams. This material included software templates and documentation; the latter will become part of the Rulebooks for the RoCKIn Competition 2015, while the software will be published to be used by teams participating to the Competition. The first version of the support material was published at the Camp. It comprised a set of software tools (still available at `https://labrococo.dis.uniroma1.it/svn/software-open/trunk/rococo-ros/rockin_logger`) and an extensive documentation about logging procedures. The work on this material went on after the end of the Camp, and its final result is superior to the corresponding documentation in previous versions of the Rulebooks. The following of this Section comprises a preliminary version of this documentation. At the time of writing, the Rulebooks for the RoCKIn Competition 2015 (i.e. Deliverables D2.1.3 and D2.1.6) are not yet finalized. Therefore, **the**

following part of this section should be considered only as an example, not as a reference. It is, in fact, an excerpt from a preliminary version of the Rulebooks.

During all task benchmarks, robots are required to log Internal Data according to the following specifications. This data must be expressed in the reference frame of the testbed which will be clearly marked on it. It will be possible for teams to define such frame in their robot before the start of the competition. Fig. 5.3 illustrates one possible position of the testbed reference frame.

Figure 5.3: Example of a testbed reference frame for RoCKIn@Home. The z-axis points towards the reader.

Only relevant data is expected to be logged (i.e. pointcloud used to recognize an object, more than one if an algorithm requiring multiple pointclouds is used). There are no restriction about the framerate: data can be saved, for the relevant parts of the benchmark, at the rate they are acquired or produced. The log may be a rosbag or the corresponding YAML representation, as specified in [*another part of the Rulebook*], here we refer to the rosbag version, the corresponding YAML translation should be direct.

Data always logged

The list of topics to be logged (i.e., for all Tasks and Functional Benchmarks) is reported in the following table

Topic	Type	Frame Id	Notes
/rockin/robot_pose[3]	geometry_msgs/PoseStamped	/map	10 Hz
/rockin/marker_pose[4]	geometry_msgs/PoseStamped	/map	10 Hz
/rockin/trajectory[5]	nav_msgs/Path	/map	Each (re)plan
/rockin/<device>/image[6]	sensor_msgs/Image	/<device>_frame	–
/rockin/<device>/camera_info[7]	sensor_msgs/CameraInfo	–	–
/rockin/depth_<id>/pointcloud[8]	sensor_msgs/PointCloud2	/depth_<id>_frame	–
/rockin/scan_<id>[9]	sensor_msgs/LaserScan	/laser_<id>_frame	10-40Hz
tf[10]	tf	–	–

The format for the name of the bag file to be saved by the Teams on their robot is the following:

```
{F|T}BM{H|W}{1|2|3}_YYYYMMDDhhmm_{teamname}.bag
```

e.g., FBMH1_201503041356_myteam.bag, TBMH3_201503041156_myteam.bag, etc.

Beside the data in the table, additional data the robot must save is specified in the particular benchmark subsection. Please note that some data streams (those with the highest bitrate) should 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 should be logged by the robot only in the time intervals when it is actually performing object recognition.

This data is not used during the competition. In particular, they are not used for scoring. The data are processed by RoCKIn after the end of the competition for in-depth analyses and/or to produce datasets to be published for the benefit of the robotics community.

Robots must save the data, as specified in the particular benchmark subsection, on a USB stick provided by RoCKIn. 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.

NOTE: while the content of the data files saved by the robot is not used for scoring, the existence of such files and their compliance to the specifications does influence the score of the robot. Teams have the responsibility of ensuring that the required data files are saved, and of delivering them to the referee at the end of the benchmark. These aspects will be noted on the score sheet and considered for team ranking.

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

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

[5] Trajectories planned by the robot, referred to the robot base, including when replanning.

[6] 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.

[7] Calibration info for /rockin/<device>/image.

[8] 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.

[9] 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.

[10] 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.

Data logged in specific benchmarks

In addition to the benchmark data to be always logged, Task and Functional Benchmarks require the logging of additional, specific, data. In the following we report the list of specific data to be logged as defined in the rulebooks delivered as D-2.1.3 and D-2.1.6. Since the data to be logged may vary from benchmark to benchmark please refer to the final document that will be delivered before the competition for the complete list of topics to be logged.

In some Task Benchmarks the audio needs to be logged, to dump directly the audio into the bag file the shortest path is to add the following to your launch file

```
<node name="audio_capture" pkg="audio_capture" type="audio_capture">
  <param name="bitrate" value="128"/>
  <remap from ="audio" to ="/rockin/audio"/>
</node>
```

to reduce the framerate copying from a topic to another you can use the throttle command from the topic_tools (specifying the frequency)

```
<node name="rockin_log_image" pkg="topic_tools" type="throttle"
  args="messages␣/you/original/images/topic␣1.0␣/rockin/image"/>
<node name="rockin_log_pointcloud" pkg="topic_tools" type="throttle"
  args="messages␣/you/original/pointcloud/topic␣0.2␣/rockin/pointcloud"/>
```

in case you want to switch on and off the logging of the audio you can use the mux tool of the topic_tools (please refer to the mux documentation on how to use it)

```
<node name="audio_capture" pkg="audio_capture" type="audio_capture">
  <param name="bitrate" value="128"/>
</node>
<node name="audio_mux" pkg="topic_tools" type="mux" args="/rockin/audio␣audio"/>
```

RoCKIn @Home TBM1: *Getting to know my home*

During this Task Benchmark only the Internal Data described in Section 5.3 will be collected.

RoCKIn @Home TBM2: *Welcoming visitors*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table NOTE:

Topic	Type	Frame Id	Notes
/rockin/command[11]	std_msgs/String	–	–
/rockin/visitor[12]	std_msgs/String	–	–
/rockin/audio[13]	audio_common_msgs/AudioData	–	–
/rockin/notification[14]	std_msgs/String	–	–

[11] The event or command causing the activation of the robot.

[12] The result of any attempt by the robot to detect and classify a visitor

[13] The audio signals of the conversation with the visitors. 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.

[14] Any notifications from the robot (e.g., alarm if a visitor shows anomalous behavior)

the images and pointclouds in the Internal Data should contain the sensorial data used to recognize the visitor.

RoCKIn @Home TBM3: *Catering for Granny Annie's Comfort*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table NOTE:

Topic	Type	Frame Id	Notes
/rockin/command[15]	std_msgs/String	–	–
/rockin/audio[16]	audio_common_msgs/AudioData	–	–

the images and pointclouds in the Internal Data should contain the object to be operated.

RoCKIn @Home FBM1: *Object Perception Functionality*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/notification[17]	std_msgs/String	–	–

NOTE: the images and pointclouds in the Internal Data should contain the sensorial data used to recognize each of the presented object, thus we expect to have (at least) 10 images (if recognition uses a camera), (at least) 10 pointclouds (if recognition uses a depth sensor), and 10 notification strings.

RoCKIn @Home FBM2: *Navigation Functionality*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/robot_pose_waypoint[18]	geometry_msgs/PoseStamped	/map	when reached
/rockin/marker_pose_waypoint[19]	geometry_msgs/PoseStamped	/map	when reached

RoCKIn @Home FBM3: *Speech Understanding Functionality*

During the execution of the benchmark, the following data will be collected:

[15]The command produced by the natural language analysis process.

[16]The audio of the conversation between Annie and the robot. 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.

[17]Tab delimited string with notification of the perceived object: CLASS OBJECT_ID X Y THETA

[18]The 2D robot pose, once each waypoint is reached, at the floor level, i.e., $z = 0$ and only yaw rotation.

[19]The 3D pose of the marker in 6 degrees of freedom once each waypoint is reached.

- Sensor data (in the form of audio files) used by the robot to perform speech recognition, 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 transcription 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.

Formats and interfaces for the transmission of internal robot data will be provided to the teams well before the Competitions. Please note that, according to the procedure described by Section [*another part of the Rulebook*] and to the definitions of 'offline' and 'online' used for the other benchmarks[20], all data acquisition occurs offline.

RoCKIn@Work TBM1: *Prepare Assembly Aid Tray for Force Fitting*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/qrcode[21]	std_msgs/Int32	–	when recognized

NOTE: the images and pointclouds in the Internal Data should contain the sensorial data used to recognize the QR code.

RoCKIn@Work TBM2: *Plate Drilling*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/dril_command[22]	std_msgs/Int32	–	when issued
/rockin/qcc_command[23]	std_msgs/Int32	–	when issued
/rockin/plate_condition[24]	std_msgs/Int32	–	when issued

NOTE: the images and pointclouds in the Internal Data should contain the sensorial data used to recognize the status of the plates.

[20] 'Offline' identifies data produced by the robot that are collected by the referees when the execution of the benchmark ends; 'online' identifies data that the robot has to transmit to the testbed during the execution of the benchmark. NOTE: the online data should also be displayed by the robot on its computer screen, for redundancy purposes, in case problems with wireless communications arise.

[21] ID of the assembly aid tray or container, deteted by the robot by analyzing the QR code.

[22] Drilling commands issued by the robot

[23] QCC commands issued by the robot

[24] Condition of each plate, as evaluated by the robot, after drilling

RoCKIn@Work TBM3: *Fill a Box with Parts for Manual Assembly*

During the execution of the benchmark the Internal Data defined in Section5.3 will be collected.

RoCKIn@Work FBM1: *Object Perception Functionality*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/notification[25]	std_msgs/String	–	–

NOTE: the images and pointclouds in the Internal Data should contain the sensorial data used to recognize each of the presented object, thus we expect to have (at least) 10 images (if recognition uses a camera), (at least) 10 pointclouds (if recognition uses a depth sensor), and 10 notification strings.

RoCKIn@Work FBM2: *Manipulation Functionality*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/grasping_pose[26]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/gripper_pose[27]	geometry_msgs/PoseStamped	/base_link	10 Hz
/rockin/arm_joints[28]	geometry_msgs/JointState	/base_link	10 Hz

NOTE: the images and pointclouds in the Internal Data should contain the sensorial data used to recognize the position of the object with respect to the robot.

RoCKIn@Work FBM3: *Control Functionality*

During the execution of the benchmark, the Internal Data defined in Section5.3 will be collected together with the additional information described in the following table

Topic	Type	Frame Id	Notes
/rockin/reference_pose[29]	geometry_msgs/PoseStamped	/base_link	–
/rockin/starting_pose[30]	geometry_msgs/PoseStamped	/base_link	When starting
/rockin/ending_pose[31]	geometry_msgs/PoseStamped	/base_link	When ending
/rockin/gripper_pose[32]	geometry_msgs/PoseStamped	/base_link	10Hz
/rockin/arm_joints[33]	geometry_msgs/JointState	/base_link	10 Hz

[25] Tab delimited string with notification of the perceived object: CLASS OBJECT_ID X Y THETA
[26] Pose of the grasping position on the object.
[27] Pose of the gripper.
[28] Joints data
[29] Pose of the gripper at the reference point.
[30] Pose of the gripper at the starting point.
[31] Pose of the gripper at the end of the trajectory.
[32] Pose of the gripper during the whole trajectory.
[33] Joints data

Appendix A

The RoCKIn Benchmarking System: technical details

A.1 Introduction

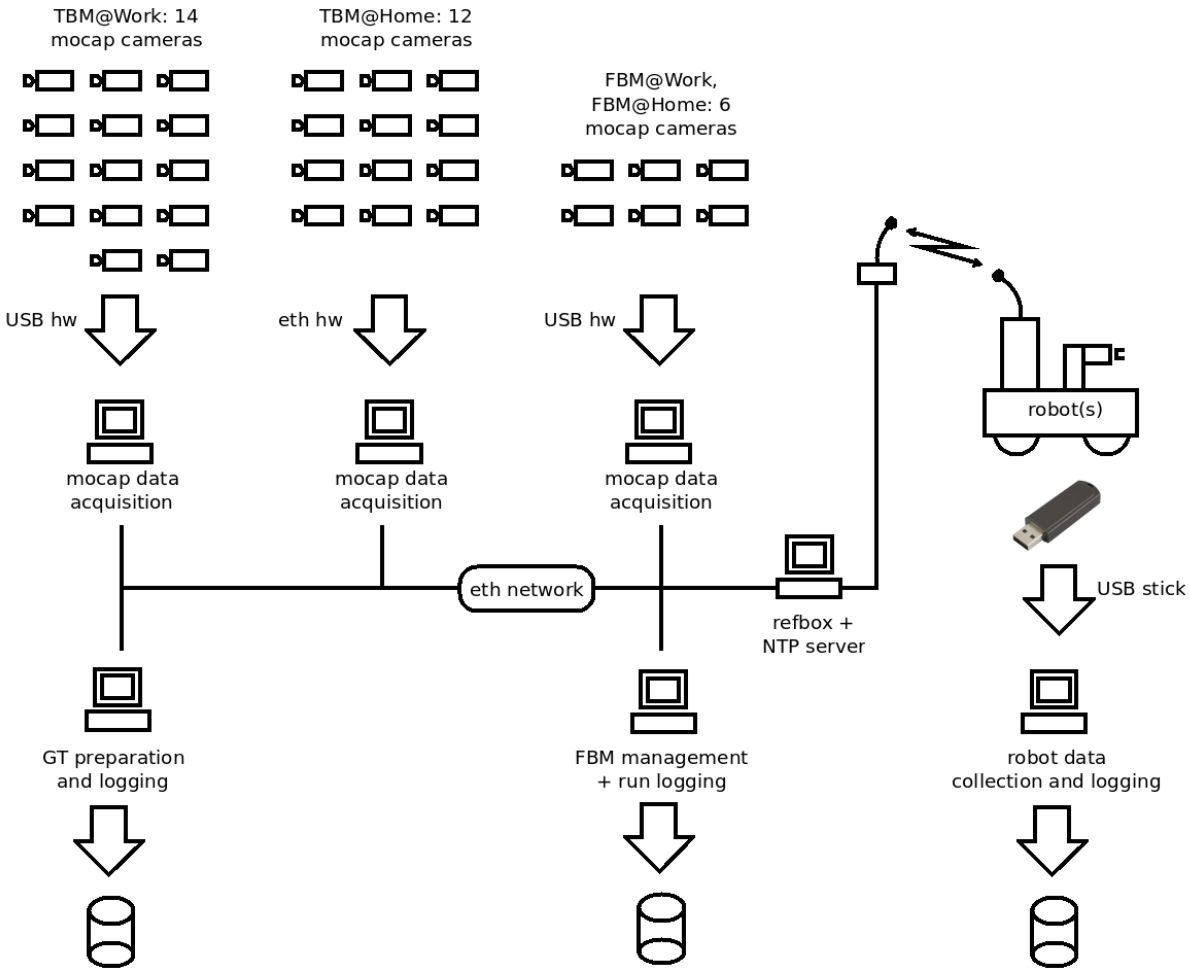
This Appendix describes the components of the RoCKIn Benchmarking System in the configuration used during the RoCKIn 2014 Competition held in Tolouse in November 2014, the only one which took place before the time when Deliverable D2.1.8 is finalized. Beside the components dedicated to ground truth data collection, the RoCKIn Benchmarking System includes other elements dedicated to data processing and interaction with the Competition infrastructure. The latter, while not directly involved in the scope of this Deliverable, may be interesting for the reader to better understand how data collection interacts with the other parts of the RoCKIn Benchmarking System.
This Appendix is organized as follows. Section A.2 gives an overview of the system architecture, with details about the involved subsystems and the network configuration. Section A.3 describes the motion capture acquisition and logging system, detailing the software architecture, the configuration for the different testbeds, and instructions to start the system. Section A.4 describes the setup for the functional benchmarks that require interactions with the motion capture system, detailing the interaction with other systems used during the benchmarks and the workflow of a generic benchmark. Section A.5 describes benchmarks FBM1@Home and FBM1@Work, which were the most structured at the 2014 RoCKIn Competition.

A.2 Benchmarking System

System architecture

The motion capture and benchmarking system involves several subsystems. The network is split into two subnets: the benchmarking systems are connected to the 10.0.0.0./24 subnet, while the Referee Box and other appliances not related to benchmarking are on the 192.168.1.0/24 subnet. A router acts as a gateway between the 10.0.0.0/24 subnet (connected to one of the "LAN" ports) and the 192.168.1.0/24 network (connected to the "WAN" port).

For motion capture, three Optitrack systems are installed in three distinct testbeds: FBM@Home/@Work (Shuttle PC #1), TBM@Home (Dell Quad-core), TBM@Work (Shut-



tle PC #2). Each system has a Windows PC to run the Optitrack Motive software [1], which is required to interface with the cameras and track the marker sets. Processed data is streamed over UDP to the clients.

For ground truth (GT) logging, a Linux PC running ROS is used (HP laptop). A ROS node receives the UDP packets broadcasted by Motive and publishes, for each tracked marker set, the corresponding *tf* frame, the 3D pose, and the 2D pose. The GT is logged by a set of other ROS nodes, as described in Section A.3.

To manage the Functional Benchmarks, a Linux PC running ROS is used (Zotac). This PC receives data from the motion capture too, when needed by the benchmark, and runs a set of ROS nodes to interact with the Referee Box, manage the execution of the benchmarks, and evaluate the final score, as described in Section A.5.

Clock synchronization is provided by NTP. The GT logging PC and the FBM PC run a NTP client (chrony [2]), configured to use the NTP server running on the @Home refbox (IP address 192.168.1.1). Listing 1 shows the configuration to use on NTP clients to sync with the server.

An additional PC is used to save all the data logged by the robots and collected on USB sticks (Fontana laptop).

[1] http://www.optitrack.com/products/motive/
[2] http://chrony.tuxfamily.org/

```
server 192.168.1.1 minpoll 2 maxpoll 4
initstepslew 2 192.168.1.1

keyfile /etc/chrony/chrony.keys
commandkey 1
driftfile /var/lib/chrony/chrony.drift
maxupdateskew 5
dumponexit
dumpdir /var/lib/chrony
pidfile /var/run/chronyd.pid
logchange 0.5
rtcfile /etc/chrony.rtc
rtconutc
rtcdevice /dev/rtc
sched_priority 1

local stratum 10
allow 127.0.0.1/8

# log measurements statistics tracking rtc
# logdir /var/log/chrony
```

Listing A.1: chrony.conf

Network configuration

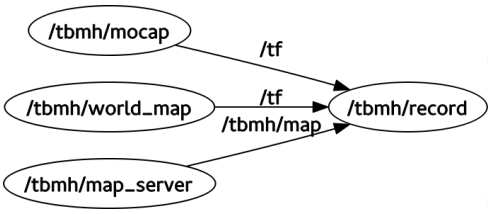
The motion capture system is configured to use the network 10.0.0.0/24. All the PC have static IP addresses, and a router has been used to provide connection to the Internet. As the Functional Benchmarks need to interact with the Referee Box, the FBM management PC is configured to use two IP addresses: 10.0.0.14 to communicate with the motion capture system, and 192.168.1.2 to communicate with the refbox. For this reason, the FBM management box, which needs to communicate with both the Optitrack System and the Referee Box, has two IP addresses configured (by defining an alias for eth0). The following table summarizes the network configuration of the RoCKIn Benchmarking System.

PC	IP address	Netmask	Gateway
Mocap FBM PC	10.0.0.11	255.255.255.0	10.0.0.254
Mocap TBM@Home PC	10.0.0.12	255.255.255.0	10.0.0.254
Mocap TBM@Work PC	10.0.0.13	255.255.255.0	10.0.0.254
FBM management PC	10.0.0.14	255.255.255.0	10.0.0.254
FBM management (alias)	192.168.1.2	255.255.255.0	
GT logging PC	10.0.0.15	255.255.255.0	10.0.0.254
Router PC	10.0.0.254	255.255.255.0	DHCP
Refbox and NTP server	192.168.1.1		

A.3 Motion capture data acquisition and logging

ROS nodes

The following ROS nodes run on the GT logging computer, which runs several instances of *roscore*, one for each motion capture setup. Each instance binds roscore to a different TCP port, to force separation between the different setups. The different environments, with the corresponding roscore instances and TCP ports, are handled by the environment scripts (see Section A.3).



map_server

This node is a ROS *map_server*[3] node, a C++ node provided by the ROS *map_server* package. It is used to publish a map of the enviornment, which is described by a bitmap file. In particular, we use this node to stream a map of the testbed, used for inspection with RViz and other ROS tools. The node is started by the ROS launch file for the particular benchmark. The map image name and the corresponding parameters are declared in YAML configuration files, one for each testbed, named and stored in the *rockin_benchmarking/config* folder.

world_map

This node is a ROS *static_transform_publisher*[4] node, a C++ node provided by the ROS *tf* package. It is used to stream a static transformation between two reference frames. In particular, we use this node to stream the transformation between the absolute reference frame (world) and the map reference frame (map). The node is started by the ROS launch file for the particular benchmark. The transformation between the world reference frame and the map reference frame is fixed to $(0, 0, 0, 0, 0, 0)$ (i.e., null translation and rotation are applied), and the map is located in the correct position by using the origin parameter in the map configuration YAML file.

mocap

This node is a ROS *mocap_optitrack*[5] node, a C++ node provided by the ROS *mocap_optitrack* package. It is used to translate motion capture data from an OptiTrack rig to tf transforms, poses and 2D poses. The node receives packets that are streamed by the Optitrack Motive[6] software, decodes them and broadcasts the poses of configured rigid bodies as tf transforms, poses, and/or 2D poses. The node is started by the ROS launch file

[3] http://wiki.ros.org/map_server#map_server-1
[4] http://wiki.ros.org/tf#static_transform_publisher
[5] http://wiki.ros.org/mocap_optitrack
[6] http://www.optitrack.com/products/motive/

for the particular benchmark. The rigid body IDs to be tracked are listed in the YAML configuration files, one for each testbed, named *{tbmh,tbmw,fbm1,fbm3w}_mocap.yaml* and stored in the *rockin_benchmarking/config* folder.

record

This node is a a ROS *record* node, a C++ node provided by the ROS *rosbag*[7] package. It is used to record data from a running ROS system into a set of .bag files, which are splitted every hour (the split period is specified in the launch files). The node is started by the ROS launch file for the particular benchmark. The list of recored topic depends on the particular benchmark, and it is declared in the ROS launch file. Data is logged using NTP time, so that offline association between GT logs and Teams logs is straightforward.

Configuration

To configure the mocap acquisition system for a particular testbed, the following steps are required.

- Calibrate the Optitrack system and define a convenient ground plane (i.e., the origin should be in a spot the robots can easy reach, and visible by many cameras).
- Create the rigid bodies to be tracked in the Optitrack Motive software, and assign them unique IDs.
- Edit the *XXX_mocap.yaml* file to match the rigid body IDs to the corresponding ROS frames and topics.
- Edit the *XXX_map.yaml* file to configure the map scale and transformation from the *world frame* (i.e., the origin defined in Motive) to the *map frame*.
- Launch the logging system by executing
  roslaunch rockin_benchmarking log_mocap_XXX.launch

[7] http://wiki.ros.org/rosbag

TBMH configuration

The TBMH configuration is intended to log the RoCKIn@Home Task Benchmarks. It acquires two rigid bodies: the robot marker set (Motive ID: 3) and the human marker set (Motive ID: 4). For each rigid body, the corresponding 3D pose, 2D pose, and tf frame are published to ROS, as specified in the *tbmh_mocap.yaml* configuration file.

```
rigid_bodies:
  '3':
    pose: robot/pose
    pose2d: robot/pose2d
    child_frame_id: robot_at_home
    parent_frame_id: world

  '4':
    pose: human/pose
    pose2d: human/pose2d
    child_frame_id: human_at_home
    parent_frame_id: world
```

Listing A.2: tbmh_mocap.yaml

The map image file is *tbmh_map.pgm* and its configuration is *tbmh_map.yaml*, both stored in the *rockin_benchmarking/config* folder.

```
# Map image file
image: tbmh_map.pgm

# Resolution of the map, meters / pixel
resolution: 0.01

# The 2-D pose of the lower-left pixel in the map
origin: [-5.02, -4.02, 0]

# Negate white/black free/occupied semantics?
negate: 0

# Occupied/free pixel thresholds
occupied_thresh: 0.65
free_thresh: 0.196
```

Listing A.3: tbmh_map.yaml

TBMW configuration

The TBMH configuration is intended to log the RoCKIn@Work Task Benchmarks. It acquires one rigid body, which corresponds to the robot marker set (Motive ID: 5). The corresponding 3D pose, 2D pose, and tf frame are published to ROS, as specified in the *tbmw_mocap.yaml* configuration file.

```
rigid_bodies:
  '5':
    pose: robot/pose
    pose2d: robot/pose2d
    frame_id: robot_at_work
```

Listing A.4: tbmw_mocap.yaml

The map image file is *tbmw_map.pgm* and its configuration is *tbmw_map.yaml*, both stored in the *rockin_benchmarking/config* folder.

```
# Map image file
image: tbmw_map.pgm

# Resolution of the map, meters / pixel
resolution: 0.01

# The 2-D pose of the lower-left pixel in the map
origin: [-3.48, -1.34, 0]

# Negate white/black free/occupied semantics?
negate: 0

# Occupied/free pixel thresholds
occupied_thresh: 0.65
free_thresh: 0.196
```

Listing A.5: tbmw_map.yaml

FBM1 configuration

The FBM1 configuration is intended to log the Functional Benchmark 1 (Object perception) for both RoCKIn@Home and RoCKIn@Work. It acquires two rigid bodies: the table origin marker set (Motive ID: 1) and the reference board marker set (Motive ID: 2). For each rigid body, the corresponding 3D pose, 2D pose, and tf frame are published to ROS, as specified in the *fbm1_mocap.yaml* configuration file.

```
rigid_bodies:
    '1':
        pose: origin/pose
        pose2d: origin/pose2d
        child_frame_id: origin
        parent_frame_id: world
    '2':
        pose: ref_board/pose
        pose2d: ref_board/pose2d
        child_frame_id: ref_board
        parent_frame_id: world
```

Listing A.6: fbm1_mocap.yaml

The map image file is *fbm1_map.pgm* and its configuration is *fbm1_map.yaml*, both stored in the *rockin_benchmarking/config* folder.

```
# Map image file
image: fbm1_map.pgm

# Resolution of the map, meters / pixel
resolution: 0.001

# The 2-D pose of the lower-left pixel in the map
origin: [-0.1, -0.1, 0]

# Negate white/black free/occupied semantics?
negate: 0

# Occupied/free pixel thresholds
occupied_thresh: 0.65
free_thresh: 0.196
```

Listing A.7: fbm1_map.yaml

Tools

Switching environments

For an easier switch between the different environment setups (e.g., roscore port, IP addresses, etc), a few bash scripts are available in the *utils/* folder. Each environment adds a prefix to the bash shell, to quickly recognize the current one. In this way, multiple benchmarking systems can run on the same machine, by opening separate terminal windows and configuring them for the corresponding environments.

Sourcing *utils/setup.bash* or adding it to the *.bashrc* file adds the following commands:

- `tbmh_env` adds the `TBM@HOME` prefix to bash and exports the following environmental variables:
  `ROCKIN_ENV="TBMH"`
  `ROS_MASTER_URI="http://localhost:11312"`
- `tbmw_env` adds the `TBM@WORK` prefix to bash and exports the following environmental variables:
  `ROCKIN_ENV="TBMW"`
  `ROS_MASTER_URI="http://localhost:11313"`

Log monitor

To be sure that log files are growing correctly, i.e., the motion capture systems and the ROS nodes are running without problems, a bash script monitors the log directory, warning the user if something is not as expected by printing error messages on a console and emitting a loud sound. The script is called *log_ monitor* and is available in the *utils/* folder. To run it, the correct environmental variables has to be set, so use one of the helpers from the previous section before running it:

```
$ tbmh_env
TBM@HOME $ log_monitor
```

Execution

To start a logger, first of all switch to the correct environment:

```
$ tbmh_env
```

Then, launch the logging system by calling *roslaunch*:

```
TBM@HOME $ roslaunch rockin_mocap tbmh_log_mocap.launch
```

In another terminal, launch the *log_ monitor* script:

```
$ tbmh_env
TBM@HOME $ log_monitor
```


A.4 Functional benchmarks

Some of the RoCKIn Functional Benchmarks depends on the motion capture system to be executed and to compute scores. In particular, the Benchmarking System is involved in the execution of the following FBMs:

- FBM1@Home (Object Perception)
- FBM1@Work (Object Perception)
- FBM3@Work (Path Following - introduced for the 2015 competition)

For these benchmarks, the Benchmarking System has to interact with the Referee Box while the benchmark is executed. Data exchange with the Robots is handled by the Referee Box, which is in charge of forwarding messages from and to the Benchmarking System (e.g., to send goals and to receive results).

Communication with the Referee Box

The FBM management PC, which runs the Benchmarking Box, and the @Home Referee Box (the RSBB Core) communicate through ROS middleware, by publishing and receiving messages on ROS Topics. As the RSBB relies on protobuf to communicate with the robots, the RSBB developers asked to avoid RPC-style interactions and use only the publish/subscribe messaging paradigm. A ROS Proxy module inside the RSBB Core takes care of converting data from ROS to protobuf and vice versa. A single instance of *roscore* is executed on the computer running the RSBB Core, and all other systems connect to this instance of *roscore*.

The Benchmark State describes the current state of the benchmark. It is constantly published on the ROS Topic *rockin_benchmark_name/state*, by sending *rockin_benchmarking/BenchmarkState.msg* messages, at a frequency sufficient to prevent dangerous situations (e.g., the robot not stopping in time).

Benchmark States

The overall state of a benchmark is descriped by 3 different states:

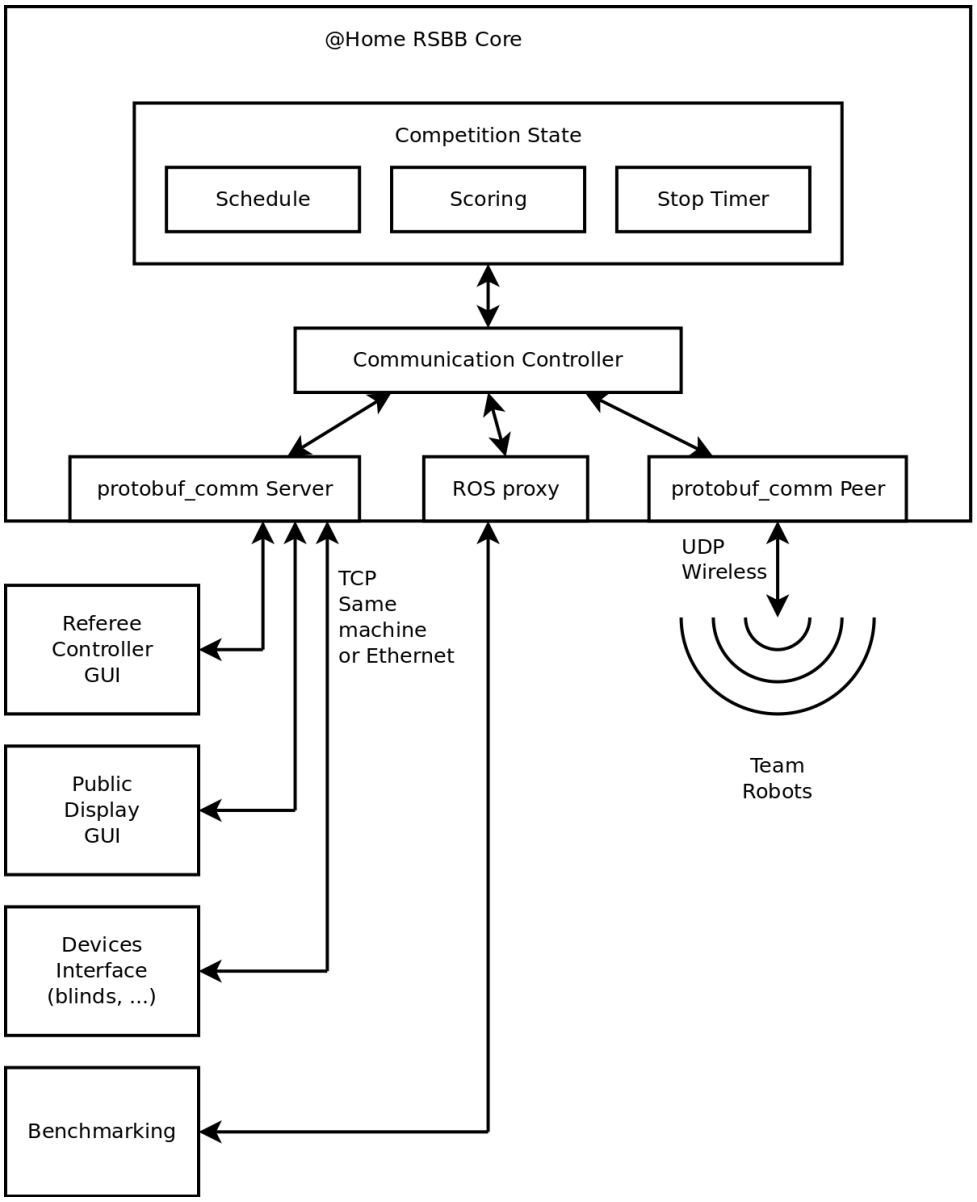
- the Benchmarking Box state
- the Referee Box state
- the Robot state

State transitions for a finite state machine are driven by state updates of the other state machines.

Benchmarking Box States

The Benchmarking Box runs a finite states machine with the following states:

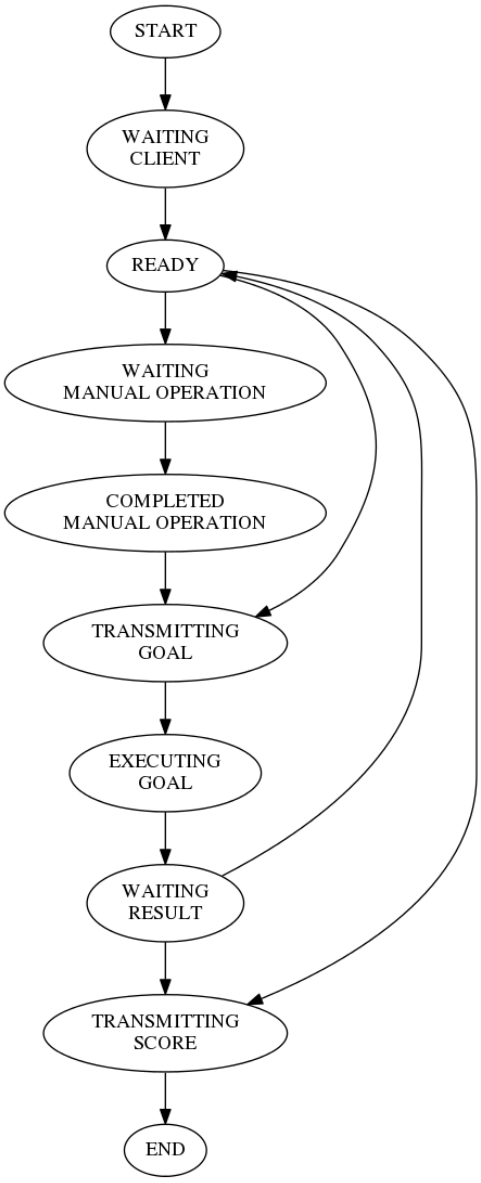
- ***WAITING CLIENT***: waiting for a connection from the Referee Box
- ***READY***: everything ready to start the benchmark



- ***WAITING MANUAL OPERATION***: a manual operation from the referee is needed
  payload contains the requested operation
- ***COMPLETED MANUAL OPERATION***: the referee confirmed that the manual operation was completed
- ***TRANSMITTING GOAL***: a new goal is beeing transmitted by the Benchmarking Box to the robot
  payload: the description of the goal
- ***EXECUTING GOAL***: waiting for the robot to complete the goal
- ***WAITING RESULT***: waiting for the robot to transmit the result to the Benchmarking Box
- ***TRANSMITTING SCORE***: transmtting the final score
  payload: the computed score

- ***END***: the benchmark is concluded

The current state is published on the *fbm_name/bmbox_state* topic, while message content is described by *BmBox.msg* (see Listing 8).

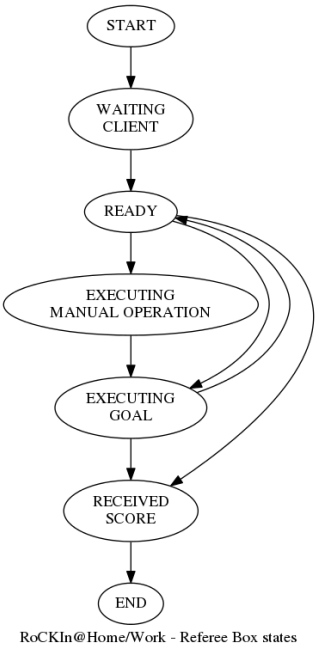


RoCKIn@Home/Work - Benchmarking Box states

```
uint8 START = 0
uint8 WAITING_CLIENT = 1
uint8 READY = 2
uint8 WAITING_MANUAL_OPERATION = 3
uint8 COMPLETED_MANUAL_OPERATION = 4
uint8 TRANSMITTING_GOAL = 5
uint8 EXECUTING_GOAL = 6
uint8 WAITING_RESULT = 7
uint8 TRANSMITTING_SCORE = 8
uint8 END = 9

uint8 state
string payload
```

Listing A.8: BmBoxState.msg



RoCKIn@Home/Work - Referee Box states

Referee Box States

The Referee Box runs a finite states machine with the following states:

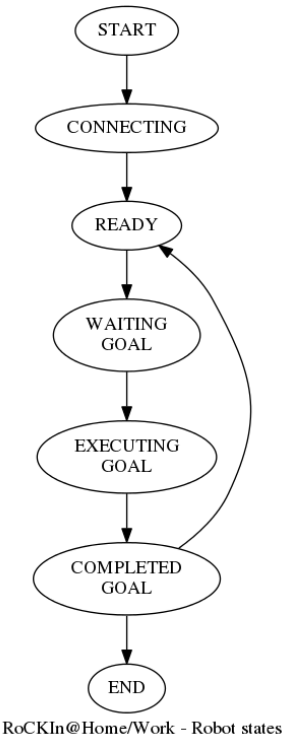
- ***WAITING CLIENT***: waiting for a connection from the Robot
- ***READY***: everything ready to start the benchmark
- ***EXECUTING MANUAL OPERATION***: the manual operation is beeing executed
- ***EXECUTING GOAL***: the robot is executing the goal
- ***RECEIVED SCORE***: received the final score from the robot
- ***END***: benchmark concluded
  payload: if the state transition has been issued by the referee box, payload specifies the reason (timeout / emergency halt)

The current state is published on the *fbm_name/refbox_state* topic, while message content is described by *RefBoxState.msg*.

```
uint8 START = 0
uint8 WAITING_CLIENT = 1
uint8 READY = 2
uint8 EXECUTING_MANUAL_OPERATION = 3
uint8 EXECUTING_GOAL = 4
uint8 RECEIVED_SCORE = 5
uint8 END = 6

uint8 state
string payload
```

Listing A.9: RefBoxState.msg



Robot States

The Robot runs a finite states machine with the following states:

- ***CONNECTING***: attempting to connect to the Referee Box
- ***READY***: everything ready to start the benchmark
- ***WAITING GOAL***: waiting for a goal from the Benchmarking Box
- ***EXECUTING GOAL***: executing the goal
- ***COMPLETED GOAL***: goal completed; the payload contains the result

The current state is published on the *fbm_name/client_state* topic, while message content is described by *ClientState.msg*.

```
uint8 START = 0
uint8 CONNECTING = 1
uint8 READY = 2
uint8 WAITING_GOAL = 3
uint8 EXECUTING_GOAL = 4
uint8 COMPLETED_GOAL = 5
uint8 END = 6

uint8 state
string payload
```

Listing A.10: ClientState.msg

Generic Functionality Benchmark Workflow

As soon as the benchmark is started (i.e., by launching the corresponding roslaunch file), the Benchmarking Box goes into *WAITING CLIENT* state. When a client connects, the Referee Box authenticates it and checks that clocks are synced; if everything is correct, the Referee Box state updates to *READY*. When the Referee Box state becomes *READY*, bot the Robot state and the Benchmarking Box state are updated to *READY* too.

The robot then can ask for a goal, and its state is updated to *WAITING GOAL*.

If a manual operation is requested (e.g., the referee must put an object in front of the robot), the Benchmarking Box updates its state to *WAITING MANUAL OPERATION*, sending the required operation as payload of the state message. When the manual operation has been concluded, the Referee Box updates its state to *EXECUTING GOAL*, and the Benchmarking Box goes into *COMPLETED MANUAL OPERATION* state.

The Benchmarking Box can now send the goal to the client, going to state *SENDING GOAL* and transmitting the description of the goal as payload of the state message. When the Robot receives the goal, it updates its state to *EXECUTING GOAL*, and the Benchmarking Box goes into state *WAITING FOR RESULT* as a consequence.

When the robot completed the goal, it updates its state to *TRANSMITTING RESULT*, with the result as payload of the state message; the Benchmarking Box saves the received results and, if there are more goals, waits for the robot requesting a new goal (i.e., the state is updated to *READY*), otherwise, the state is updated to *TRANSMITTING SCORE*, with the final score as payload, and the benchmark is concluded.

At any time, the state can be updated to *END* by the Referee Box, with payload "timeout" if it runs out of time, or payload "halt" if the benchmark was halted by a human request. The RSBB Core runs a stop timer, and provides a way to halt the benchmark by a human.

A.5 RoCKIn@Home and RoCKIn@Work Functional Benchmark 1

In this benchmark, the Robot is required to identify the class, the instance, and the pose of a set of objects. The benchmark works as follows (see the Rule Book for further details):

1. an object of unknown class and unknown instance is placed on a table in front of the robot
2. the robot must determine the object's class, its instance within that class as well as the 2D pose of the object w.r.t. the reference system specified on the table
3. the preceding steps are repeated until time runs out or 10 objects have been processed

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

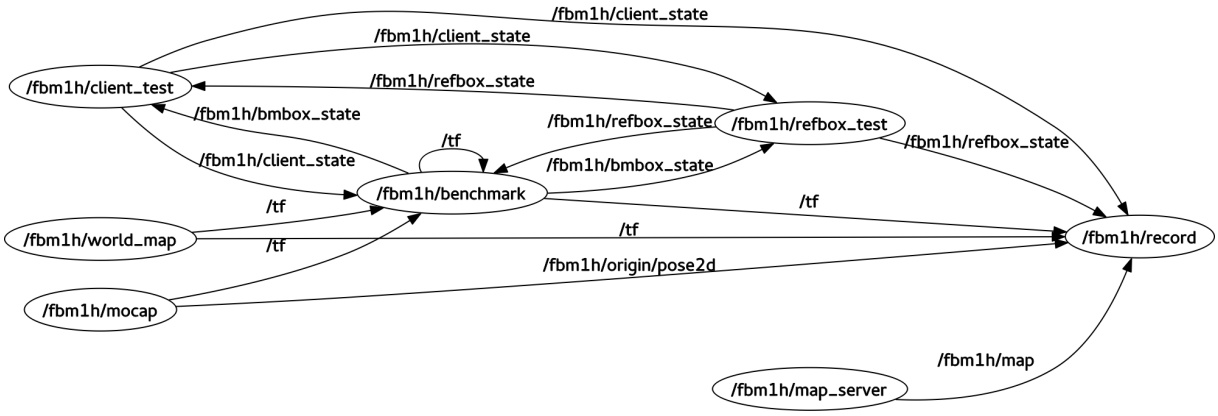
- object class name [string]
- object instance name [string]
- object pose (x [m], y [m], theta [rad])

The motion capture system is used to acquire the actual pose of the object. For this purpose, two marker sets are defined:

- one fixed to the table, which specifies the origin
- one fixed to a reference board, which has a slot to accomodate the objects

In this way, given the transformations between each object and the reference board marker set, we can acquire the pose of the object with respect to the origin.

ROS nodes



map_server

ROS *map_server* node, as described in Section A.3

world_map

ROS *static_transform_publisher* node, as described in Section A.3

mocap

ROS *mocap_optitrack* node, as described in Section A.3

record

ROS *record* node, as described in Section A.3

benchmark

This is the ROS node in charge of running the benchmark. It is developed in Python, and takes care of all the needed tasks:

- runs the benchmark state machine
- loads the object list, with the corresponding transformation from the reference board marker set to the object frame, from a YAML configuration file
- acquires the position of the reference board from the motion capture system
- computes the actual position of the object reference frame
- computes the score

The only difference between the RoCKIn@Home and the RoCKIn@Work FBM1 is that in @Home the benchmark node is in charge of picking a random object from the object list, while in @Work it is the Referee Box that selects the object.

The script that runs this benchmark is named *fbm1h* (for RoCKIn@Home) or *fbm1w* (for RoCKIn@Work), and the source code is stored in the *rockin_benchmarking/scripts* folder.

refbox_test

This node simulates the Referee Box. It is developed in Python, and implements the Referee Box state machine to test the interaction with the rest of the system.

The script that simulates the Referee Box is named *fbm1h_refbox_test* and its source code is stored in the *rockin_benchmarking/scripts* folder.

client_test

This node simulates the Robot. It is developed in Python, and implements the Robot state machine to test the interaction with the rest of the system.

The script that simulates a Robot is named *fbm1h_client_test* and its source code is stored in the *rockin_benchmarking/scripts* folder.

Object list

The object list is a YAML file representing a Python list of all the objects. The YAML files are named *fbm1h.yaml* (for @Home objects) and *fbm1w.yaml* (for @Work objects), and they are stored in the *rockin_benchmarking/config* folder. The *fbm1h-vanilla.yaml* and *fbm1w-vanilla.yaml* YAML files contain the list of objects without the corresponding translations and rotations, and may be used to acquire the transoformations as described in Section A.5.

For each object, 5 parameters are specified:

- the ID, a unique key starting from 1
- the class
- the instance
- the translation with respect to the reference board frame (expressed in meters)
- the rotation with respect to the reference board frame (expressed in radiants)

```
items:
- class: a
  id: 1
  instance: a1
  rot: [-0.00751, 0.00246, -0.00631, 0.99994]
  trans: [0.01410, -0.25499, -0.01289]
- class: a
  id: 2
  instance: a2
  rot: [-0.00642, 0.00193, -0.00176, 0.99997]
  trans: [0.01642, -0.25131, -0.01429]
```

Listing A.11: fbm1h.yaml

Motion capture configuration

The motion capture system has to be carefully configured and calibrated for the correct execution of the benchmark.

First of all, two marker set have to be defined in the motive Optitrack system:

- the *origin* marker set, composed of the 4 markers fixed to the table (Motive ID 1)
- the *ref_board* marker set, composed of the 5 markers fixed to the reference board (Motive ID 2)

Then, to precisely align the origin frame with the AR codes attached to the table, proceed as follows:

1. align the Optirack ground plane calibration tool with the AR codes (a replica of the original tool with correct offsets has been made, and should be taped to the back of the table)
2. in Motive (calibration view), calibrate the ground plane

3. in Motive (creation view), select the origin marker set and add an offset to its coordinates so that they coincide with the origin (0, 0, 0, with angle 0)

4. after the calibration, the ground plane is not required to stay on the table any more (i.e., it can be removed, or placed in any other place covered by the Motion Capture System if needed, without compromising the benchmark execution)

Acquisition of new objects

To acquire the object pose, the transformation from the reference board marker set to the frame of the particular object has to be known. Indeed, the objects may have different frame origins and orientations, depending on their shape. For this reason, when adding new object to the list, they must be carefully acquired with the motion capture system to find the exact transformation.

The recommended setup is to place a camera on the top of the table, pointing at the origin, aligned with the Z axis. Then, display the video stream placing an overlay image showing the 3 axes (e.g., it can be done with VLC [8]), carefully aligning them to the origin. In this way, objects can be precisely placed in the origin, and at the same time images with the superimposed axes can be saved as a reference. A reference frame image is available in the *doc/items* folder. Images of the objects used at the RoCKIn 2014 Competition are stored in the *doc/items/roah* and *doc/items/roaw* folders.

Object acquisition is automated by a ROS script, which cycles the object list and acquires the transoformation for each object. The user is only required to place the objects in the desired position and press ENTER. The result is a YAML configuration file ready to be used for the FBM1; the file is saved in the */home/rockin* folder, and the user is required to overwrite the benchmark config file to use the new one.

The object acquisition script can be launched by calling roslaunch:

```
$ roslaunch rockin_scoring fbm1h_acquire_items.launch
```

Execution

Using the Referee Box

To start the FBM1 using the Referee Box, first of all switch to the correct environment:

```
$ fbmh_env
```

Then, launch the benchmark by calling *roslaunch*:

```
FBM1@HOME $ roslaunch rockin_scoring fbm1h.launch
```

The benchmark execution is managed by the refbox: whenever a client is ready, the benchmark node requests a manual operation (i.e., place the given object on the table), sends the goal, and waits for the result. At the end of the runs, the final score is computed and sent to the referee box.

[8] https://www.vlchelp.com/add-logo-watermarks-over-videos-vlc/

Figure A.1: RoCKIn FBM1 table and reference frame

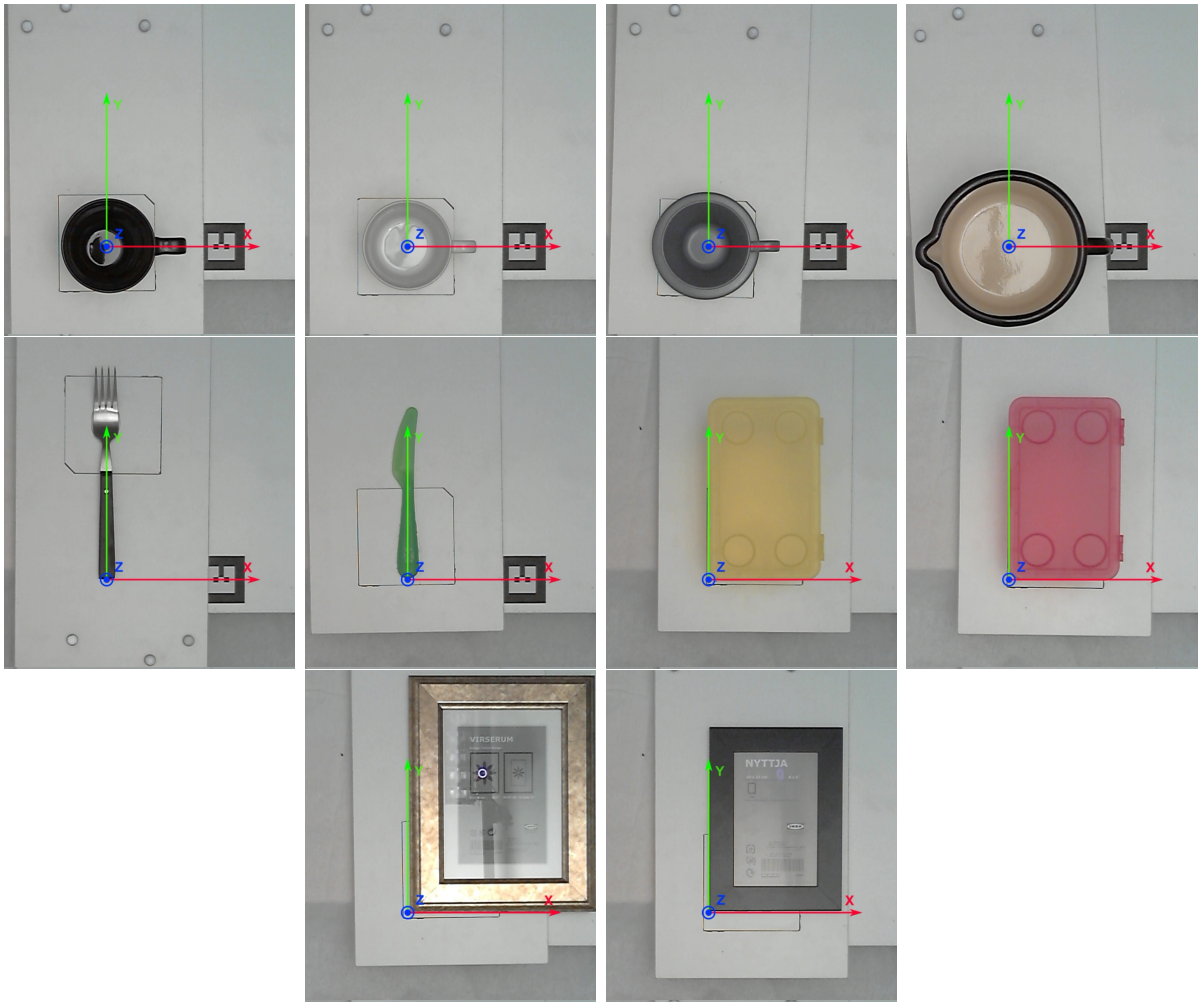


Figure A.2: RoCKIn@Home FBM1 objects

Manual execution

To start the FBM1 using the Referee Box, first of all switch to the correct environment:

```
$ fbmh_manual_env
```

Then, launch the benchmark by calling *roslaunch*:

```
FBM1@HOME (MANUAL) $ roslaunch rockin_scoring fbm1h_manual.launch
```

A terminal opens, requesting a manual operation (i.e., place the given object on the table). The referee places the object on the table, and the manual operation is confirmed by pressing ENTER on the terminal. When the robot has recognized the object, press ENTER again to conclude the run. The procedure is repeated for the number of runs, and a report is printed at the end of the benchmark.

Testing

A set of python scripts simulating the referee box and the robot can be used to test the FBM1 workflow. To run a test, firt of all switch to the correct environment:

```
$ fbmh_manual_env
```

Then, launch the test by calling *roslaunch*:

```
FBM1@HOME (MANUAL) $ roslaunch rockin_scoring fbm1h_test.launch
```

Logging

The benchmark launch files log each execution by launching a rosbag node. The logged topics are:

```
/fbm1h/bmbox_state
/fbm1h/client_state
/fbm1h/refbox_state
/fbm1h/map
/fbm1h/map_metadata
/fbm1h/origin/pose
/fbm1h/origin/pose2d
/fbm1h/ref_board/pose
/fbm1h/ref_board/pose2d
/fbm1h/info
/tf
/rosout
```

Resulting logs are stored in the */home/rockin/logs* folder.

