Shakey 2016 - How Much Does it Take to Redo Shakey the Robot?

David Speck, Christian Dornhege, and Wolfram Burgard

***Abstract*—Shakey the robot was one of the first autonomous robots that showed impressive capabilities of navigation and mobile manipulation. Since then, robotics research has made great progress showing more and more capable robotic systems for a large variety of application domains and tasks. In this paper we look back on decades of research by rebuilding Shakey with modern robotics technology in the open-source Shakey 2016 system. Hereby we demonstrate the impact of research by showing that ideas from the original Shakey are still alive in state-of-the-art systems, while robotics in general has improved to deliver more robust and more capable software and hardware. Our Shakey 2016 system has been implemented on real robots and leverages mostly open-source software. We experimentally evaluate the system in real-world scenarios on a PR2 robot and a Turtlebot-based robot and particularly investigate the development effort. The experiments documented in this paper demonstrate that results from robotics research are readily available for building complex robots like Shakey within a short amount of time and little effort.**

***Index Terms*—Autonomous Agents, AI-Based Methods**

I. Introduction

SHAKEY the robot was one of the first autonomous robotic systems. Decades ago, Shakey included capabilities that are still relevant for modern robot systems including localization, navigation, object detection, and manipulation by pushing, as well as high-level reasoning [1]. Shakey had significant impact on robotics and artificial intelligence research. This is not only due to the algorithms developed, but also because Shakey was a complete system and therefore individual components were shown to work together enabling the desired functionality. This demonstrated, what was possible at that point in time and thereby established the state of the art.

Highlighting the state of the art is—from a research perspective—a major reason for building complete systems. Shakey and many other systems integrate different approaches and thereby not only prove that they work well individually, but also allow us to learn how components developed independent from each other perform in combination. In addition, insights about the reliability and generality of algorithms can be gained. Another central point for the research community is availability. Complex systems require multiple different components that are ideally available to any roboticist to be able to utilize state-of-the-art algorithms.

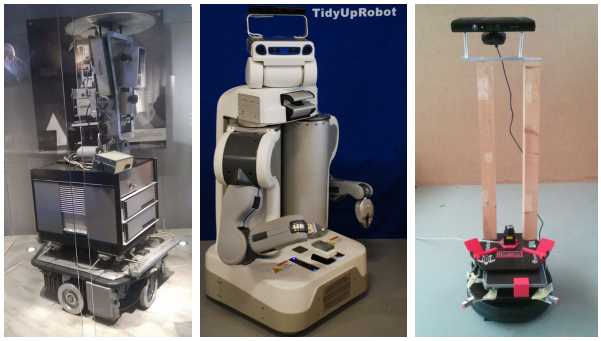

Fig. 1: Shakey at the Computer History Museum in Mountain View, California (left); The PR2 (middle) and the modified Turtlebot (right) used for the Shakey 2016 system.

In this paper, we recreate the Shakey system with state-of-the-art technology in hard- and software. We focus on algorithms that are available to the public as open-source libraries leveraging the progress made by the research community. We limit the amount of customized work as much as possible thereby proving that our Shakey 2016 system is the result of research and engineering of the last decades.

Our contribution is to demonstrate the impact of robotic systems by showing, which components of Shakey are in some form still relevant today, and in addition, we show the progress in robotics made during the years in terms of

- *Generality*: We make less assumptions about the environment, the algorithms work on different hardware platforms and are more robust to failures.
- *Availability*: The complete system is mainly constructed from open-source software and available online.[1]
- *Ease of use*: Shakey 2016 was created by a single master student and anyone should be able to reproduce it easily.

We first discuss the impact of robotics systems with examples of various systems that have been developed through the years. Then we shortly illustrate what the original Shakey system could do, before describing our Shakey 2016 system in detail. The evaluation not only shows the capabilities of a modern Shakey system, but also addresses the effort required to develop such a system.

Many robotic systems have been built over the years, each demonstrating what is possible at that time. With the

Manuscript received: September, 10, 2016; Revised December, 4, 2016; Accepted January, 21, 2017.

This paper was recommended for publication by Editor Tamim Asfour upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the German Research Foundation under research unit FOR 1513 (HYBRIS) and grant number EXC 1086.

All authors are with the Department of Computer Science at the University of Freiburg, 79110 Freiburg, Germany. david.speck@pluto.uni-freiburg.de, {dornhege, burgard}@cs.uni-freiburg.de

Digital Object Identifier (DOI): see top of this page.

[1] https://github.com/speckdavid/shakey2016

capabilities of robots becoming more and more general, so have the application areas.

A first focus was indoor navigation, where Shakey was one of the first systems [1]. Applications were pursued early on. For example, HelpMate was designed to carry medicine in hospitals [2] and Dervish performed navigation and delivery tasks in offices [3]. Such scenarios were also investigated in the 1993 robot competition [4]. Ultrasound was the main distance sensor and localization was performed mainly on local features or markers. Envisioned tasks also went beyond navigation, where manipulation in the form of pushing objects was not only performed by Shakey, but also a subject at the 1993 robot competition. Robotics was already connected with artificial intelligence especially in the form of high-level reasoning. Shakey with the Stanford Research Institute Problem Solver (STRIPS) introduced a formalism for planning that in some form is still in use today [5]. Xavier was instructed via a Web interface to perform various tasks in an office environment, among these telling "knock-knock"-jokes, and used at the highest level a task planner [6]. RHINO was an interactive tour guide robot that combined high-level problem solving with low-level probabilistic reasoning [7]. The topic of indoor navigation is still relevant today, especially with regard to reliability that allows robots to operate without interventions for long term deployments. One example for this is the CoBot system that performs tasks at CMU. CoBots mitigate their own limitations by actively asking help from humans [8].

Turning from indoor to outdoor navigation, a major application area is autonomous driving. Early systems demonstrated that this is within reach of state-of-the-art systems. A software named RALPH steered a car "no hands across America", where throttle and brakes were still controlled by humans [9]. Within the PROMETHEUS project in 1994 a fully autonomous car drove at speeds of up to 175 km/h on the German Autobahn. Other notable systems resulted from the DARPA challenges. Stanley, an autonomous SUV drove on unpaved roads within the DARPA grand challenge in 2005 [10]. The following urban challenge, where cars drove in urban environments was won by Boss [11]. These systems not only impacted science, but have been picked up by numerous companies for development into a product. Navigation has also considered city navigation on sidewalks. Recently, the robot Obelix managed to autonomously drive a three kilometer long route to the city center of Freiburg [12].

Service robots should be able to actively change their environment. One could consider early robots, such as Shakey, that push objects to be capable of manipulation. On a hardware level going beyond that has been possible for a long time by industrial robots, starting with the Unimate. However, these usually follow pre-programmed paths. Systems that operate in areas that are not fully specified, like household environments, became available only more recently. This is the result of more light-weight manipulators and better perception sensing. Integrated systems such as the mobile manipulation robot PR2 [13] or the manipulation platform Baxter [14] are now commonly found in many research institutions. Progress in this area was often driven by research competitions. The RoboCup@Home league focuses on service robots in household scenarios and tests integrated systems that work in various areas ranging from navigation over object recognition and manipulation to human robot interaction and cognitive reasoning [15]. Early systems here showed navigation capabilities, but also already high-level reasoning based on Golog [16]. Nowadays complex object handling is possible, for example by team Nimbro demonstrating mobile manipulation skills, such as watering a plant [17], [18]. While Shakey focused on a full system integration from high-level reasoning to low-level execution, competition systems mostly show individual skills combined into tasks. Besides Shakey's skills this includes among others grasping, people recognition and tracking, and speech or gesture recognition [19]. Recently, the Amazon picking challenge targeted detecting and grasping arbitrary objects. Team RBO from the TU Berlin team won this challenge by using a suction cup attached to a vacuum instead of mechanical actuation for the gripper [20]. Impressive robot systems were built for the DARPA robotics challenge that addressed disaster recovery scenarios. This required robots to be capable of complex manipulation tasks and navigation over rough terrain. In addition suitable user interfaces for controlling the robots had to be designed. Notable systems are the robot built by Schaft winning the trials and Hubo by KAIST that won the finals [21].

All these systems and many others have paved the way towards where we are now. Nevertheless, a look back on Shakey is valuable as shown in workshops at ICRA, AAAI and RSS during its 50 year anniversary in 2015. We now explain how we connect the research work started by the original Shakey system and continued by many researchers throughout the years to build Shakey 2016. We demonstrate that techniques developed for Shakey are still relevant today and that following research led to better systems like ours. Moreover, such state-of-the-art techniques are available to be leveraged by everyone and make it possible to easily recreate Shakey on modern hardware. The implementation is available as open source software and thus makes it possible to reproduce experiments with the PR2. In addition a minimal hardware setup with a Turtlebot-based platform allows anyone to recreate the Shakey 2016 system.

II. Shakey The Robot

The mobile robot system developed by the Artificial Intelligent Center at the Stanford Research Institute (SRI) nicknamed "Shakey" (Fig. 1) led to various improvements in AI and robotics. In this paper we discuss the second Shakey system, which was more advanced than the previous system and is well known for the film "SHAKEY: Experiments in Robot Planning and Learning". The system and its capabilities were tested with object rearranging experiments. In particular, two tasks are described. One to block and one to unblock a doorway in an environment consisting of connected rooms and pushable objects (boxes and wedges).

The robot used a differential drive and was equipped with a vidicon camera and an optical range finder mounted on a movable head. In addition, touch sensors were attached to the robot base. Instructions and information were sent via

radio links from the robot to a computer and vice versa. The architecture of Shakey was structured in five major levels. The lowest level was the interface between the robot system itself and its connection to the software programs. The next higher level were the Low Level Actions consisting of simple motion controls like ROLL and TILT. The third level called Intermediate Level Actions were predefined programs calling Low Level Actions. The purpose of the fourth level was to handle and solve planning problems. The fifth and highest level executed the plans and monitored them. Robot localization was computed based on the executed motion commands. A resulting localization error was from time to time corrected by an Intermediate Level Action called LANDMARK that took a picture of a near feature (e.g., corners or a door-jamb). That picture was analysed and used to calculate the corrected robot pose [1].

III. SHAKEY 2016

The Shakey 2016 system is todays successor of Shakey. Similar to the original Shakey experiments we define object rearranging tasks for our system. The setting is extended, generalized and set in a context of tidying up several connected rooms. The goal is to search for boxes and wedges that should be tidied up by pushing them to goal positions. More specifically, the goal is to search for pushable objects at particular search locations and to push the observed objects to defined goal locations or to the closest wall, if no goal location in the same room is defined or free. Thus, the Shakey 2016 system requires various skills. First, detection, classification and pose estimation of pushable objects, as well as pushing objects to the closest wall or a chosen goal location is essential. In addition, the system determines blocked doorways and clears them, if necessary. Finally, the robot is able to move from a start pose to a goal pose, i.e., perform autonomous navigation. These skills are combined into plans to solve tasks of the described tidy up scenario.

Shakey 2016 has been implemented on the PR2 [13] robot. This platform has an omni-directional drive, a Hokuyo UTM-30LX laser scanner at the base for navigation, and a Kinect RGBD camera mounted on a movable head for perception. Computation is performed onboard on two servers with quad-core i7 Xeon processors with 24 GB of memory. Other hardware, such as manipulator arms is not required for Shakey 2016. To demonstrate a minimal hardware setup, we also tested the Shakey 2016 system on a modified Turtlebot (see Fig. 1) that has been equipped with a Hokuyo SimpleURG laser, a Kinect RGBD camera and a standard consumer laptop. The Robot Operating System (ROS) [22] on the software side forms the lowest level of the Shakey 2016 system. The next higher levels are the navigation level, the action level encapsulating individual skills and at the highest level a continual planning loop including a planner, action execution, state estimation and monitoring. The remainder of this section describes this system in detail.

A. Continual Planning

Like Shakey, Shakey 2016's decisions are driven by a task planner. Its main purpose is to integrate the individual skills into a system and to reliably combine these to solve a complex task. We rely on the principle of continual planning, which was implemented in ROS in our earlier work [23], [24]. This solves the problem that a robot must react to the outcome of its perceptions and actions, even when they are not executed as predicted. Fig. 2 illustrates, what happens in each cycle of the continual planning loop. The robot estimates its current state consisting of symbolic values (e.g., in what room the robot is) and numeric values (e.g., the 3d pose of an object) from perception data. Monitoring then determines, if the current plan still leads from this state to the goal. Otherwise a task planner is called to produce a new plan. The next action of the current plan is mapped to the individual skills and executed resulting in a new world state.

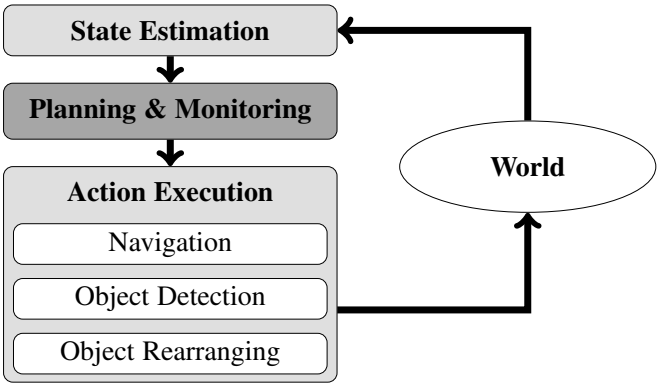

Fig. 2: Shakey 2016 is governed by a planning loop that continually estimates the current state, monitors or replans if necessary, and executes the appropriate robot skill.

In contrast to Shakey that used a classical task planner, we use an integrated task and motion planner, in our case Temporal Fast Downward/Modules (TFD/M) [25]. Here, a planning problem is not only solved symbolically, but geometric sub-solvers are integrated in the planning process as external reasoning modules. For Shakey 2016 we implemented two such modules, one for navigation and one for object rearranging. The navigation module determines, if goal positions are blocked in the world. For object rearranging we check, if push trajectories are valid.

B. Navigation

A basic requirement for goal-oriented navigation are mapping and localization, which are well understood nowadays, especially for structured indoor scenarios. First, the system builds a grid-based map of the environment from laser range data using *GMapping*[2] that implements a Rao-Blackwellized particle filter [26]. Localization in this map is realized with Monte-Carlo localization [27] implemented under the name *amcl* in ROS. In comparison to the original Shakey system, the localization in state-of-the-art mobile systems runs continuously and in parallel to navigation and for our system is not feature-based and can process dense data.

Navigation is handled by a combination of a long-term planner and a reactive execution component. Long-term collision-free paths on the grid map are generated using Dijkstra's algorithm. A local trajectory planner using the dynamic window

[2]http://www.openslam.org/gmapping.html

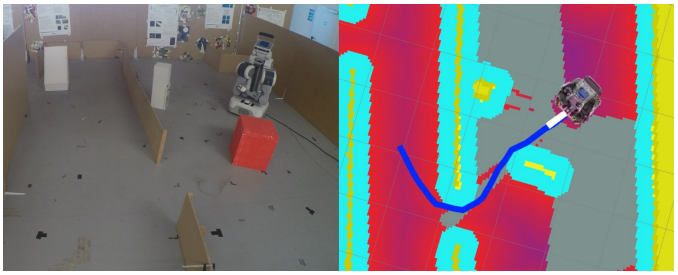

Fig. 3: A navigation task (left), where the robot needs to move past a box. The global plan (blue) and the local plan (white) drawn in a global cost map (right) guide the robot in the navigation task.

approach [28] generates velocity commands to drive the robot following the path. Fig. 3 shows an example of this.

C. Object Detection

We detect objects by shape and similar to the original Shakey setup use boxes and wedges. We identify pushable objects by planes (surfaces) that are characteristic for an object. For a box this is the square plane on top and for a wedge this is an inclined plane. We use plane segmentation of point clouds created from an RGBD camera. We apply random sample consensus (RANSAC) [29] implemented in the point cloud library (PCL) [30].

A plane parallel to the ground plane is considered to contain top surfaces of boxes. Surfaces are detected by point clusters lying on such a plane. Minimal 2D bounding boxes of the detected point clusters determine the corner points and normal vectors of the object. By this it is possible to compute the dimensions and poses of detected boxes (see Fig. 4). Similar to boxes, the system determines wedges and their corresponding dimensions and poses by searching for surfaces on a plane with a certain angular deviation (e.g., 45 ± 15 degrees) relative to the ground plane. In our experiments we search for boxes and wedges with a minimal side length of 20 cm.

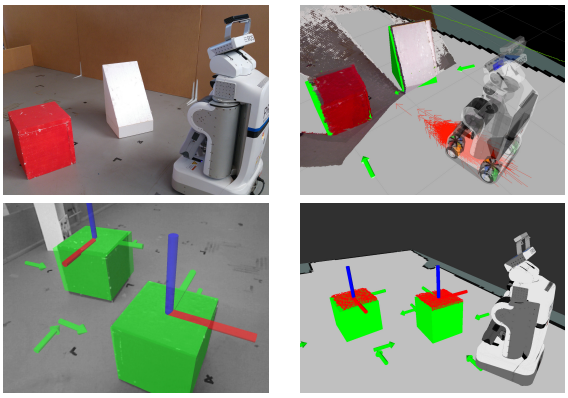

Fig. 4: The robot observes a box and a wedge with its RGBD camera (top left) and projects the objects (green) into its internal representation (top right). An overlay on the camera image shows two detected boxes with normal vectors (bottom left). The corresponding view in the representation of the robot shows overlaid point clusters (bottom right).

D. Object Rearranging

There are two different ways to tidy up objects in the Shakey 2016 scenario. The first one is to push objects to particular goal positions, if that is possible. This means that a goal position is defined in the room and it is not blocked by another object. In order to push an object to a goal position the robot always pushes the corresponding object according to its pushable sides. Thus, objects are pushed along the normal vectors of its sides. In this way, objects are aligned to the robot base. The push directions and positions are indicated in Fig. 4 (bottom) and Fig. 5 by green arrows on the ground. If a goal location is not aligned with the object, the robot pushes in two subsequent push actions to the goal position using two different pushable sides of the object. As mentioned in Section III-A it is necessary to check if such a two-step push action is possible given the object location and other objects or walls in the environment. This is handled during task planning by an external module that determines if a push is possible. An example where only one push series is possible is illustrated on the left-hand side of Fig. 5.

The second possibility to tidy up an object is chosen, if no suitable goal position for an object exists. In this case the object is pushed to the closest wall, which is determined using the map of the environment (right-hand side of Fig. 5). Finally, blocked doorways are cleared by pushing the blocking objects aside, which enables the transition between the connected rooms.

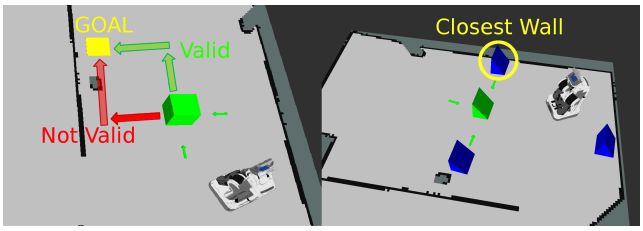

Fig. 5: Examples for object rearrangement plans. Left: Only one of two possible push trajectories of the green box to a defined goal location is valid due to an object on the left. The right side shows a wedge with its possible target locations (blue) towards the walls.

IV. Evaluation

We evaluate the Shakey 2016 system in two ways. First, we perform robot experiments illustrating the system capabilities. Second, we investigate the effort to recreate Shakey with state-of-the-art technology.

A. Experiments

The robot experiments are designed to show generality, robustness and computational performance. We performed 20 different runs with a simulated PR2 and an additional ten runs on the real robot to verify simulation results. In addition the Shakey 2016 system was executed on a modified Turtlebot and another ten runs on this platform were performed. Fig. 6 shows two simulated environments from the experiments and the real-world experimental scenarios used for the PR2 and for the Turtlebot experiments. We gave a generic task to the planner that defines a tidy-up setting as described in Section III. Object poses were not given to the robot, so that it needed to search for objects and robustly react to newly discovered objects becoming part of the problem to solve. The

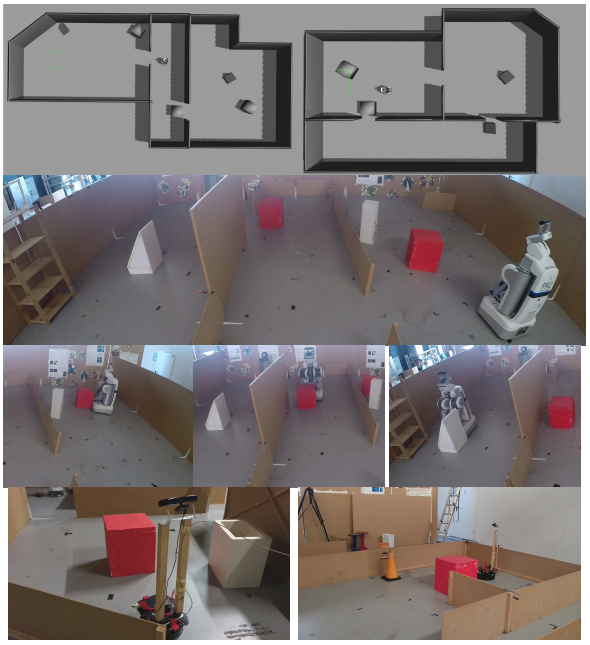

Fig. 6: This figure shows the evaluation scenarios. The two simulation scenarios are shown on the top. An overview of the real-world environment for the PR2 is shown in the middle. In this example task Shakey 2016 tidies two objects and clears a blocked doorway. The third row shows three actions: pushing the red box to its goal position, clearing a doorway, and pushing the wedge to its goal. The bottom row shows two small connected rooms used for the Turtlebot experiments.

Run	1	2	3	4	5	6	7	8	9	10
Scenario 1 (PR2)	14	14	18	16	18	20	24	22	24	26
Scenario 2 (PR2)	14	14	16	18	20	22	–	22	–	26
Real world (PR2)	12	12	16	16	–	18	–	–	18	18
Real world (Turtlebot)	11	11	9	9	–	–	13	13	17	13

TABLE I: This table shows the number of actions executed for each run. Real-world experiments used up to three, simulation experiments up to five objects. Entries with '-' show tasks that were not completely solved.

experiments consist of ten different runs in each scenario with up to five objects for the simulated and up to three objects for the real-world experiments. Search locations were given to the robot, so that it has a chance to detect the objects by moving the head.

Overall 33 of the 40 runs were completed successfully leading to a tidied up world with the planning loop terminating. An example run is shown in Fig. 6. Multiple replanning steps were usually required mainly because either a new object was detected or there were problems during navigation. Table I gives an overview of the executed actions, e.g., driving to a location or pushing a box for the individual tasks. Naturally tasks with more objects lead to more actions. Seven of the runs were ultimately unsuccessful due to the robot being stuck during navigation. The setting, which requires the robot to push boxes, produces such situations as it forces the robot to operate close to obstacles, something that navigation algorithms usually avoid. We are confident that a more sophisticated navigation setup and failure recovery procedures would mitigate this. However, the goal of this work was to show, what is achievable "out of the box".

Regarding the successful runs all objects were detected despite of one in the fifth run and all such detected objects were tidied up according to the domain specifications. In total, 99% of all objects were detected, all of the detected objects were successfully pushed to their goal locations or the closest wall and an overall 83% of all runs were a full success. The times over all runs are visualized in Fig. 7. The average total time of a run and the corresponding components, i.e., the action execution time and planning time are illustrated. It turns out that the planning time, which includes all replanning steps in the continual planning loop, is only a fractional part of the action execution time. Planning and execution times are smaller for the real-world environment as there are only three objects involved and the environment is smaller than what could be constructed in simulation. Execution times are smaller for the Turtlebot than the PR2 as the real-world test environment in this case was smaller. Overall, for these kind of scenarios, planning times even including external geometric reasoning are not a limiting factor. State-of-the-art task planning and integrated task and motion planning are only being pushed to their limit with more complex scenarios that involve a large number of objects, and especially when pick and place actions are involved. The results indicate a similar performance for both platforms, although the system was developed on the PR2 and only ported to the Turtlebot. This shows that the algorithms for Shakey 2016 are generic and independent of a specific platform.

B. Implementation Effort

In this section we address the initial question of "How much does it take to redo Shakey the robot?" with state-of-the-art hardware and software. As already mentioned in Section III, many robots capable of navigation are nowadays already equipped to build a system like Shakey 2016. In other words, in the field of mobile robotics hardware components such as laser scanners are generally available and with RGBD cameras such a system is even affordable for hobbyists. Although holonomic drive systems are often used, many robots, like the Turtlebot, still use a differential drive as on Shakey. Sensor technology has made remarkable improvements. Ultrasound sensors have been replaced by laser scanners and RGBD cameras make 3d perception possible. Another obvious progress from Shakey has been made in terms of computing power. Our robot experiments have shown that for this system it is not a limiting factor as it runs on a consumer laptop on the Turtlebot.

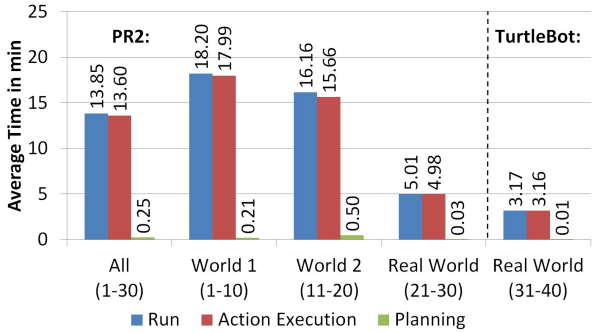

Fig. 7: This figure shows the average total run time, which consists of the action execution and planning time.

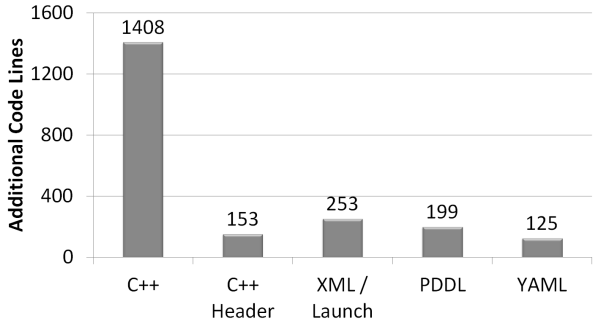

Fig. 8: Creating the complete Shakey 2016 system required less than $2,000$ additional lines of code over available open-source software and less than $1,000$ lines in launch scripts and configuration files.

In contrast to the amount of robotics algorithms at the time of Shakey, the algorithm knowledge in robotics nowadays is enormous. Localization, navigation and object detection is possible by drawing from a large amount of powerful algorithms. An important point here is that many state-of-the-art algorithms are included in robotic software collections such as the Robot Operating System (ROS) [22] or Player [31] and many more. Easily combining different components is especially important when developing robotic systems that require a variety of algorithms to perform a task. Availability of infrastructure software together with robotics algorithms leads to fast developments of robot systems and given that popular approaches are tested by many users to more robust ones.

In fact, this allowed us to easily redo Shakey. The packages and algorithms part of Shakey 2016 contain a large amount of code. We estimate the additional effort that it takes to redo Shakey the robot on the software side given, what is available publicly. Therefore, first, we counted the lines of code that were developed specifically for Shakey 2016, i.e., excluding all software that is already available as open source and has been used in another project. Fig. 8 gives an overview of the effort needed to build Shakey 2016. From the 2 000 lines of code the components on object recognition and object rearrangement required the most algorithmic work. A major part is interface code integrating the different components of the system with each other. Second, we estimated the amount of work required to actually build this system for a person not already familiar with the techniques involved. In our case a single Master student spent about eight man weeks in total from learning the basics of ROS to running the Shakey 2016 experiments on the PR2.

Afterwards this system was ported to the Turtlebot-based platform shown in Fig. 1. This was done within two to three work days and includes hardware modifications, such as mounting the Kinect sensor on top in a similar position as on the PR2 and software modifications that mainly consisted of integrating and calibrating the sensors and actuators of the robot. While the underlying low-level software, such as drivers and hardware parameters, for example the footprint of the robot, had to be adapted, it is important to note that the PR2 and Turtlebot systems both ran the same mid and high level algorithms without any major changes or tunings. The only notable difference is that the Turtlebot does not possess a movable head and therefore head turning actions had to be removed. In our case that meant that objects needed to be in the field of view of the camera to be recognized. Another alternative to recreate a similar behavior to the PR2 would be to use multiple search locations with the Turtlebot. The fact that—besides adaptations due to the changed hardware—the same system runs on both platforms shows that the Shakey 2016 system is generic and not build towards a specific robot.

The overall effort demonstrates that rebuilding a robot system that was groundbreaking 50 years ago is today possible with reasonably small effort. This was only possible by relying on published algorithms. Shakey becoming within reach of any roboticist is therefore the result of research and engineering progress in the last decades.

C. Shakey and Shakey 2016

The original Shakey system is well known for influential algorithmic developments. To name a few examples, the A* search algorithm, STRIPS, and the Hough transform are still in use in robotics and other computer science fields. In comparison to Shakey changes were made in Shakey 2016 although the system still requires the same kind of basic components. Localization and navigation for Shakey was based on vision and required distinct actions to localize based on known locations. In contrast Shakey 2016 uses laser range scanners with a full geometric map. Two factors contribute to the robustness of the navigation system. First, a probabilistic formulation of the localization system maintains a distribution of pose estimates instead of a single pose. Second, the complete navigation system is continuously updated during navigation in real time.

While Shakey could only perceive objects from 2d images, object detection for Shakey 2016 uses 3d data from an RGBD camera. Our method determines shapes by their characteristic planes in 3d. One could see this as an analogous extension of detecting lines in 2d images. The object rearrangement problem in the Shakey 2016 system uses its geometric map to compute possible push actions before these are applied. The fact that this does not only happen, when the action is to be executed, but already during task planning is the most distinctive feature of the high-level planning system of Shakey 2016. The STRIPS planner in Shakey was a symbolic system that as such dealt with symbolic actions. Integrated task and motion planning that considers motion planning during task planning has become an established technique in recent years especially when complex mobile manipulation scenarios are addressed. Our planner itself (TFD/M) at its core still uses a variant of the A* search algorithm. Although other approaches to task planning have been investigated search-based techniques are still state of the art.

Shakey 2016 has been developed to recreate the original Shakey system. Our goal was to demonstrate this by leveraging accessible open-source algorithms. While this was successful and resulted in a working system improvements are possible by replacing components with newer or adapted versions. One example is the SLAM component of the navigation system. Here graph-based methods are nowadays preferred, although the particle-filter approach worked well for the scenarios that

we tested. Shakey 2016 is only capable of solving similar tasks to Shakey. This means that our system is limited to the skills required in these tasks. There are multiple capabilities that other state-of-the-art systems demonstrate reliably. For example, the detection of specific objects beyond generic shapes, e.g., a corn flakes box, which can be solved by learning methods or manipulation that allows to pick and place objects. This is a skill that many robots, including the PR2, already have and a distinct advancement of modern systems over the original Shakey.

V. Conclusion

In this paper we presented the development of a mobile robot system called Shakey 2016—a state-of-the-art version of the famous robot Shakey. We evaluated with simulated and real world experiments showing that it is a stable, robust and well-performing system. We compared our Shakey 2016 system with the methods used in the original Shakey and pointed out advancements that are the result of progress in robotics. In addition, we showed which algorithms or ideas are still relevant today. The system is available online and a minimal hardware setup demonstrated by the Turtlebot platform allows anyone to reproduce these experiments.

We believe that we hereby demonstrated and quantified that robotics has improved in many different ways over the last 50 years. First of all, the required hardware is these days available for a wide range of people. Second, a system based on the original Shakey project for the original problem setting can be built with under 2,000 lines of code within eight weeks. One thing that could be improved by the community is to not only provide algorithms, but also a common directory similar to OpenSLAM. Thus, creating a complex robot system only requires a comparably small amount of effort allowing other researchers to build on prior achievements. Pointing out which algorithms stand the test of time is only possible by integrating them into systems—something that indeed has been done repeatedly. The fact that crucial algorithms for a system like Shakey 2016 are publicly available is not only valuable for building systems, but also for research in general as this allows to verify and compare new algorithms as well as test different components in a state-of-the-art setting.

References

[1] N. J. Nilsson, "Shakey the robot," AI Center, SRI International, Tech. Rep., 1984.

[2] J. M. Evans, "HelpMate: an autonomous mobile robot courier for hospitals," in *Int. Conf. on Intelligent Robots and Systems (IROS)*, 1994.

[3] I. Nourbakhsh, R. Powers, and S. Birchfield, "Dervish: An office-navigating robot," *AI Magazine*, vol. 16, no. 2, 1995.

[4] I. Nourbakhsh, S. Morse, C. Becker, M. Balabanovic, E. Gat, R. Simmons, S. Goodridge, H. Potlapalli, D. Hinkle, K. Jung, and D. V. Vactor, "The winning robots from the 1993 robot competition," *AI Magazine*, vol. 14, no. 4, 1993.

[5] R. Fikes and N. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intellgence*, vol. 2, no. 3–4, pp. 189–208, 1971.

[6] R. G. Simmons, R. Goodwin, K. Z. Haigh, S. Koenig, J. O'Sullivan, and M. M. Veloso, "Xavier: Experience with a layered robot architecture," *ACM magazine*, vol. 8, no. 1-4, pp. 22–33, 1997.

[7] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *National Conference on Artificial Intelligence (AAAI)*, 1998.

[8] J. Biswas and M. Veloso, "Localization and navigation of the cobots over long-term deployments," *Int. Journal of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, 2013.

[9] D. Pomerleau, "RALPH: Rapidly adapting lateral position handler," in *Symp. Intelligent Vehicles*, 1995, pp. 506–511.

[10] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal on Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[11] C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms, "Autonomous driving in traffic: Boss and the urban challenge," *AI magazine*, vol. 30, no. 2, 2009.

[12] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "Autonomous robot navigation in highly populated pedestrian zones," *Journal on Field Robotics*, vol. 32, no. 4, pp. 565–589, 2015.

[13] S. Cousins, "ROS on the PR2," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 23–25, 2010.

[14] C. Fitzgerald, "Developing baxter," in *Int. Conf. on Technologies for Practical Robot Applications (TePRA)*, 2013, pp. 1–6.

[15] D. Holz, L. Iocchi, and T. Van Der Zant, "Benchmarking intelligent service robots through scientific competitions: The RoboCup@Home approach." in *AAAI Spring Symposium*, 2013.

[16] S. Schiffer, A. Ferrein, and G. Lakemeyer, "AllemaniACs@Home 2007 team description," in *RoboCup 2007 team description papers*, 2007.

[17] J. Stuckler, D. Holz, and S. Behnke, "Demonstrating everyday manipulation skills in RoboCup@Home," *IEEE Robotics and Automation Magazine*, pp. 34–42, 2012.

[18] J. Stückler, I. Badami, D. Droeschel, K. Gräve, D. Holz, M. McElhone, M. Nieuwenhuisen, M. Schreiber, M. Schwarz, and S. Behnke, "Nimbro@ home: Winning team of the robocup@home competition 2012," in *RoboCup 2012: Robot soccer world cup XVI*, 2013.

[19] L. Iocchi, D. Holz, J. Ruiz-del-Solar, K. Sugiura, and T. van der Zant, "RoboCup@Home: Analysis and results of evolving competitions for domestic and service robots," *Artificial Intelligence*, vol. 229, pp. 258–281, 2015.

[20] C. Eppner, S. Hfer, R. Jonschkowski, R. Martn-Martn, A. Sieverling, V. Wall, and O. Brock, "Lessons from the amazon picking challenge: Four aspects of building robotic systems," in *Robotics, Science and Systems (RSS)*, 2016.

[21] J. Lim, I. Shim, O. Sim, H. Joe, I. Kim, J. Lee, and J. H. Oh, "Robotic software system for the disaster circumstances: System of team kaist in the darpa robotics challenge finals," in *Int. Conf. on Humanoid Robots (Humanoids)*, 2015.

[22] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, 2009.

[23] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[24] C. Dornhege and A. Hertle, "Integrated symbolic planning in the tidyup-robot project." in *AAAI Spring Symposium*, 2013.

[25] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," in *International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[26] G. Grisetti, C. Stachniss, , and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[27] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Int. Conf. on Robotics & Automation (ICRA)*, 1999.

[28] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[30] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Int. Conf. on Robotics & Automation (ICRA)*, 2011, pp. 1–4.

[31] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Int. Conf. on Advanced Robotics (ICAR)*, 2003.