DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Philosophy Doctoral Thesis in Information Engineering

USING KUKA YOUBOT FOR TEACHING ASSISTANCE

NGUYEN Duc Thien
1500916

Supervisors:

Professor Luca Iocchi
and
Professor Massimo Mecella

Department of Computer, Control and Management Engineering
Sapienza University of Rome, Italy
November 2014

Contact:
NGUYEN Duc Thien
Department of Computer, Control and Management Engineering Sapienza
University of Rome, Italy
Via Ariosto 25, 00185 Roma, Italy.
e-mail: thiennd@dis.uniroma1.it
http://www.dis.uniroma1.it/∼dottoratoii/students/nguyen-duc-thien

Abstract

Robotics in education has been rapidly become a popular way to engage students in the fundamental STEM (Science, Technology, Engineering, Physic and Math) concepts. Robots will soon become a useful tool for teaching process.

In this thesis, we investigated the effectiveness of robotics in education when a robots acts as the teaching assistance. Rather than focus just on the technology and robotic itself, we examined the overall learning and teaching environment where the robot becomes a teaching assistance when groups of students participate in teams with robotic activities.

We first begin by outlining the history of educational robotic in practice, illustrating its growing popularity on an international scale and describing how it has been evolved into the international uses. Although the popularity of the event is self-evident, we proposed a solution to identify and quantify the educational benefits of the initiative.

Second, in terms of robotic as experiments, our robot has an arm with 5 joints, it can pick up, place, drop and move several small objects such as pens, rolls of tape, wooden cubes size 3x3x3 (cm), 3x6x3 (cm), 3x9x3 (cm), 3x12x3 (cm), 3x15x3 (cm), etc. It can also push the button such as turn on/off the light, push the little wheel or push the door. Moreover, our robot can solve automatically some basic problems, for example, "Writing", "Cleaning", etc and some physic and informatics problems, for example, Elastic Collision, Tower of Hanoi problem, Sorting algorithm, etc.

Third, involving robotic in education, we made several experiments when the robots acts as a teaching assistance in the classes. Robot, beyond attracting the attention of students, also supports teachers in the teaching process as a very useful utility. We also design a lesson plan involving the robot in the classes to help the teacher in the classes.

For validation, we made questionnaires in a case of study for lectures with and without using robot. To clearly recognize the effect of using the robot in the classes, we designed the questionnaires and clustering the group study with and without robot to perform statistic data and comparison. Based on this study, we also made several observations about factors that we believe contributed to the success of the education.

Finally, we present results from a questionnaire study conducted with the robot and the comparison of the two groups is made for validation. As a primal results, the using of the robot as the teaching assistance has improved the percentage of correct answers of roughly 15%, level of attention of roughly 12% and the comprehension of roughly 11.2%.

For the future works, we will refine and repeat more experiments with other groups of students and let the robot also solve more problems in different subjects.

Acknowledgements

To my parents, my beloved wife and son!

First of all, I wish to thank the Sapienza University PhD Fellowships for my full-three-year financial support.

Special thanks to my supervisors Prof. Luca Iocchi and Prof. Massimo Mecella for their greatest support throughout my three years PhD study.

I would like to thank all members of laboratory RoCoCo (Cognitive Cooperating Robots), Prof. Daniele Nardi, Prof.Giorgio Grisetti, Dr.Domenico Daniele Bloisi, Andrea Pennisi, Taigo Maria Bonanni, Fabio Previtali, Guglielmo Gemignani, Martina Deturres, Maurilio Di Cicco, Jacopo Serafin, etc.

I also thanks Annalisa Terracina for organizing our educational experiments, thanks Maurizio Fuda, thank all my friends for helping me enjoy in Italy in the last three years.

Finally, I would like to thank my parents, my sister, my wife and my son! They are all of my life.

Thank you all!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we first present the problems and motivations beginning by outlining the history of educational robotic in practice, illustrating its growing popularity on an international scale and describing how it has been evolved into the international sensation and we propose our solutions as goals to identify and quantify in educational benefits. We also present the outline of this thesis and our main contributions of this work.

1.1 Problems, motivations and goals

Figure 1.1: Examples of robot as the teaching assistance in the class.

Robotics in education is fast becoming a popular way to engage students in the fundamental STEM concepts (Science, Technology, Engineering, Physic and Math). Robots will soon become a useful tool for teaching process. Educational robotics is the term now commonly used to refer to robotics being used as a tool for learning ([1], [2], [3], [4]). Since the early 1980s, robotics platforms designed for education represent a wide range of costs, types of parts, and complexities ([5], [6], [7], [8]). Many robotics kits include a programmable brick or controller and can be programmed in one or more languages [9]. Educational robotics programs can be grouped by the purpose for using robots; trends of using robotics as the learning objective, robotics as a learning aid, and robotics as a learning tool [10]. Throughout the programs, educators find robotics appealing because of the robots ability to catch the attention of youth; robots are highly engaging and motivating and encouraging learning of STEM concepts ([10], [11], [12], [13]). (See Figure 1.1)

In addition, robots provide a way for youth to quickly apply abstract concepts like mathematical equations to tangible tasks [14]. Further, robotics activities promote collaboration, teamwork, positive youth development, and foster learning of 21st Century Skills and computational thinking ([15], [16], [17], [18]). For example, in an introductory engineering course project, robots inspired increased shared leadership and engagement when compared to a similar assignment without robots [19]. Robots have even been used for storytelling [20] and by kindergartens to express aspects of their identities [21].

Based on existing problems, in this thesis, we investigated the effectiveness of robotics in education when the robots acts as the teaching assistance. Rather than focus just on the technology and robotic itself, we examined the overall learning and teaching environment where the robot become a teaching assistance when groups of students participate in team with robotic activities.

1.2 Thesis outline

This thesis is organized into the following chapters:

In Chapter 1, we first begin by outlining the history of educational robotic in practice, illustrating its growing popularity on an international scale and describing how it has been evolved into the international sensation and we find out the problems, motivations.

In Chapter 2, we demonstrate that the issue is still open and our solution is effective, we review relevant literature in robotic, in robotic education and in robotic teaching assistance and propose our solutions as goals in table 2.1 to

identify and quantify in educational benefits.

We introduce the KUKA youBot platform and its hardwares and softwares in Chapter 3. Chapter 4 presents the basic methodologies, including grasp planing and inverse kinematics which are required for the KUKA youBot to solve some problems or tasks automatically.

We show our first main contribution in Chapter 5. In terms of robotic as experiments, we present that our robot (namely KUKA youBot) can pick up, place, drop or move several small objects such as pens, rolls of tape or paper, wooden blocks size 3x3x3(cm), 3x6x3(cm), 3x9x3(cm), 3x12x3(cm), 3x15x3(cm), etc. These cubes are ideal for counting activities, patterns, beginning addition, subtraction, multiplication and division, pentomino building, explorations with volume and surface area as well as Tower of Hanoi or Sorting problem in informatics.

We show our second main contribution involving robotic in education in Chapter 6, we made several experiments when the robots acts as a teaching assistance in the classes. Robot, beyond attracting the attention of students, also supports teachers in the teaching process as a very useful utility. We also design a lesson plan engaging robots in the classes to help the teachers.

We present our third main contribution in Chapter 7. We made questionnaires in a case of study for lectures with and without using robot. To clearly recognize the effect of using robots in the classes, we designed the questionnaires and clustering the group study with and without robot to perform statistical results and experimental evaluations. Based on this study, we also made several observations about factors that is believed to contribute to the success of the education.

Finally, we present some results in terms of robotic and some results from the questionnaire study conducted with the robot and the comparison of the two groups is made for validation. As a primal results, the using of the robot as the teaching assistance has improved the percentage of correct answers of roughly 15%, level of attention of roughly 12% and the comprehension of roughly 11.2%.

In Chapter 8 shows the limitations, discussions, future works and conclusions. We will refine and repeat more experiments with other groups of student and let the robot also solve more problems in different subjects.

1.3 Thesis contributions

The main contributions in this thesis are:

1. Using some basic methodologies described in chapter 4, we designed, implemented and experimented the grasping object by the KUKA youBot automatically such as pens, rolls of tape or paper, wooden blocks size 3x3x3(cm), 3x6x3(cm), 3x9x3(cm), 3x12x3(cm), 3x15x3(cm), etc.. in order to solve some specific tasks. (see chapter 5)

2. We made several experiments in the classes when the robots acts as a teaching assistance. In this case, we invited some classes from different Roma high schools to carry out the experiments. (see chapter 6)

3. To validate and clearly recognize the positive effect of using robots in the classes, we designed questionnaires and analyze the results in a case of study for lectures with and without using robot (see chapter 7)

As a primal results, using of the robots acts as teaching assistance has improved the percentage of correct answers of roughly 15%, level of attention of roughly 12% and the comprehension of roughly 11.2%.

Chapter 2

Literature Reviews

In this chapter we demonstrate that the issue is open and our solution is effective, we review relevant literature in robotic education and in robotic teaching assistance.

2.1 Literature reviews in robotic education

In the [22], the authors design the course: Educational robotics for promoting 21st century skills. Although the focus of the course is the educational robotics and programming to control robots created with LEGO Mindstorms, the students identified their learning of collaboration and cooperation skills as well as communication skills as one of the best learning outcomes from the course.

Bonvillian [23] pointed out that Robert Solow of MIT won the Nobel prize in 1987 for establishing that technology and related innovation are responsible for at least half of U.S. economic growth. Industries based on technology need new scientists and engineers every year to help propel their success. It is optional to our schools to produce these graduates. Data suggests that United States is producing fewer science and technology workers.

Meanwhile, other countries are increasing the number of graduates in science and technology fields. Currently, U.S. students are less prepared than many other first-world countries in terms of science and math. At the fourth grade level, U.S. students are competitive in science but fall behind most first-world countries in math [24]. By the age of fifteen, U.S. students are still relatively poor math performers and fall behind the international average in science literacy as well [25]. If innovation is going to continue to drive the United States

economy, its educational system must improve these scores and entice graduates into STEM careers [23]. Waddell [26] tells a story that illustrates the risks of missing technological opportunities and falling behind competitively. At first, they tried to sell the technology to U.S. firms but they found little interest in their inventions. Meanwhile, in Japan the U.S. robotic technology was extremely well received and was eventually sold to a Japanese manufacturing company. Thirty years later there were five times as many manufacturing robots in Japan as there were in the United States.

One new approach to improving STEM education that is gaining popularity is the use of robots to teach STEM concepts. Advances in technology have brought down the cost of robots and made it easier to bring them into classrooms with tight budgets. Researchers and implementers have suggested several reasons why robots are one of the best ways to teach these subjects. Seymour Papert [1] laid much of the groundwork for using robots in the classroom. As far back as the 1970s, he and his team at MIT were pioneering the technological advances that make educational robots practical as well as the educational methods and theoretical foundations that explain how robots can help teach STEM concepts.

Many of the ideas that Papert wrote about in Mindstorms: children, computers, and powerful ideas were 20 years ahead of their time. Breaking with traditional computer aided instruction models where computers essentially programmed children, Papert wanted to create an environment where children programmed computers and robots. In doing so, the children would gain a sense of power over technology. He believed that children could identity with the robots because they were concrete, physical manifestations of the computer and the computer's programs.

The children learning with robots were able to imagine themselves in the place of the robot and understand how a computer's programming worked. Papert ([1], [2], [3]) believed that what makes many concepts difficult for children to understand is a lack of real-world materials that demonstrate the concept. He believed that programmable robots were flexible and powerful enough to be able to demonstrate ideas that previously had no easy real-world analogy. For example, the children could see that for a robot to move in a circle, it must move forward a little and then turn a little. This realization helped them see the connections between the abstract geometric theory of angles and a real-world example of them.

Other researchers have also identified the concrete nature of robots as being one of their important advantages. Students can understand abstract concepts and gain a more functional level of understanding by testing scientific and mechanical principles with the robots [27]. Students can also learn that in the real world there is not necessarily only one correct answer to every question [28]. Beer et al. [28] pointed out that it was more important for students to come up with creative solutions to problems than it was to recite answers they learned in class.

The success or failure of the robots demonstrated what their students learned better than any written exam could. Similarly, Papert [1] believed that learning with robots helps eliminate the fear of being wrong. He claimed that students using robots to learn from their experiences. Rather than seeing failure as something to be forgotten, children see it as more information they can put toward a correct solution.

Another argument for teaching children with robots is that they see the robots as toys [29]. One widely used kit of robotic equipment is made by Lego. Children using this kit can build and program robots out of the same materials they have in their toy chests at home. This makes anything they learn with them seem to be fun as well. Furthermore, it demonstrates that children can learn by playing.

Still another argument for using robots is that they tie into a variety of disciplines. A robot is made up of component parts of motors, sensors and programs. Each of these parts depends on different fields of knowledge like engineering, electronics, and computer science. This interdisciplinary nature of robots means that, when students learn to engineer robots, they will inevitably learn about the many other disciplines that robots depend on ([1], [30]). For instance, building a study robot requires learning about engineering strong structures. Meanwhile, understanding how to make a wheeled robot move faster requires an understanding of gears and torque. Understanding how gears increase or decrease torque requires an understanding of multiplication. Children learn these basic STEM concepts and, at the same time, they connect these concepts together into a greater understanding of robotics as a whole. Once children develop a sense of how the robotics work, they can apply these concepts as they make the robots interact with the real world. Papert [1] noted that before a child could program a robot to do a task, he or she would have to learn how to do the task first.

The children were motivated to understand the task by their desire to make the robot do their will. In the same way, teaching students how to build robots teaches them how all of the parts of a complex system interact and depend on each other [26]. Beer et al. [26] point out that this is an important lesson for computer science students who will ultimately need to create software that operates inside much larger systems. They go on to point out that this is as important a lesson to biologists as it is to engineers. A final practical reason to teach robotics is that the field of robotics is growing rapidly [26]. There are more than 80,000 robots working in American industry and those robots must be installed, maintained, and programmed. As advances in technology make robots more mobile, more capable, and more numerous, the need for workers with expertise in robots will grow.

2.2 Literature reviews in robotic teaching assistance

Researchers and instructors have incorporated robots into their curricula in a wide variety of ways and for many different age groups. Some approaches have been used robots as tools to assist in the teaching of actual programming languages ([23],[31]). For example, Fagin and Merkle [23] and Barnes [31] used robots to help teach the programming languages of Ada and Java, respectively. The main emphasis for their courses was on teaching the programming languages and basic programming structures rather than the engineering and mechanical aspects of robots. The robots were used to display tangible, physical feedback about what was happening in their programming code [31].

Other courses that use robots are focused on the construction and programming of the robots themselves ([27], [28]). Nourbakhsh et al. [27] taught a seven-week summer course for high-school seniors using custom-built robots designed specifically for the course. Their course started by teaching the basics of 7 motors, wiring and simple programming. The instructors used weekly challenges to focus the students activities and incrementally increase the sophistication of the topics being taught. By the end of the seven weeks, the students had taught their robots to dance, simulate defusing bombs, and navigate mazes. In the process of making their robots do these things, the students were taught about Java language programming, robotic sensors, electronics, wireless computer communication and other advanced topics. Beer et al. [28] used a similar approach in their classroom but, instead of simply focusing on the programmed behavior of the robots, the students designed and built their own robots. The

students were given kits that include more than 1300 pieces.

As the class progressed, students were taught both engineering and programming skills. Ultimately, the students put what they had learned to the test and built robots to compete in the final course activity, a robotic egg hunt. In this competition, the robots needed be able to navigate an arena, identify eggs of a particular colors, and bring those eggs back to a nest faster than the opposing team. To accomplish these goals they had to design, build, and program robots that could locate their nest, sort eggs based on their color, and grab and move them back to their nest. They also needed to avoid the other team's robot or risk being damaged or disabled in an accident. Each of these behaviors was complicated in itself. The real challenge for the students was putting all the pieces together. Moore [32] used robots to teach her fourth-grade students several different topics under the umbrella of examining robots. She used the topic of robots as a hook to capture her students attention. Then she weaved other disciplines into this central theme and asked her students to think critically about robots. She challenged her students to look at the history of robots, discover how robots are being used, what their advantages and disadvantages were, and speculate on how they might be used in the future. She had her students look at robots from a variety of different perspectives. Using robots, she taught subjects ranging from geometry to poetry. To assess how well her students had learned the topics, she gave them a task for a robot to perform and asked them to design, draw, and build a robot out of construction paper.

Some educators have used robotics to teach engineering subjects as early as kindergarten ([21], [30]). Using Lego robots and software called Robolab they designed a curriculum to teach kindergarten though fifth-grade students about engineering. Robolab is an icon-driven programming environment that allows the children to create simple programs visually without needing to know mysterious programming code words. Students taught with the curriculum were given tasks to accomplish with their robots that put math and engineering ideas into real world practice. Some researchers have proposed using virtual reality robots instead of real robots because of the considerable expense of robotics kits [33]. Instead of programming a physical robot, students manipulate and program a virtual robots represented on their computer screens. Geissler et al. [33] claim that virtual robots have many of the same advantages of real robots while costing significantly less.

Regarding positive impacts of using robots to teach science and technology, most of the literature describing the use of robots to teach science and technol-

ogy reports positive impacts on what their students learned about STEM. The positive impacts fall into six categories.

First, learning with robots is more interesting and improves students attitudes about STEM subjects ([34], [5], [23], [29], [30]). In interviews with teachers using robots in after-school programs, Robinson (2005) found that teachers thought their students had more positive feelings and were more aware while learning with robots. The teachers claimed that, when successful in building a robot, students felt good about it. Fagin and Merkle [29] reported that students taught with robots felt that the robots were interesting, fun, and relevant.

Second, learning with robots helps teach scientific and mathematic principles through experimentation with the robots. Rogers and Portsmore [30] reported success in teaching decimals at the second grade level by making a robot move for a time between one and two seconds. Through experiments, the students were able to learn decimals on their own. They discovered that 1.4 is less than 1.6 because their robots went further when they told it to travel 1.6 seconds. The also learned that 1.50 is the same as 1.5 even with the extra decimal place because the robot traveled the same distance with each value. Papert [1] used robots to teach concepts of geometry. He created robots with a pen which could be programmed to draw geometric shapes on paper. His students saw the relationships between programming, mathematics, and movement of the robot.

Third, building and programming robots require that the students develop problem solving skills (Nourbakhsh et al. [27], Beer et al. [27]; Mauch [29]). Beer et al. [27] emphasized that designing an entire system that was needed to work in the real world require problem solving skills that would serve them well in their future careers no matter what discipline they chose. In a comparison of students expectations from before and after a robotics summer camp, Nourbakhsh et al. [27] discovered that students were much more likely to claim problem solving as something they learned after the workshop.

Fourth, female students are more likely to appreciate learning with robots than traditional STEM teaching techniques. Rogers and Portsmore [30] also observed that females were likely to be interested in robotic activities if they could see a larger purpose for the robots they were constructing. Nourbakhsh et al. [27] found that, before exposure to robots, the high-school females in their study were significantly less comfortable with technology than the males. After the workshop, the females had increased their comfort level with technology more than the males.

Fifth, researchers reported that their students learned teamwork skills. Nourbakhsh et al. [27] noted that students mentioned teamwork on their surveys far more often after their exposure to robots than before. Beer et al. [27] identified teamwork as being one of the important outcomes of their robotics course.

Finally, Wagner [35] found that using robots as tools to teach science subjects can have the side effect of creating greater understanding of computer programming. Her experiment used robotics as a tool to demonstrate physics concepts. She compared the results of students using robotics-based demonstrators to students who used either battery-powered or no demonstrators. Students who used robotic demonstrators had an improved understanding of computer programming logic than students using the other methods. She did not find that students who used robots learned the physics concepts better than students using simpler battery powered demonstrators. There was no significant difference in scores between the two groups.

In negative impacts , while it is quite natural for researchers to want to find positive impacts of the methods they are using, every educational experience comes at the expense of other educational experiences. Sometimes these missed experiences are more important than the experiences offered by new methods. For these reasons, it is even more important to look at negative impacts of new educational technologies and methods. Not all of the results of using robots in the classroom have yielded positive results. In one of the very few quantitative studies that examined the use of robots effectiveness in the classroom with test scores, Fagin and Merkle [29] found that robots did not help introductory computer science students learn to program. In fact, the robots were a liability. Fagin and Merkle [29] compared the grades from standard exams given to students who were taught with robots to those taught with more traditional techniques. The students taught using robots had significantly lower scores. Furthermore, course surveys given to the robotics students found that students felt that the robots were hard to work on and time consuming during class. Fagin and Merkle [29] noted several limitations in their study that may have affected the performance of the students taught with robots.

First, because the budget for their computer science courses would not allow for enough equipment for 12 every student in every section of the course, students could only work on the robots during class time and lab periods. Downloading to the robot and testing them used even more valuable lab time. The traditionally taught students could work on their assignments outside of class

for as long as necessary, however, and did not need to wait for their programs to load on the robot. This put the robot-taught students at a significant disadvantage in terms of the time they could be on task.

A second factor was that instructors were teaching the robotics-based curriculum for the first time. Even though steps were taken to account for this, Fagin and Merkle [29] admitted that the instruction could probably be improved with more experience. They recommended that future efforts to teach with robots should allow the students to check out equipment to take back to their rooms.

Alternatively, they recommended that students should at least be given a way to simulate the robots in virtual reality without needing the robotic equipment itself so that they can work on their assignments outside of class time.

2.3 Literature reviews in mobile robot manipulation

A robot is a machine that can interact with its environment. One form of interaction is the manipulation of objects, e.g. picking up objects, moving them around, and putting them down again. Every attempt at object manipulation depends on an accurate and precise manipulator. These two properties enable the end-effector to move exactly where the object is. Conventional robot arms consist of rotatory and prismatic joints. While these can be controlled accurately and precisely in the joint space, it is more difficult to control a robotic arm directly in the Cartesian space. The calculation of the current end-effector pose in the Cartesian space from the joint positions is done using the forward kinematic chain. The inverse problem is much harder to compute. Depending on the degrees of freedom of a robot arm, the same end- effector pose might be reached by different joint positions [36].

On the other hand, not every end-effector pose in Cartesian space has a feasible solution in the joint space [37]. Even if all joint positions for a Cartesian space trajectory are known and feasible, there is no guarantee that tracking of the trajectory is precise. Control in joint space does not guarantee smooth motion in Cartesian space. Precise and accurate following of trajectories is important for various reasons. Unpredicted movements may damage the environment or the robot itself, or move objects out of position that have to be picked up. Inaccurate tracking of trajectories will cause the end effector to be

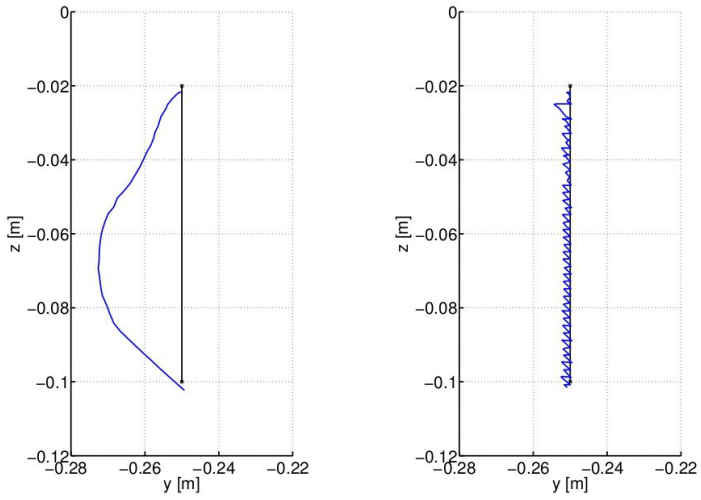
out of position. This could cause the robot to be damaged or simply cause a task to become impossible to complete. Especially tasks in dynamic surroundings consisting of moving objects [38] would suffer from inaccurate or imprecise tracking of trajectories. They [36] assume that for a feasible Cartesian trajectory, the joint positions for every point on the trajectory are known. As they are considering discrete time control, the number of points will be limited to the number of time-steps. Depending on how they choose to move from one point to the next, tracking of the trajectory will be accurate and smooth or inaccurate with unpredicted manipulator movements. The easiest way to move from one point to the next, is to simply command the joint to change its current position to the next. They call this the position control mode. Another method is to command a certain velocity to a joint until the desired position is reached. This is the so called velocity control mode.

For both of these methods, they do not know how the end-effector is moving between two points. A third method is to implement a torque controller, which calculates the required joint torques to switch from the current position to the next. The main advantage of the torque controller compared to the other methods is that it also takes the joint configuration and manipulator dynamics into account. This allows for smoother tracking of trajectories.

Figure 2.1: KUKA youBot

For the test platform, the KUKA youBot in Figure 2.1 which is presented

in detail in chapter 3, the position and velocity control methods deliver results which are unsatisfactory for certain tasks such as grasping of objects. As an example object, boxes are chosen. It was assumed that the best way to grasp a box is to position the end effector directly above the object and then lower it straight down.



(a) Direct step $\Delta t = 1.5\,\mathrm{s}$ (b) 1 mm step $\Delta t = 15.4\,\mathrm{s}$

Figure 2.2: Left: Lowering of the end effector in velocity mode without feedback. Right: Lowering of the end effector in velocity mode with feedback. Both trajectories are compared to the ideal trajectory in form of a straight line.

In Figure 2.2, it is shown that when lowering the end effector by directly commanding the new joint positions, it does not move in a straight line at all. Such a trajectory is very inconvenient when trying to grasp an object, as it is almost certain that the object will be moved and grasping fails. When commanding the lowering in 1 mm steps, the trajectory is much closer to the straight line. However, the task is taking up to 10 times longer to complete than any other tested method. In addition to the long time required, the end effector is also oscillating by a small measure which is inconvenient. Moreover, it was shown that the trajectory if the lowering is commanded in the velocity control mode. Without a joint position feedback to correct the velocity, the reached position differs vastly from the desired position. Even with a position feedback the manipulator does not move in a straight line. In short, none of the existing control schemes is able to accurately and quickly follow the desired trajectory. That is why a torque controller is implemented based on a dynamical model of the KUKA youBot arm. The torque controller is able to track a Cartesian

end-effector trajectory with a low position error while remaining fast. It is successfully used for autonomous grasping of objects and grasp while driving. Grasping objects requires detection of the objects in simple flat scenes. For this tasks the youBot is equipped with a Kinect camera system. A Kinect system is equipped with a color as well as an infrared camera which allows capturing of an image with a corresponding depth map. These two images can then be combined to form a color point cloud.

They developed an algorithm to detect and extract objects and their position and orientation in the point cloud. Afterwards, the robot moves independently to that position and picks up the objects using the designed torque controller.

In [39], the authors developed an unified closed form solution for the inverse kinematics of the youBot which allows fast computation of joint positions for Cartesian end- effector positions. Using this approach, generated Cartesian trajectories are easily transformed to the corresponding joint space trajectories required by the torque controller, as long as all the positions are feasible. In [40], an approach is proposed for the object manipulation on the KUKA youBot. Their system is able to detect objects autonomously in the environment then pick them up and put them down using a simple user interface. In order to effectively control the youBot arm they make use of the existing ROS (Robot Operating System) arm navigation stack and adapt it for the KUKA youBot. The stack allows to control the manipulator in Cartesian space based on the implemented inverse kinematic chain. After objects get detected using a Microsoft Kinect attached to the arm, the youBot is then navigated to a position from where the objects are grasped using a set of overhead cameras.

For grasping, the ROS object manipulation stack, which provides a framework for pick and place actions, was used. Contrary to this thesis, external stacks for efficient manipulator control, grasping of objects and was used, even with an external camera system to track the base. Authors also developed a torque controller which is designed for close tracking of trajectories with the assumption that grasping is always optimal when attempted from directly above.

In addition, the base is only controlled using the odometry messages generated by the KUKA youBot ROS stack. The object detection algorithm was developed by Paul Malmsten [41] and is very similar to the one implemented. The main difference is that their algorithm is using iterative closest point to match objects to a database, while no database is used and objects are simply detected by finding an appropriate bounding box.

Table 2.1: Outlining the history of educational robotic in practice and our proposals

Papers	Tasks	Age	Robots	Control/ Interaction	1st Learning Objective, Skills	2nd Learning Objective, Skills
Papert S. et al., (1980, 1986, 1993)	Tool Integrated Learning System	5-12	-	-	General science	General science
Eguchi, A et al., (2003, 2012)	RoboCup Junior: soccer	12-18	NAO, FischerTechnik Mobile Robot, Soccer-Robo	Programming	Robot Programming	Physics, mechanic, experimental skills
Lund, H.H et al., (1998. 2000)	LEGO Mindstorms: soccer	12-18	LEGO-bot	Control, Construct	Electronics	Communication, Teamwork, Organization
Eguchi, A et al., (2003, 2012)	RoboCup Junior: dance	6-12	LEGO-bot	Control, Construct	Electronics	Communication, Teamwork, Motivation
Eguchi, A et al., (2003, 2012)	RoboCup Junior: sumo	12-15	LEGO-bot	Control, Construct	Electronics	Electronics Communication, Teamwork, Motivation
Eguchi, A et al., (2003, 2012)	RoboCup Junior: rescue, navigation	12-15	LEGO-bot	Control	Robot Programming	Electronics Communication, Teamwork, Motivation
Our proposal	Informatics: Hanoi Tower, Sorting, etc	14-18	youBot manipulator	Control	Informatics	Mechanic, Communication, Teamwork, Motivation

Chapter 3

KUKA youBot

This chapter introduces the KUKA youBot platform and its hardwares and softwares.

3.1 KUKA youBot: Hardware

The robot used in this thesis is the KUKA youBot [42] (See Figure 3.1). The KUKA youBot is a mobile manipulator that was primarily developed for education and research. It become a standard platform for the community that is interested in education and research in mobile manipulation.

The KUKA youBot arm has five degrees of freedoms (DOF) and a two-finger gripper. If connected to the mobile platform, the arm can be controlled by the onboard PC. Alternatively, the arm can be controlled without the mobile platform by using an own PC connected via Ethernet cable.

The KUKA youBot omni-directional mobile platform consists of the robot chassis, four mecanum wheels, motors, power and an onboard PC board. Users can either run programs on this board, or control it from a remote computer. The platform comes with a Live-USB stick with preinstalled Ubuntu Linux and drivers for the hardware. Details of the software are described below.

The KUKA youBot comes with fully open interfaces and allows the developers to access the system on nearly all levels of hardware control. It further comes with an application programming interface (KUKA youBot API), with interfaces and wrappers for recent robotic frameworks such as ROS (Robot Operating System) [43], with an open source simulation in Gazebo [44] and with some example code that demonstrates how to program the KUKA youBot. The

Figure 3.1: KUKA youBot platform with some sensors

platform and the available software shall enable the user to rapidly develop his/her own mobile manipulation applications.

Additional sensors can be mounted on the robot, for example a 3D vision sensor Kinect [45] or sensor rangefinders Hokuyo [46]. This thesis will also provide you with information on the sensors working with the KUKA youBot, so far.

Giving full access to nearly all levels of control of the robot comes at a price, however. It allows the user to access and control each of its actuators and sensors directly as he or she thinks is best. Users can develop applications whose functionalities are only limited by the KUKA youBot's kinematic structure, its drive electronics, and motors. Such applications could damage the robot itself or other objects in its workspace or could result in harm of a human being or an animal. The KUKA youBot qualifies as a so-called partly completed machinery. Users will notice that the KUKA youBot does not come with any preinstalled software. All software will have to be installed by the user him-/her-self (from a Live-USB stick delivered with the robot) to turn the KUKA youBot into an operational machine. This is to emphasize that it is solely the responsibility of the user to turn to the KUKA youBot into a safe machine.

To work with and develop software for the KUKA youBot requires some basic knowledge in robot manipulation, mobile robotics, and mobile manipulation.

It further requires some basic understanding of the operating system Linux (Ubuntu).

3.2 The KUKA youBot omni-directional mobile platform

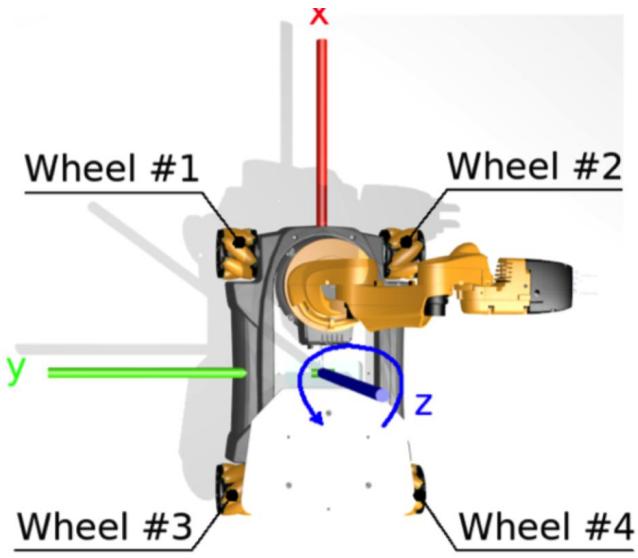


Figure 3.2: Overview of the KUKA youBot base. The figure illustrates the attached base frame. Positive values for rotation about the z axis result in a counterclockwise movement, as indicated by the blue arrow.

The KUKA youBot base is an omni-directional mobile platform with four mecanum wheels. It consists of a mini PC running Ubuntu, a battery and four omnidirectional wheels with separate motors. These allow the youBot to turn on the spot as well as move in any direction without having to turn first. The maximum speed of the base is 0.8 m/s . It has a weight of 20 kg, and measures 580 mm by 380 mm by 140 mm.

Figure 3.2 illustrates the attached base (coordinate) frame, as it will be used in the KUKA youBot API. It is located in the center of odometry. Positive values for rotation about the z axis result in a counterclockwise movement, as indicated by the blue arrow. The wheel numbering for the mecanum wheels is also shown.

The motor controllers of the four omni-directional wheels can be accessed via Ethernet and EtherCAT. The Ethernet cable can be either plugged into the

onboard PC or to an external computer. The plug is a standard network cable plug that fits into each standard Ethernet port. On the top side of the KUKA youBot base you will find a panel with power plugs (labeled as 24V OUT) and Ethernet ports (labeled as EtherCAT 1 and EtherCAT 2) for up to two KUKA youBot arms. The third power plug (labeled as 24V IN) can be used to power the KUKA youBot and recharge the battery (at the same time). There is a single Ethernet slot (labeled as Ethernet) for connecting the onboard PC to a LAN via network cable. A LCD screen will indicate the battery status. If you want to connect the internal EtherCAT controllers to an external PC you have to open the cover of the base on the right side, unplug the Ethernet cable and replug it to an extension cable that goes into the external computer [42].

3.3 KUKA youBot arm

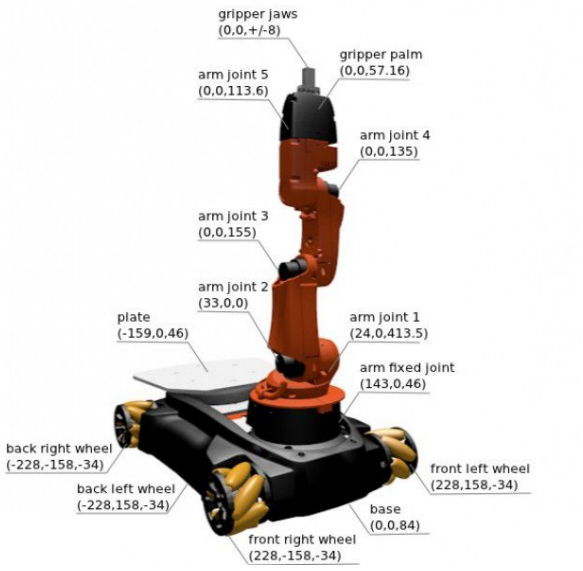


Figure 3.3: The KUKA youBot arm and base

The arm of the KUKA youBot (see Figure 3.3) consists of five rotatory joints and a two-finger gripper as an end effector. It reaches a height of 655 mm, has a weight of 6.3 kg and is designed to carry a payload of up to 0.5 kg in weight. The rotatory joints of the arm rotate around two different axes. Joints 2nd, 3rd, and 4th are rotating in parallel around one axis and joints 1st and 5th in parallel around the other if the arm is pointing straight up. The maximum rotation speed of a joint is 90 degree/s and the end effector can be opened 11.5 mm per finger.

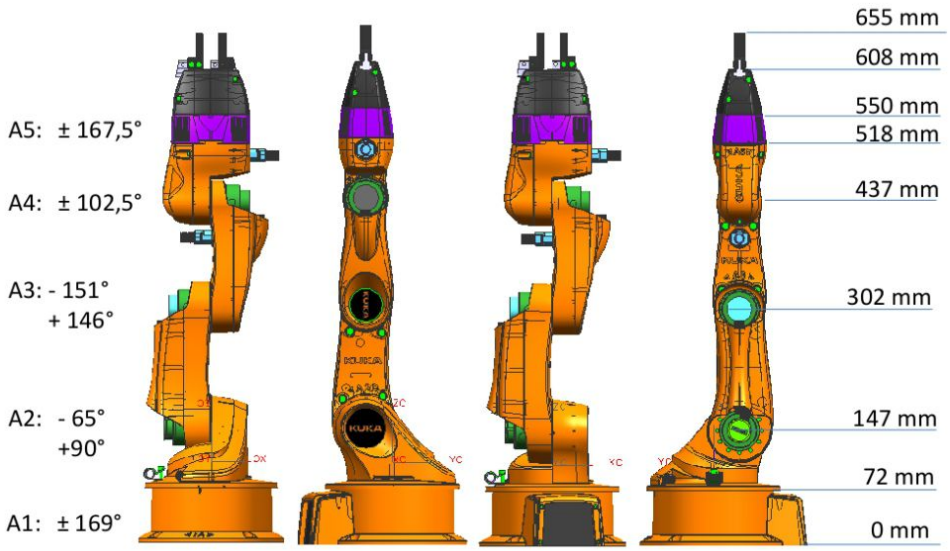


Figure 3.4: Overview of the kinematic structure of the KUKA youBot arm. The figure illustrates joints with limits, and the length of links between joints and base.

The KUKA youBot arm is a serial chain with five revolute axes. The end effector is a two-finger gripper, that can be removed. Figure 3.4 illustrates the basic kinematic structure. Similar to the base the motor drivers for the individual joints can be accessed via Ethernet and EtherCAT.

It is possible to work locally on the KUKA youBot PC by connecting a keyboard, a mouse and a monitor to its ports. This is the recommended and easiest way for the initial installation. You may further have to resort to this configuration when you face problems with the network connectivity. Mouse and keyboard must have USB connectors. For the monitor a VGA cable is required. Please, note that the onboard PC has neither PS2, DVI nor HDMI connections. If you want to use such interfaces you may have to use appropriate adapters.

Since 2012, RoboCup@Work [47] is a new competition in RoboCup that targets the use of robots (mainly used the KUKA youBot) in work-related scenarios. It aims to foster research and development that enables use of innovative mobile robots equipped with advanced manipulators for current and future industrial applications, where robots cooperate with human workers for complex tasks ranging from manufacturing, automation, and parts handling up to general logistics. RoboCup@Work (see Figure 3.5) addresses many of the standard

scientific challenges for robotics, including the following non-exhaustive list: (1) Perception in static and dynamic environments under varying environmental conditions. (2) Path planning and motion control of mobile bases in dynamic environments. (3) Grasp planning, trajectory planning, and motion control of mobile manipulators. (4) Planning and decision making. (5) Representation of plans, knowledge, strategy and tactics. (6) Adaptivity and learning. (7) Cooperation in both cooperative and competitive environments. (8) Human-robot and robot-robot interaction. (9) Design, construction, and operation of robust robots at affordable cost. (10) Simulation, evaluation, and benchmarking of advanced robot systems.

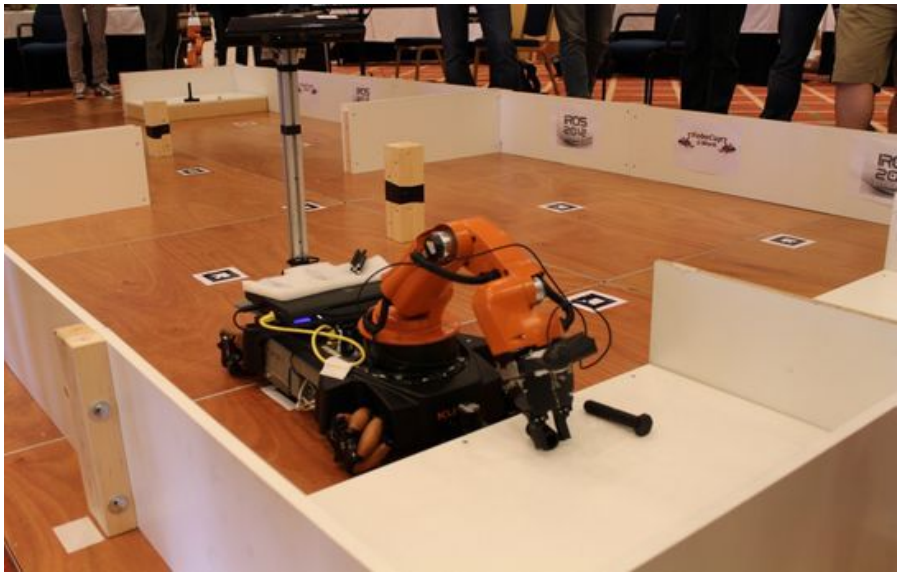


Figure 3.5: RoboCup@Work

In addition, the RoboCup@Work League specifically targets several new challenges, which so far are not pursued by other competitions or RoboCup leagues: (1) Mobile Manipulation: While until the recent past industrial robotics concentrated on highly-precise but non-mobile manipulators, mobile robots either had no manipulators or only low-DoF robot arms. Recently, this situation is changing, and both the research community and industry have developed a strong interest in robust mobile manipulators. Introducing a competition that fosters research in that direction comes very timely. (2) Logistics: Logistics is an enormously important area in practically every business-related domain. It plays practically no role in any of the established RoboCup leagues so far. Rescue simulation league is a notable exception, but poses quite different constraints on logistics problems than those targeted by RoboCup@Work. (3) Cooperative

Mobile Manipulation: Once mobile manipulators are not a fantasy any more but reality, the next step would be to have such mobile manipulators cooperate with humans and/or other mobile manipulators. Reaching this objective requires solving numerous additional problems, but also opens many new application scenarios. (4) Multiagent Planning and Scheduling, and Multi-Criteria Optimization: The value of classical task planning in soccer and rescue leagues is rather limited; if it used at all, these planners have to deal with constraints that are quite different from those in most industrial domains. Task planning may eventually be of great interest in RoboCup@Home, but so far most teams work with preprogrammed scripts or state machines executing routine activities. In RoboCup@Work, however, multiagent planning and scheduling with multi-criteria optimization is of immediate interest, and has a large potential for innovative applications [47].

3.4 3D vision sensor kinect

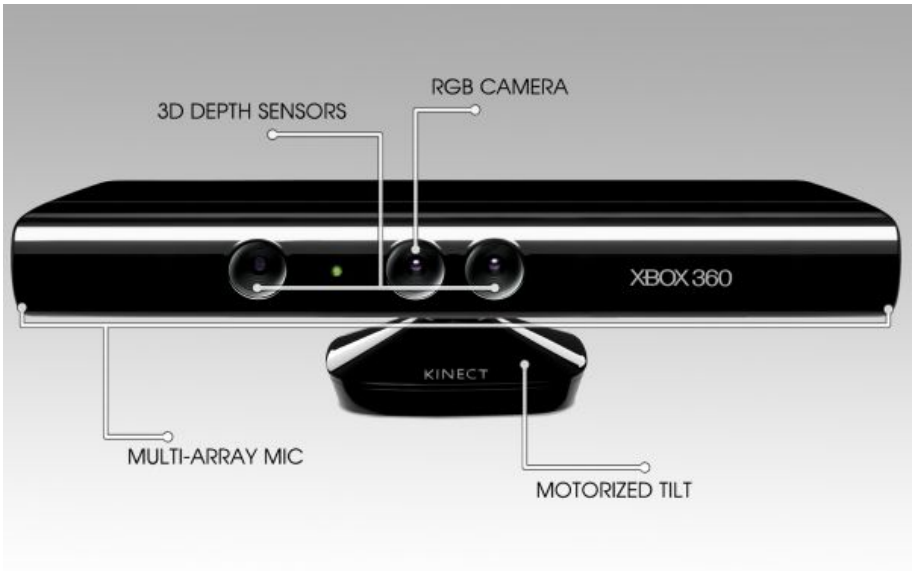


Figure 3.6: Microsoft 3D vision sensor kinect

We investigated a handful of tools to help us discover objects in our robot's environment and produce suitable representations of them in software. These include: The Microsoft Kinect [45] in Figure 3.6, a structured-light 3D camera designed for the Xbox game console. The Object Recognition Kitchen (ORK), an effort by Willow Garage to develop a general-purpose object detection and recognition library for ROS. The Tabletop Object Detector, an object

detection and recognition library for objects on tables the Point Cloud Library (PCL), a general purpose library for working with point cloud data structures Each of these will be introduced in the following sections.

The Microsoft 3D vision sensor Kinect is a consumer device originally designed by Microsoft as a peripheral for the Xbox game system. It was designed to compete against the motion-sensitive controller introduced by Nintendo for the Wii game system. The Microsoft Kinect is composite device which includes the following components: (1) A color digital camera (2) A structured-light infrared projector (3) An infrared digital camera (4) A microphone array (5) A tilt motor, which can adjust the pitch of the attached cameras and projector. The Kinect derives depth information from an environment by projecting a grid of infrared points in a predictable pattern. The resulting projection on the environment is viewed by the integrated infrared camera and interpreted to produce depth information for each point. An example of this projection is illustrated in Figure 3.7.

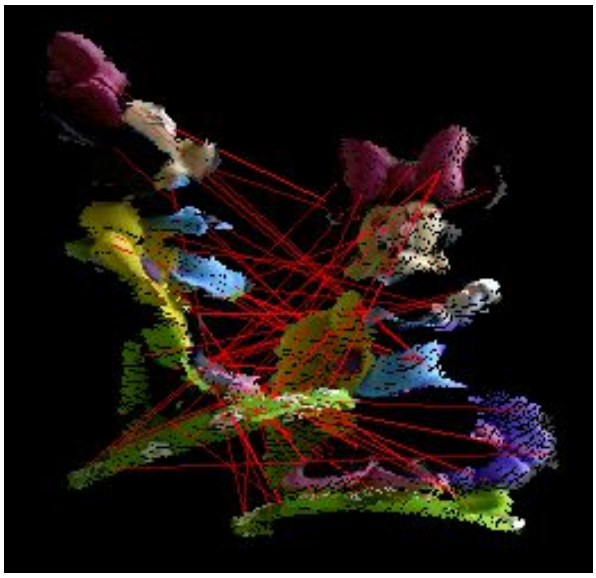


Figure 3.7: An example of PCL correspondence given by microsoft kinect

This structured light approach poses challenges for objects particularly near and objects particularly far from the sensor. For objects closer than 0.8 meters, the projected points appear too closely together for the sensor to measure; this results in a short-range blind spot that can have implications for where the sensor is mounted, particularly for small robots. For objects farther than 4 meters, the projected points fade into the background. This maximum range

is also adversely affected by the presence of infrared noise in the environment. As such, the Kinect's range is significantly degraded outdoors. To support game development, Microsoft has developed a software library for interpreting a human figure in this point cloud. This library allows users of the Microsoft tool chain for C++ and C to easily discover human figures in an environment and determine the 3D position of the figure's extremities. The impressive quality of the depth information produced by the Kinect and the Kinect's low price make it a very attractive sensor for robotics research. As a result, a variety of open-source drivers have been developed for the Kinect which allow one to process the information produced by a Kinect as a cloud of points located in 3D space. In addition, some of these libraries also provide depth registration, wherein each depth point is annotated with the RGB color of that point in the environment. We have investigated the *openni-camera* package for ROS.
This package produces ROS point cloud message data structures which makes it easy to use a Kinect with many existing ROS packages and infrastructure. In addition, *openni-camera* also supports depth registration.

3.5 KUKA youBot software: ROS

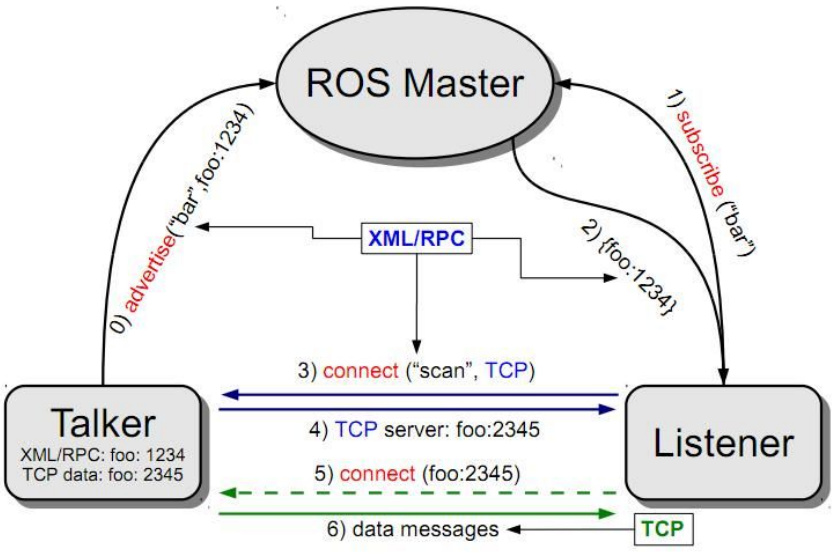


Figure 3.8: Nodes communicate in a ROS environment

ROS (Robot Operating System) [43] is an open source software framework for robotic development. The primary goal of ROS is to provide a common platform to make the construction of capable robotic applications quicker and

easier. Some of the features it provides include hardware abstraction, device drivers, message-passing, and package management. ROS was originally developed starting in 2007 under the name Switchyard by the Stanford Artificial Intelligence Laboratory, but since 2008 it has been primarily developed by Willow Garage and is currently in its seventh release. The fundamental purpose of ROS is to provide an extensible interprocess communication framework which simplifies the design of distributed systems. The building blocks of a ROS application are nodes; a node is a named entity that can communicate with other ROS nodes on behalf of an operating system process. At this time, ROS provides support for nodes written in C++ and Python, and experimental libraries exist for a handful of other languages.

There are three ways that nodes may communicate in a ROS environment: (1) By publishing messages to a topic. (2) By listening to messages published on a topic. (3) By calling a service provided by another node. Messages represent data structures that may be transferred between nodes. Messages may contain any named fields; each field may be a primitive data type (e.g. integers, booleans, floating-point numbers, or strings), a message type, or an array of either type. ROS provides a mechanism for generating source code from text definitions of messages. ROS topics represent named channels over which a particular type of message may be transferred. A topic provides one-directional communication between any number publishing and consuming nodes. Figure 3.8 provides a visualization of a ROS graph; ovals represent nodes, and the directed arrows represent publish/subscribe relationships between nodes via a particular topic.

A node that publishes a topic is called a publisher. Publishers often run continuously in order to provide sensor readings or other periodic information to other nodes; however, it is possible to write a publisher that publishes a message only once. A node which listens to messages on a topic is called a subscriber. A subscriber specifies which topic it wants to listen to as well as the expected message type for the topic, and registers a callback function to be executed whenever a message is received. Similar to publishing a single message, a subscriber may instead block until a message is received if only a single message is required. Finally, a node may provide a service to other nodes. A service call in ROS resembles a remote procedure call work-flow: a client node sends a request to the service provider node, a registered callback function in the service provider node performs the appropriate action(s), and then the service provider sends a response back to the client. Since a service call is an exchange between one client node and one service provider node, they do not employ

topics. Service request and response messages, however, are defined in a similar manner as standard messages, and they may include standard messages as fields. Any node can perform any combination of these actions; for example, a node could subscribe to a topic, call a service on a message it receives, and then publish the result to another topic. This allows a large amount of flexibility in ROS applications. ROS also allows applications to be distributed across multiple machines, with the only restriction that nodes which require hardware resources to run on a machine where those resources are available. More information about ROS including tutorials, instillation instructions, and package information can be found on their website: *www.ros.org/wiki*.

3.6 KUKA youBot software: Gazebo and webots simulation

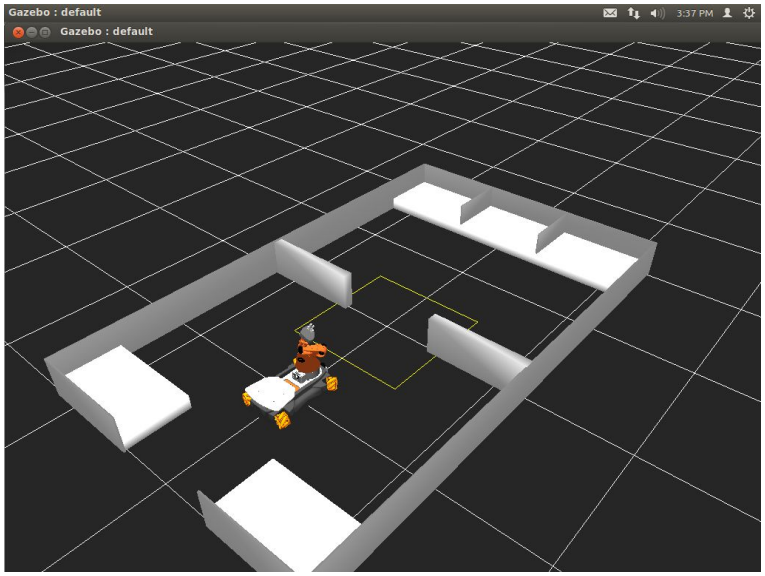


Figure 3.9: Gazebo simulation

To simulate the actions and the tasks of the KUKA youBot, we usually used two simulators. First, the Gazebo simulator [44] is a multi-robot simulator, primarily designed for outdoor environments. The system is compatible with ROS, making it a good choice for representing the robot's environment. Gazebo also features rigid body physics simulation, allowing for collision detection and object manipulation. Finally, Gazebo allows for the simulation of robot sensors, allowing us to incorporate the Kinect's full functionality and the overhead cameras into the simulation.

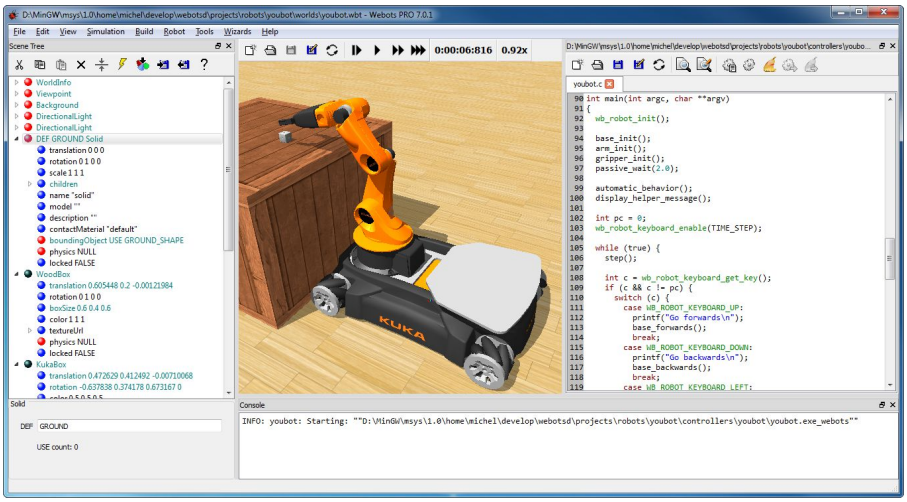


Figure 3.10: Example of a webots simulation with the KUKA youBot and graspable objects

The room as it is envisioned in the final product is modeled in the Gazebo simulation. An example image of the simulation is shown in Figure 3.9. The flat images projected in the air represent the overhead cameras point of view, which is used in the web interface. Both the robot and arm are fully represented and capable of being manipulated within the simulation. All of the objects that the robot is expected to interact with in the room are also present and can be added and moved. As we determine what objects should be present in the environment based upon the design decisions that we make and the experience that we gain working with the robot, the gazebo simulation and the objects included have to be updated to reflect the changes.

Second, Webots [48] is a development environment for robotics which is used to create advanced simulations. It comes with an implemented model of the KUKA youBot. Webots supports physics supplied by plug-ins, which are adapted for different cases. Figure 3.10 shows the main window of a Webots simulation. On the left is the scene tree which consists of all the elements in the scene. The console output can be seen at the bottom while on the right side, there is a text editor where code can be adapted directly. We integrated Webots in our ROS code so it sends and receives the same messages as the physical robot.

3.7 KUKA youBot software: Point cloud library

To detect an object, we used Point Cloud Library (PCL) [49]. PCL is an open-source library of algorithms for point cloud processing tasks and 3D geometry processing, such as occur in three-dimensional computer vision. The library contains algorithms for feature estimation, surface reconstruction, registration, model fitting, and segmentation. It is written in C++ and released under the BSD license. These algorithms have been used, for example, for perception in robotics to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them.

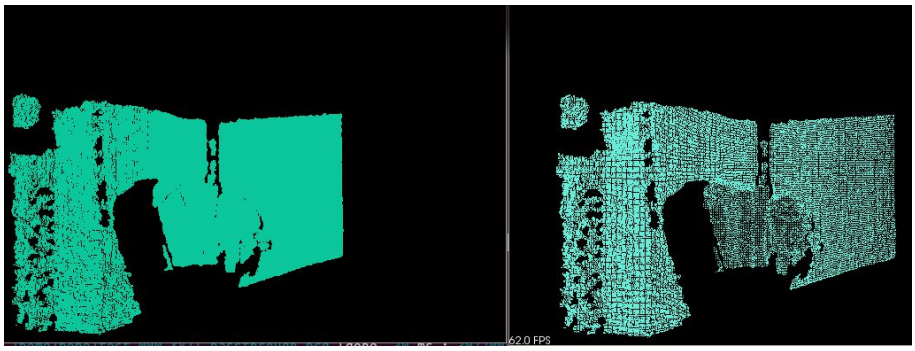


Figure 3.11: Example of PCL filter by using 3D sensor kinect

PCL was originally developed by Willow Garage as a package for ROS; however, its utility as a standalone library quickly became apparent. It is now a separate project maintained by the Open Perception Foundation, and is funded by many large organizations. The PCL is split into multiple libraries which can be compiled and used separately. These include libraries and support for: filtering, feature finding, key point finding, registration, kd-tree representation, octree representation, image segmentation, sample consensus, ranged images, file system I/O and visualization. This project relies on reading and writing point clouds to files, point cloud visualization, down-sampling point clouds using a filter, plane segmentation, and object segmentation using Euclidean Cluster Extraction. Euclidean Cluster Extraction works by separating the points into groups where each member of a group is within a specified distance of at least one other member of the group. Figure 3.11 shows the result PCL filter. Note how the top of the table and floor have been removed and that different colored clouds represent separate clusters. PCL also provides tutorials and examples for most of its features allowing easy implementation and modification of the

PCL algorithms. ROS package is also provided to allow convenient use of the PCL in a ROS node. This package provides functions for converting between ROS and PCL point cloud types and many other features [49].

Chapter 4

Basic Methodologies

This chapter presents the basic methodologies. By using grasping plan and inverse kinematics, the youBot can pick up, place or move some objects in order to solve some tasks automatically.

4.1 Arm navigation and grasp planning

One of the early major goals of our project was to get the newly installed arm on the youBot moving. While the low level controls, such as direct joint control, existed, there was no inverse kinematic software for the arm. We took advantage of the ROS *arm-navigation* open source stack in order to create several levels of inverse kinematic calculators. Included with the *arm-navigation* stack [50], these kinematic tools varied from simply avoiding self-collision, to avoiding other detected obstacles in the environment. We decided to take advantage of the existing manipulation code developed within ROS, and implemented the *arm-navigation* stack. This stack was designed to generate the configuration files for the manipulator motion from the *.urdf* robot description file. A configuration wizard is included as a part of this stack, which allows for the selection of the joints that compose the arm. A sample screen-shot of the configuration wizard is displayed below, as run on the youBot. By selecting the links that make up the arm, the configuration wizard uses the physical robot description provided to generate the files it needs to calculate the forward and *inverse kinematics* for that arm.

Once the arm's configuration had been generated, we had access to several useful kinematic services. The most straightforward of these was constraint aware *inverse kinematics*, which allowed us to specify a goal pose, and returned a trajectory of all of the joints to reach that location. This service also avoids

any path that would cause the arm to collide with itself. This was used in several places as a quick but robust method to generate arm joint trajectories. A level higher than that is the move arm package, which actually takes the physical environment into account as well as self-collision when planning the joint trajectory. This is the most robust kinematic solver included in the *arm-navigation* stack, but also has the highest computational time among the movement options. Like the constraint aware *inverse kinematics*, a goal pose is specified as part of a *MoveArmAction*, and a set of joint trajectories is returned. In fact, the move arm package uses the constraint aware *inverse kinematics*, after adding constraints that signify the environmental obstacles. This particular kinematic solver is used heavily in the pickup and place actions, to position the arm in the pre-grasp or pre-place pose.

Both of these kinematics solvers generate joint trajectories, an array of positions, velocities, and accelerations for all links in the arm to reach the desired location. To use this properly, the joint trajectories had to be sent to a joint trajectory action server, which essentially steps through the trajectory to move the arm properly. This ensures that the arm follows the path that the *inverse kinematics* solvers produced as closely as possible, as simply setting the joints to their final configuration might cause them to collide with other links or environmental obstacles. Thus, a good joint trajectory action server is required to really take advantage of the features of *youbot-arm-navigation*. Once the arm configuration wizard was run and the *arm-navigation* code was able to be run, the system was functional, but somewhat buggy. A big chunk of the early issues stemmed from overly tight constraints on the orientation of the desired goal position.

As the youBot arm is a 5 degrees of freedom arm, yaw is constrained by the x and y position of the goal, rather than being able to be set, as the *arm-navigation* code initially assumed. Once we wrote the code to calculate the yaw based on the desired position, the arm was able to reach all poses around the base. Arm navigation requires very few other services to run properly, as it is intended to be a low-level calculation tool. Specifically, it requires an *urdf* model of the robot in order to run the configuration wizard on. This gives the arm navigation stack the physical data required to calculate the joint limits and positions. As the robot configuration that the *urdf* file contains is the one that the arm's *inverse kinematics* will be calculated with, this model should be updated and not changed. If it does change, then the configuration wizard has to be run again, as we found out when we added the Kinect to the youBot arm. This resulted in bumping the Kinect against the robot frame until we ran

the wizard on the updated *urdf* file. Once the arm navigation stack was implemented, actually calculating the paths to any given goal does not require any outside components. However, actually transitioning the trajectory into motion requires a trajectory action server, as mentioned above.

One of the primary goals of our project was giving the youBot the ability to pick up and place objects. The ROS object manipulation stack provides a framework for pick and place actions, including standardized message types for describing the object you want to pick up or a grasp to be performed. We had to add several robot-specific components to get the object manipulation pipeline to work with the youBot, including a grasp planner and our implementation of the arm navigation stack.

The standard gripper on the youBot consists of two parallel fingers with a range of 2.3 cm. There are three mounting points for the gripper's fingers which allow it to pick up a larger variety of objects. The first gripper position allows the gripper to close fully, with a range of 0 cm to 2.3 cm. This allows the youBot to grasp objects that are less than 2.3 cm wide, such as an Expo marker. This position also makes it possible to grasp objects such as cups or bowls by grasping the rim of the object. The second gripper position gives the gripper a range of 2 cm to 4.3 cm. This allows the youBot to grasp most of the blocks and cylindrical objects in our environment as well as objects such as books when placed on end. This position has a larger variety of objects that can be grasped using standard external grasps, but no longer allows cups and bowls to be picked up by the rim. The third gripper position gives the gripper a range of 4 cm to 6.3 cm. This allows some larger objects to be grasped, but in this position the fingers are only connected with one screw, leaving them unstable which could result in inconsistent grasps. Figure 4.1 shows an example of the graspable object: Grasp a pen.

We decided to use the second gripper position for the youBot's gripper. This position gives us more options for objects to pick up than the other two, and does not have the stability problems of the third position. The first position would limit the graspable objects to very small objects and objects with rims. Grasping the rim of an object makes grasp planning more difficult, especially when using only a partial point cloud cluster to represent the object. Very small objects are also more difficult to find using object detection. We found that the second position gave us a good variety of objects that the youBot would be able to grasp, and also eliminated the problem of planning grasps for the rim of an object.

Figure 4.1: Picture of the graspable object: grasp a pen

The object manipulation pipeline supports using two different types of grasp planners. If the model of the object you want to pick up is known, a set of good grasps for the object can be precomputed and stored in a database, using software such as the GraspIt! simulator. A database grasp planner can then simply find the model in the database that matches the target object and use the precomputed grasps. However, this requires that every object in the robot's environment has an accurate 3D model that grasps can be precomputed against. It also requires accurate object recognition to choose the correct model that corresponds to the object you want to pick up.

The second type of grasp planner plans grasps based on the point cloud cluster that represents the target object. This type of cluster grasp planner is more flexible than a database grasp planner since it does not require a 3D model of the object to be picked up or object recognition. The downside is that the point cloud cluster for an object as seen by the robot's sensor is usually an incomplete representation of the object, which results in grasps that are generally less effective than the optimal grasps precomputed for the object's model. For the youBot, since we did not have accurate models of the objects in our environment and did not have object recognition, we designed a cluster grasp planner. Even if we had object recognition and an effective database grasp planner, the cluster grasp planner can still be used when object recognition fails or for objects that

are not in the database.

The process of placing an object is very similar to picking up an object, only in reverse. To specify a place action we need to know the grasp that was used to pick up the object relative to the object's coordinate frame, and the location to place the object relative to the robot. If multiple place locations are specified, the first one that is feasible will be attempted. We created an object place location server that makes it easier to specify a place action. This server subscribes to the pickup action goal and result messages from the object pickup. The pickup action goal message includes information about the object that was picked up, while the pickup action result message includes the grasp that was used and whether or not the grasp was successful.

The grasp pose relative to the object is calculated by multiplying the inverse of the original object pose relative to the robot from the pickup action goal message with the grasp pose relative to the robot from the pickup action result message, using functions from the tf package. This transformation is combined with the desired object pose to find the place location relative to the robot's base frame. Our object place location server also calculates the height to place the object. When the server receives the pickup goal message for an object, it finds the support surface for the object in the collision environment, and uses it to calculate the object's height above the surface. When an object is placed, its height is set to the height of the new support surface, plus the object offset above the previous surface calculated earlier, plus a small offset so the object will be dropped slightly above the surface. If a user specifies the height of the object in their call to the place location server, that object height will be used to place the object.

When placing an object that was picked up with an angled grasp, we need to constrain the place angle to align it with the first joint of the arm like we did when picking up the object. When placing with angled grasps, the transformation for the grasp relative to the object is rotated to counter to yaw of the original grasp. We then calculate the yaw of each place location using the desired x and y coordinates relative to the first joint of the arm. This creates a place location that is reachable by the 5 DOF arm when the two transformations are combined.

4.2 Inverse kinematics

To solve some tasks and problems with the KUKA youBot, we used *inverse kinematics* solution to find parameters for the youBot approach to objects.

In robotic, kinematics is the description of motion without regard to the forces that cause it. It deals with the study of position, velocity, acceleration, and higher derivatives of the position variables. The kinematics solutions of any robot manipulator are divided into two solutions, the first one is the solution of *forward kinematics*, and the second one is the *inverse kinematics* solution. Forward kinematics will determine where the robot's manipulator hand will be if all joints are known. Where the *inverse kinematics* will calculate what each joint variable must be if the desired position and orientation of end-effector is determined. Hence, Forward kinematics is defined as transformation from joint space to Cartesian space where as Inverse kinematics is defined as transformation from Cartesian space to joint space [51].

The solution of inverse kinematic is more complex than direct kinematics and there is not any global analytical solution method. Each manipulator needs a particular method considering the system structure and restrictions. There are two approaches namely, geometric and algebraic used for deriving the *inverse kinematics* solution. We used geometric approach to calculate each joint variables. The youBot has 4 links numbered from 0 to 3 starting from the base of the robot, which is taken as link 0. The joints are numbered from 1 to 5, and z_i is a unit vector along the axis in space about which the links $i-1$ and i are connected. The $i-th$ joint variable is denoted by q, In the case of a revolute joint, q, is the angle of rotation, while in the case of a prismatic joint q, is the joint translation. Next, a coordinate frame is attached rigidly to each link. To be specific, we choose frames 1 through 5 such that the frame i is rigidly attached to link i. Figure 4.2a illustrates the idea of attaching frames rigidly to links.

youBot has five rotational joints and a moving grip as shown in Figure 4.2b. Joint 1st represents the shoulder and its axis of motion is z_1. This joint provides a rotational θ_1 angular motion around z_1 axis in x_1y_1 plane. Joint 2nd and its axis is perpendicular to Joint 1st axis. It provides a rotational θ_2 angular motion around z_2 axis in x_2y_2 plane. z_3 axes of Joint 3rd and Joint 4th are parallel to Joint 2nd $z-axis$; they provide θ_3 and θ_4 angular motions in x_3y_3 and x_4y_4 planes respectively. Joint 5th are identified as the grip rotation. Its z_5 axis is vertical to z_4 axis and it provides θ_5 angular motions in x_4y_4 plane [23]. A graphical view of all the joints was displayed in Figure 4.2b. Using *inverse*

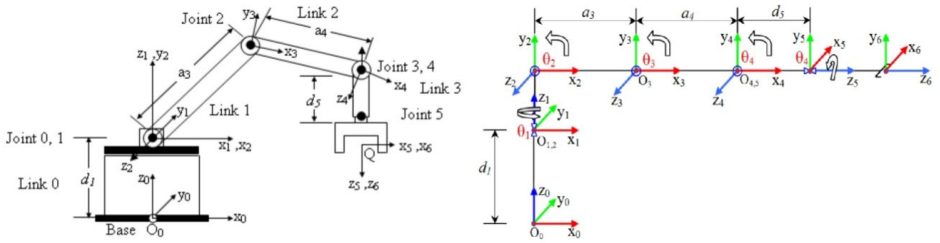


Figure 4.2: Coordinate frames assessment of the youBot

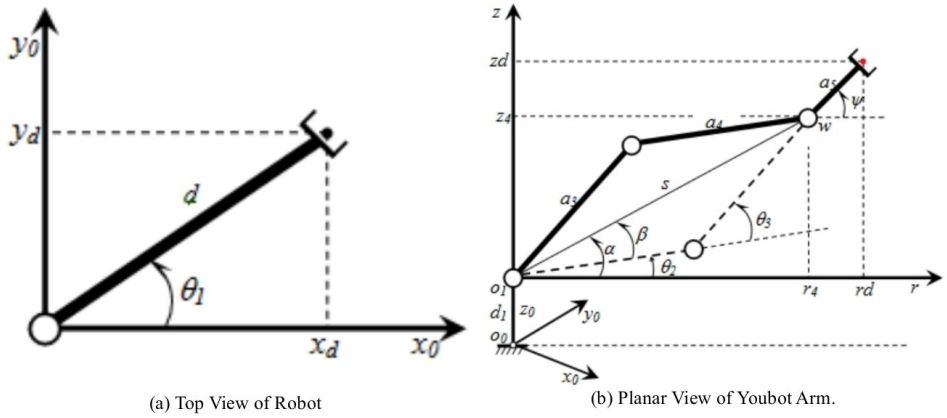


Figure 4.3: 2D coordinate frame of youbot

kinematic Cartesian mode, we specifies the desired target position of the gripper in Cartesian space as (x, y, z) where z is the height, and the angle of the gripper relative to ground, ψ (see Figure 4.3), is held constant. This constant ψ allows us to move objects without changing the object's orientation (the holding a cup of liquid scenario). In addition, by either keeping ψ fixed in position mode or keeping the Joint 4 fixed relative to the rest of the arm, the inverse kinematic equations can be solved in closed form as we now show for the case of a fixed ψ.

The lengths d_1, a_3, a_4 and a_5 correspond to the base height, link 1 length, link 2 length and gripper length, respectively are constant. The angles θ_1, θ_2, θ_3, θ_4 and θ_5 correspond to joint 1 rotation, joint 2, joint 3, joint 4, and end effector (joint 5), respectively. These angles are updated as the specified position in space changes. We solve for the joint angles of the arm, $\theta_1 : 4$ given desired position *(x,y,z)* and ψ which are inserted by the us. From Figure 4.3a, we clearly see that $\theta_1 = artanh(y, x)$ and the specified radial distance from the base d are

related to x and y by:

$$d = \sqrt{x^2 + y^2} \;\; and \;\; x_d = d\cos(\theta_1) \;\; and \;\; y_d = d\sin(\theta_1) \tag{4.1}$$

Moving now to the planar view in Figure 4.3b, we find a relationship between joint angles $\theta_2, \theta_3, \theta_4, \psi$:

$$\psi = \theta_2 + \theta_3 + \theta_4 \tag{4.2}$$

Since ψ is given, we can calculate the radial distance and height of the wrist joint:

$$r_4 = r_d - a_5 cos(\psi) \tag{4.3}$$

$$z_4 = z_5 - a_5 sin(\psi) \tag{4.4}$$

Or

$$r_4 = a_3 cos(\theta_2) + a_4 cos(\theta_2 + \theta_3) \tag{4.5}$$

$$z_4 = a_3 sin(\theta_2) + a_4 sin(\theta_2 + \theta_3) + d_1 \tag{4.6}$$

Now we want to determine θ_2 and θ_3. We first solve for α, β and s (from Figure 4.3b) uses the law of cosines as:

$$\beta = atan2(s^2 + a_3^2 - a_4^2, 2a_3 s) \tag{4.7}$$

$$\alpha = atan2(z_4 - d_1, r_4) \tag{4.8}$$

$$s = \sqrt{(z_4 - d_1)^2 + r_4^2} \tag{4.9}$$

With these intermediate values, we can now find the remaining angle values as:

$$\theta_2 = \alpha \pm \beta \tag{4.10}$$

$$\theta_3 = atan2(s^2 - a_3^2 - a_4^2, 2a_3 a_4) \tag{4.11}$$

$$\theta_4 = \psi - \theta_2 - \theta_3 \tag{4.12}$$

4.3 Control the youBot by some ROS stacks

To provide control over the youBot's integrated 5-DOF arm, we focused on two ROS stacks: The Arm Navigation stack and the Object Manipulation stack. Originally developed for the PR2, these stacks comprise most of what is called the object manipulation pipeline. This pipeline provides a robot-independent set of interfaces, message types, and tools to help one implement common object

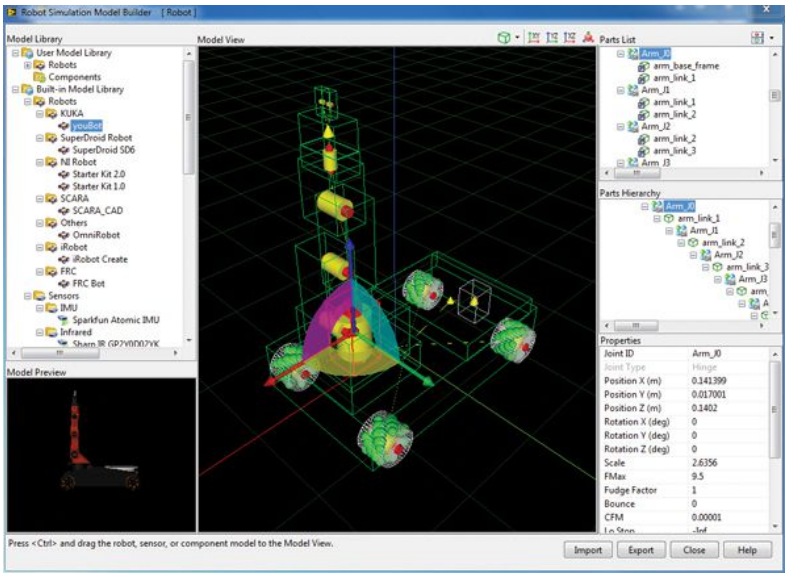


Figure 4.4: Example of arm navigation stack by simulators

manipulation actions for a particular robot. Each of the aforementioned stacks will be introduced in the sections as follow.

The arm navigation stack [50] was developed by Willow Garage to provide for the collision-free motion of a multiple degree of freedom robot arm. While only implemented originally for the PR2 robot arm, the stack was designed in such a way that it could be used for any arm, with the proper setup. Once that setup is complete, the stack handles collision avoidance, *inverse kinematics*, and publishes status updates on the arm's progress. In order to move the arm to a given position and orientation, only a relatively simple message is required to set the arm's goal. Once that is received, the stack plans a collision-avoiding route to the target location, and produces a set of joint positions to create a smooth path to the destination. Oddly enough, the arm navigation stack did not provide anything to actually run through that path. Despite this, we chose to use this stack for the built-in features, the relative ease to set up, and the support for further arm tasks. As previously mentioned, the motion path that is created takes collisions, both with the arm itself and objects discovered in the environment, into account. That would be very difficult to program in a timely manner, sparing us significant development time.

There were only two major sections of the program that had to be created for the arm navigation stack to operate properly: the above mentioned program to

take the path and execute it, and a description of the arm to be used. The first was easy to create, and the second was generated based on the robot model, which made it simple to implement once the requirements were understood. Finally, the arm navigation stack is used by the PR2 to feed directly into picking up an object with one of its arms, so if we wished to also leverage that code, it would be of a great benefit to follow the same process. In fact, arm navigation actually contains both of the kinematics models used by the object manipulation stack below. We looked into a few other possible kinematics models and arm controllers, but none of them provided the level of functionality or support that the arm navigation stack did. The official KUKA youBot arm manipulation software was designed for a previous version of ROS, and had not been updated, in addition to not containing collision avoidance capabilities. Compared to all other options, the arm navigation stack provided the most useful features and the easiest implementation.

The Object Manipulation stacks [52] The object manipulation stack provides the framework for picking up and placing objects using ROS as show in Figure 4.5. The stack is designed to be fairly robot independent, but requires some robot specific components to work properly. These robot specific components include a grasp planner, a gripper posture controller, an arm/hand description configuration file, and a fully implemented arm navigation pipeline. The object manipulation pipeline is fully implemented for the PR2 and some other robots, but not for the youBot. The object manipulation pipeline was designed to work either with or without object recognition. For unknown objects, the grasp planner must select grasp points based only on the point cluster perceived by the robot's sensors. If object recognition is used, grasps for each item in the object database are pre-computed, and should be more reliable than grasps planned based only on a point cluster. There are two types of motion planners used by the object manipulation pipeline.

The standard arm navigation motion planner is used to plan collision free paths. However, this motion planner cannot be used for the final approach to the grasp point since it will most likely "think" the grasp point will be in collision with the object being picked up. For the final grasp approach, an interpolated *inverse kinematics* motion planner is used, which will move the gripper linearly from the pre-grasp point to the final grasp point. The object manipulation pipeline also has the option of using tactile feedback both during the approach to the grasp point and during the lift. This is especially useful to correct errors when executing a grasp that was planned based only on a partial point cluster. Unfortunately, the youBot does not have tactile sensors on its gripper. Picking up an object using the object manipulation pipeline goes through the following

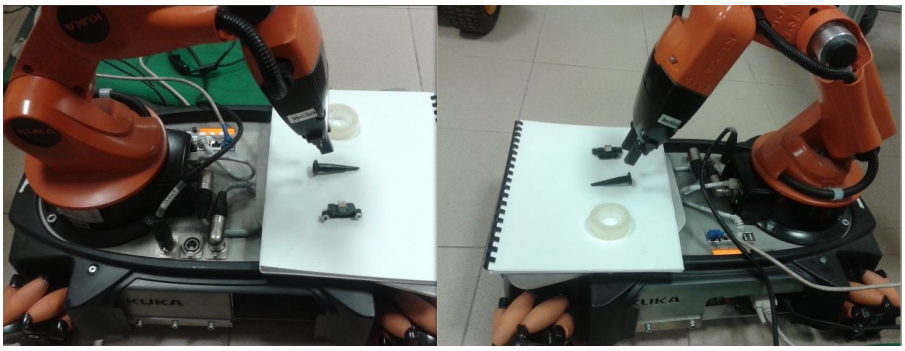


Figure 4.5: Example of object manipulation stacks to pick up and drop some objects

steps: (1) The object to be picked up is identified using sensor data (2) A grasp planner generates set of possible grasp points for the object Sensor data is used to build a collision map of the environment (3) A feasible grasp point with no collisions is selected from the list of possible grasps A collision-free path to the pre-grasp point is generated and executed The final path from the pre-grasp point to the grasp point is executed using an interpolated IK motion planner (4) The gripper is closed on the object and the object model is attached to the gripper (5) The object is lifted using the interpolated IK motion planner to a point where the collision free motion planner can take over.

Chapter 5

Developments and Implementations

In this chapter, we show our first main contribution involving In terms of robotic by experiments. We present our development and implementation when our robot can pick up, place, drop or move several small objects such as pens, rolls of tape or paper, a wooden block size 3x3x3(cm), 3x6x3(cm), 3x9x3(cm), 3x12x3(cm), 3x15x3(cm), etc. It can also push the button as turning on/off the lights, push the little wheel or door. Moreover, the robot can solve automatically some basic problems as "writing", "cleaner", etc and some physic and informatics problems, for example, "Elastic Collision", "Tower of Hanoi" problem, "Sorting" problem, etc.

5.1 Graspable objects

We tested the youBot's ability to pick up the objects as in Figure 5.1. For this test, we placed each object centered approximately 15 cm in front of the youBot, within overhead grasping range. Table 5.1 shows the success rates for each of the objects tested.

All of the objects chosen has sizes within the gripper's range. Some objects, including the expo eraser and book, are only graspable when in certain orientations, since the dimension of some of their edges is too large for the youBot's gripper. If the robot is being controlled remotely, and one of these objects falls into an orientation that is not graspable, the user will not be able to pick up the object or move it to a position that is graspable. There are 4 sizes of wood cube: 3x3x3(cm), 3x6x3(cm), 3x9x3(cm), 3x12x3(cm) indicate by 4 disks in Tower of Hanoi problem (See Figure 5.5a). The average success rate for the

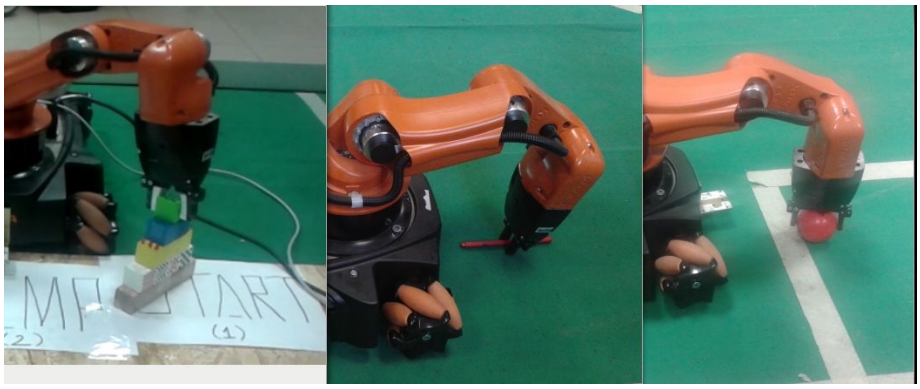


Figure 5.1: YouBot pick up some objects

Table 5.1: The success rates for each of the objects tested

Object	Attempts	Successful	Success %
Ball	10	5	50%
Pen	10	6	60%
Cube 3x3x3cm	10	8	80%
Cube 3x6x3cm	10	7	70%
Cube 3x9x3cm	10	6	60%
Cube 3x12x3cm	10	6	60%

objects tested, was 63%.

We tested the workspace where the youBot can successfully pick up objects off the floor, and compared the workspace when using only overhead grasps to the workspace when angled grasps were allowed. For these tests we used the red cylindrical block, which is graspable by both overhead and angled gasps. The workspace for picking up objects is circular, centered at the first joint of the arm. Using overhead grasps, the block can be picked up at a distance of up to 31.5 cm from the first joint of the arm. This corresponds to 18 cm from the edge of the robot if the object is centered in front of the youBot.

Next, we tested the workspace for picking up the same block using angled grasps. We found that the block could be picked up with angled grasps at a distance of up to 42.3 cm from the first joint of the arm. For objects centered in front of the robot, angled grasps increase the pickup range to 28.8 cm from the front of the robot, which is a 60% increase over using only overhead grasps. Objects that are not cylindrical are only graspable using overhead grasps, limiting the maximum range that they can be picked up from floor.

The minimum range for picking up objects depends on several factors, making it not well defined. For an object to be picked up, it needs to be visible by the Kinect and must be recognized by the object detection. If the object is too close to the robot, it may not be successfully recognized or may be combined with part of the robot by the object detection. The grasp must also be reachable without collision for a grasp to be attempted. The Kinect, which is mounted to the last link of the arm, would be in collision with the robot body during many of the grasps for objects very close to the robot.

On the floor, the youBot can consistently place objects that were successfully picked up as long as the specified place location is reachable by the arm. Objects are released about 2 mm above the floor, which avoids collisions with the floor before releasing but is low enough so the block falls into the desired location without bouncing. The orientation for the object to be placed can be specified if the object was picked up with an overhead grasp. For objects picked up with angled grasps, the place orientation cannot be specified.

We could not get placement on tables to work correctly. Since the table is at the edge of the arms workspace, we were unable to find a place location that was reachable by the arm. Placement may be more effective on a lower table than the 29 cm table we used. Table placement may also be more effective if we removed the constraint that the object must be placed at the same orientation as it was picked up in. This would allow the object to be placed at an angle that is reachable by the gripper, but would place the object in an unstable orientation that would fall into an unpredictable position. Even though table placement does not work, objects picked up off a table can be placed in a different location on the floor.

We measured the accuracy of the object detection code by placing a few of our objects at set positions and then recording the distance the Kinect detected them at. The first set of positions was a row two meters away from the robot, one directly in front of the robot and the other 2.75 meters on either side.

The second set of positions was similar to the first but only one meter away with the other two positions only half a meter away on either side. We put the positions on either side closed for the short distance because the Kinects field of vision narrows closer to the robot.

For navigation, the goal of our experiment was to provide accurate and reliable autonomous driving throughout the workspace. To quantify this we took

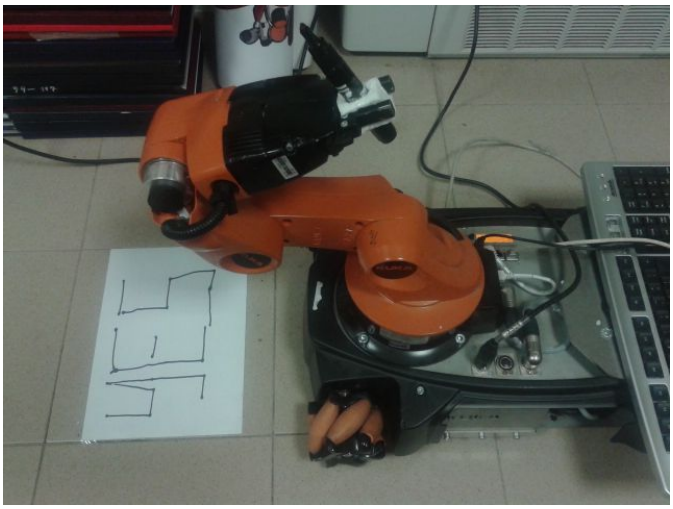


Figure 5.2: youBot pick up a pen and Writing a word

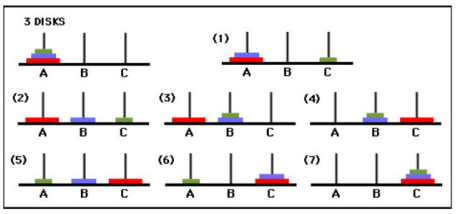
two statistics, in the first we told the robot to drive one meter straight from a fixed point and recorded the actual distance driven. The second experiment was similar but instead of one fixed point we used locations throughout the room.

Figure 5.2 show an example when youBot can pick up a pen and writing a word.

The Average accuracy of the fixed one meter drive was 0.99 meters; however this is not the most telling statistic as the robot went both over and under shot the goal point. Far more informative is the standard deviation which was 0.03 cm. This tells that around 95% of the time the robot will be within 6cm of the goal point when driving in a single direction. This is a reasonable accuracy. The average when driving from different points was 0.96 and the standard deviation was 0.017. While the standard deviation was lower it should be noted that is because the actual drive distances were clustered at slightly less than a meter. The overall result is that the navigation is reasonably accurate and works similarly well throughout the room.

We also measured how well the navigation and arm worked with each other and the object detection with two tests. In the first we put the robot at a point and the object at another fixed point then instructed the robot to detect and move to the object. After the robot moved we measured the distance from the arm to the object and if the arm was able to pick up the object. The other test was similar but the robot and objects position were varied to test if the results

(a) Wooden-toy of "Tower of Hanoi"



(b) Solution with 3disks

Figure 5.3: The Tower of Hanoi problem.

were similar throughout the workspace.

In the first test the average resulting distance from the object was 27 cm and the standard deviation was 7 cm. Only one move out of five failed to reach a position where it could grab the object. That move resulted in a distance of 40 cm from the object. In the second test the average distance was 26 cm and the standard deviation was 8 cm. Similar to the first test only one of five failed to reach a position where it could grab the object. In that case it ended 41 cm away from the object. It is important to note that the goal for this test was not to reach exactly zero centimeters away as this would mean the object was under the base of the robot. Instead around 20 to 30 centimeters was an ideal range.

5.2 "Tower of Hanoi" problem for computer scientists

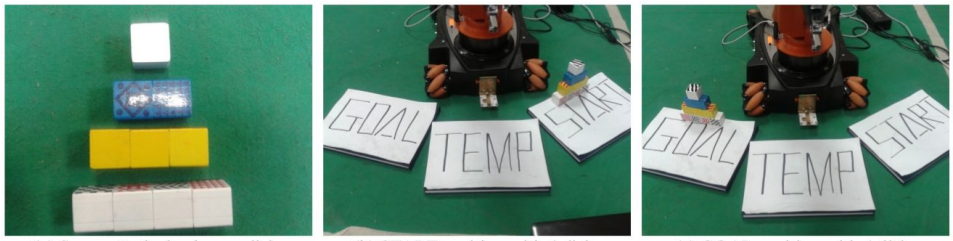
The Towers of Hanoi (see Figure 5.3a) is a problem that has been known to generations of computer science students. Nice example to teach recursion, experimental series, etc. The task is to move a pile of disks with the smallest amount of moves from one location to another. While solving the task the following two rules have to be obeyed:

1. Only one disk must be moved at a time;
2. A disk with a smaller diameter must be placed on top of a disk with a larger diameter.

Figure 5.3b show an example of solution with 3 disks. The second rule enforces that the disks at all three locations are sorted from the bottom to the top with a decreasing diameter. To make the problem solvable without violating the rules, a third auxiliary location is introduced at which disks might be temporarily deposited. In principle, the disks can be arbitrarily moved back and forth

(a) Connecting, speech, rotation, stop-based interface (b) Joints interface (c) Some fix-positions interface

Figure 5.4: Control the youBot by the mobile platform of the KUKA youBot via Wifi



(a) Some attached cubes as disks (b) START position with 4 disks (c) GOAL position with 4 disks

Figure 5.5: Goal, Temp and Start position with 4 disks

between the three locations (Start, Goal, Temp). An algorithm, which works on a trial and error principle and arbitrarily moves disks back and forth may of course lead to solutions which are everything but optimal. Hence a smart strategy for planning the actions is required.

5.3 YouBot solves "Tower of Hanoi" problem

To let the youBot solves "Tower of Hanoi" problem, we used the youBot control software which you can control the mobile platform of the KUKA youBot via Wi-Fi given by XITASO. Using the acceleration sensor, the platform and joints moves in any direction and additionally you can steer the platform with buttons as Figure 5.4.

A possible set up for the Youbot solving "Tower of Hanoi" is presented in Figure 5.5. The workspace area will contain three distinguished locations related to the task: one will serve as a goal position to erect the final tower, the two others will serve as start and temp position. To simplify the problem of grasping difficult objects such as thin disks stacked on a rod we use cubes 3x3x3 (cm) instead. We use a color code to distinguish the objects and also to

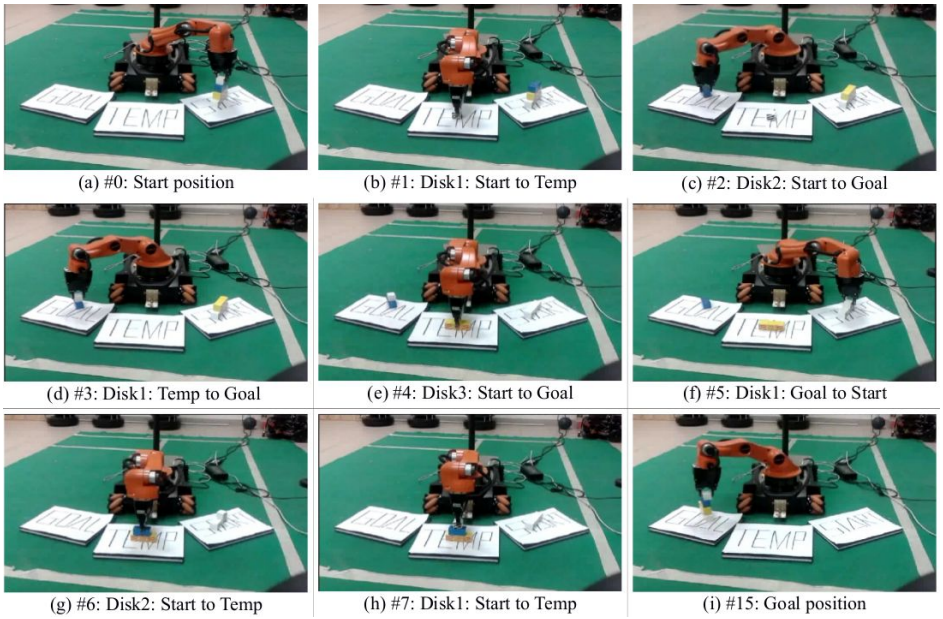


Figure 5.6: Some different positions step of solving with 4 disks by the youBot.

establish an order in which they must be stacked or be placed with respect to each other. We use three different colors: white, blue, yellow. In general, the same color cubes are attached to each other, hence, the youBot takes only one time to move the disk (Figure 5.5a). The task is to move the cubes from start position (Figure 5.5b) to the goal position (Figure 5.5c) and to stack them there in a certain order to form a tower in the fastest way possible. While doing so the robot has to obey certain rules:

1. The longer cubes must under the shorter one or the "color code" from the bottom to the top of a stack must be yellow $<$ blue $<$ white.

2. The robot never transports more than 2 objects at a time;

3. The task is accomplished once all objects are stacked in the goal position such that rule 1st is satisfied.

At the beginning of a run the objects are typically stacked at the start position obeying rule 1st. To solve "Tower of Hanoi", we focus on inverse kinematics solution to find parameters for the youBot approach to objects as section 4.2: Basis methodologies: Inverse Kinematics. Figure 5.6 and Figure 5.7 show some different positions of solving "Tower of Hanoi" with 4 disks and 5 disks respectively. We also show here the video on youtube when the youBot solves the "Tower of Hanoi" with 5 disks : https://www.youtube.com/watch?v=h404XDL2DH4

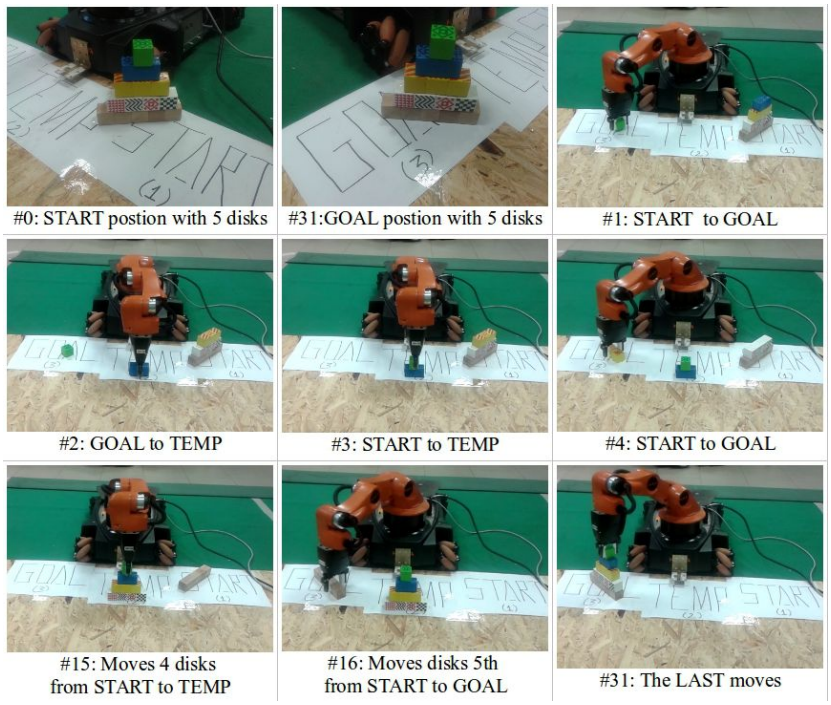


Figure 5.7: Some different positions step of solving with 5 disks by the youBot.

Using results of those equations and by apply programming to solve and configure, we can calculate the joints for each the position of the arms (see Figure 5.8). By modified the youBot controller with mobile platform as well, the youbot solved automatically "Tower of Hanoi" 3 disks, 4 disks,and even 5 disks with 7, 15 and 31 optimal moves respectively. Figure 5.8 show various value of Joints for 4 disks with 15 optimal moves solution in 120 seconds. If 5 disks with 31 optimal moves optimal solution, it takes 4.3 minutes.

5.4 Youbot solves "Elastic collision" problem

A collision is an interaction of two bodies during a short period of time, while momentum and energy are being exchanged. It is called an elastic collision if no energy in the production of heat, irreversible deformations, electronic excitations or other non-kinematic effects is used up. During the collision, there are complicated forces at work between the two bodies. The laws of conservation of momentum, energy, and (under certain circumstances) angular momentum allow a satisfying determination of the behavior of the interacting bodies after the collision. In this experiment, this will be illustrated in the elastic collision of two balls.

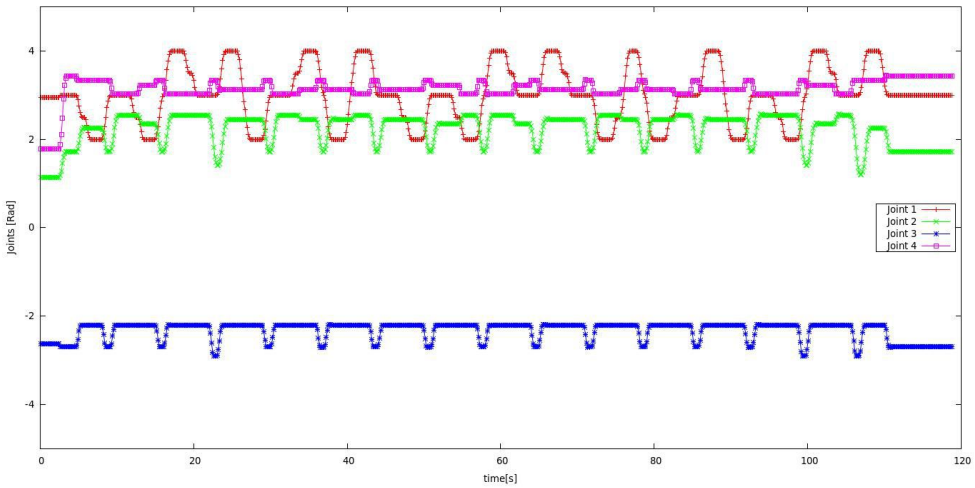


Figure 5.8: Various value of Joints for 4disks with 15 moves optimal solution conducted about 115 seconds.

(a) Elastic Collision experiment with youBot: Keep two balls vertically

(b) Elastic Collision experiment with youBot: Drop two balls vertically

Figure 5.9: Elastic collision experiment with the youBot: Keep and drop two balls vertically

In the experiment, the transmission dynamics of a ball, after placing 2 balls vertically, the youBot will drop 2 balls at the same time until touching the ground, the kinetic energy of the tennis ball will be passed on to the ping-pong ball, which can be transmitted to a ball at rest is determined as you can see in Figure 5.9

We really need the youBot in this experiment, because it can keep two balls with an emphasis on the vertical, and drop both balls at the same time, even we can hardly to do it with two hands.

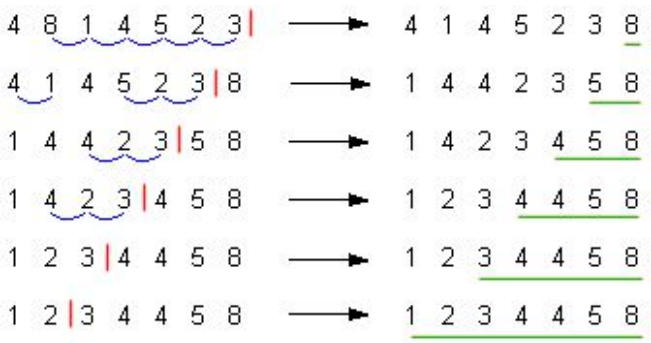


Figure 5.10: An example of sorting numbers by ascending order

5.5 Youbot solves "Sorting" problem

Sorting is the process of placing elements from a collection in some kind of order. For example, a list of numbers could be sorted by ascending order as in Figure 5.10.

Sorting an objects not only teaches children about attributes and relationships, but also promotes thinking logically and applying rules. Sorting exercises can also provide children with models for organizing things in the real world, such as putting blocks away or setting the table for dinner. First, children learn to compare the numbers, then categorize or exchange them.

Concept development is based on classification, so helping children develop this skill is an important responsibility of a teacher of young children. Learning theorists tell us that a large part of the cognitive development of young children is driven by classification and classification for kindergarten mathematics begins with ideas of making, describing, sorting and comparing sets. Teachers should therefore involve children in classroom exercises that require them to use a systematic classifying scheme to make comparisons among objects and encourage young learners to construct their own sets. Children learn to classify by focusing on attributes of included objects, then later focus on which objects were not included and why. For example, certain people belong to the set called family, and other people do not. Teachers can also practice sorting by moving objects into different groups based on observable characteristics such as size, shape, color, or number. By learning to follow rules that take into account the defining characteristics of sets, they are grasping concepts that form a basis for understanding mathematical functions.

There are many sorting algorithm such as Selection Sort, Insertion Sort, Bubble Sort, Merge Sort, Quick Sort, etc, but in reality, Bubble Sort is an

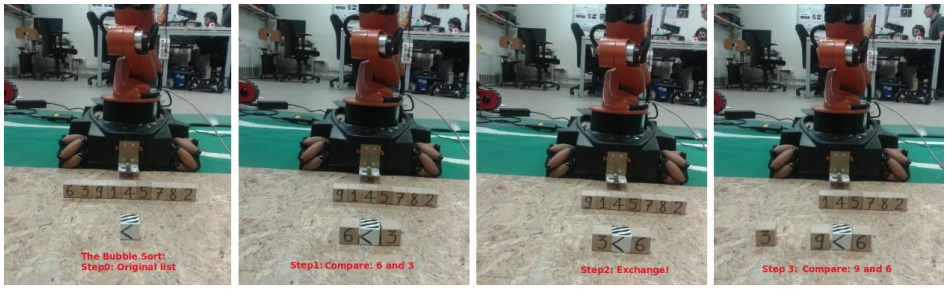


Figure 5.11: Some snapshots of bubble sort by the youBot

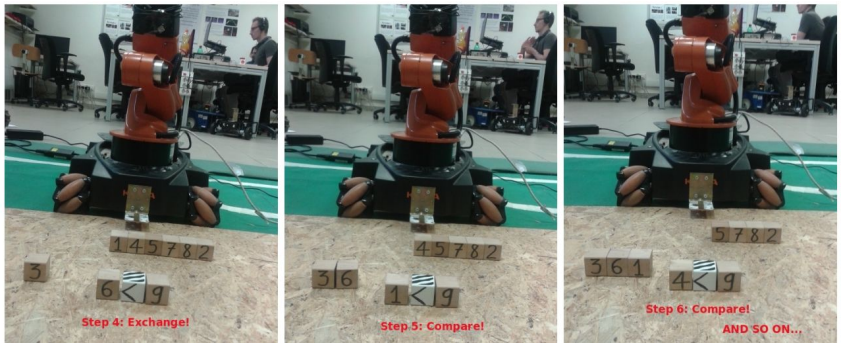


Figure 5.12: Some snapshots of bubble sort by the youBot(continue)

extremely simple algorithm the most distinctive feature of which is that this is most often incorrectly implemented sorting algorithms in existence. It is not only incorrectly implemented in popular web pages, it is also often implemented incorrectly in textbooks and animations. The bubble sort makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. We developed and implemented the Bubble Sort by the youBot as show in Figure 5.11 and Figure 5.12

Chapter 6

Experimental Evaluations

This chapter show our second main contribution involving robotic in education. We made several experiments when robots acts as the teaching assistance in the classes. Robot, beyond attracting the attention of students, also supports teachers in the teaching process as a very useful utility. We also design a lesson plan involving robots in the classes to help the teachers.

6.1 Training for the teachers

For the layer of the teacher who is able to use youBot as teaching assistance, we have one program guide and help the teacher uses youBot a quick and convenient way. It only takes about 2-3 hours of exposure and control the youBot, teachers will be able to use the youBot without additional technicians in the class. We also have one questionnaire for teachers on effective education and using youBot (See chapter 7).

6.2 Tower of Hanoi: Lesson plan overview

In terms of teaching, we designed the lesson plan overview of Tower of Hanoi problem in table 6.1.

6.3 Tower of Hanoi: Lesson plan involving youBot as the teaching assistance

The Tower of Hanoi experiment involved the children from 14 and 18 years old. Along with suggestions and feedbacks from Italian teachers, we have designed a particular lesson plan for the youBot to support teaching the "Tower

Table 6.1: Lesson plan of the Tower of Hanoi: Overview

Teaching assistance	KUKA youBot	8-12th grade (14 to 18 years old)
Mentor aeacher	*Name of teacher*	Mathematics/Informatics
School	*40*	2 groups (20 students/group)
Numbers students	*Name of school*	Date of Experiment
Class schedule	*120 minutes*	youBot, computer, cubes, timer clock
Knowledge, skills, and academic background: Students do not know about recursiveness and series. This is the first introduction to sequences and series. Students know how to work in groups and have been using problem-solving techniques.		
Lesson objectives (skills or concept): Students will be able to identify a pattern between two datasets, to understand how the robot works and how to control the robot. They will write an equation that describes a possible solution of the Tower of Hanoi problem inferred from practical experience. Students will be required to write the least number of moves and try to guess the relation between the least number of moves and the number of disks involved.		
Teachers know about this particular concept/topic: The teacher knows how to write a sequence, the rules for a sequence, how to solve a multi-step problem, the least number of possible moves, as well as the equation that represents the least number of possible moves. The teacher has played the game to discover that with n rings, there is a minimum number of moves of $2^n - 1$. The teacher knows how to control the youBot.		
Learning activity: After looking at the youBot playing, students will explore the tower of Hanoi playing the game, writing down the number of disks in relation to the least number of moves. They should attempt to write a function that describes the relationship between the two quantity.		

of Hanoi" problem as shown in table 6.2. We also show here the video on youtube when youBot become a teaching assistance: https://www.youtube.com/watch?v=vfjbsMHg8L0

Figure 6.1 also shown some photos of the lecture with the youBbot at the class.

Figure 6.1: Some photos of the lecture "Tower of Hanoi" with the youBbot at the class.

Table 6.2: Lesson plan of the Tower of Hanoi: Timetable

Time	Teacher/youBot's activities	Student's activities
00-05'	The teacher introduces the tower of Hanoi: The French mathematician, Edouard Lucas, invented it in 1883. Given a stack of 64 gold disks, each one a little smaller than the one beneath it. Their assignment was to transfer the 64 disks from one of the three poles to another; with one important provisional large disk could never be placed on top of a smaller one.	Students will be given real world application to the exploration that they are about to begin. This anticipatory set will provide motivation as well.
06-09'	The teacher tells the objective of the lesson: Students will be able to identify a pattern between two datasets, using the youBot to solve it.	The stated objective will give students an idea of what they are going to learn in class.
10-14'	The teacher uses the youBot to solve Tower of Hanoi with 3 disks. The transfer rule are: 1. Larger disk never upper smaller one. 2. Move only one disk at a time.	Students see the youBot playing Tower of Hanoi
15-54'	The teacher should assist students in robot control and during the game phase.	Students will be given 30 minutes in order to know how the robot works, how to control and explore the problem settled in front of them. Students will be given 10 more minutes to play the game without the youBot.
55-109'	50 minutes will be given to play the game with the youBot. Teacher will be monitoring the class, answering questions when applicable and asking questions to spark students interest. If students finish early, the teacher can increase the number of disks.	In this phase students should write the least number of steps and they should try to infer which is the relation between the disk and the least number of moves.
110-120'	The teacher distributes a questionnaire about the experience and about the topic learned.	Students will be asked to compile a questionnaire.

Chapter 7

Questionnaires and Results

In this chapter, we present our third main contribution with the results. We made questionnaires in a case of study for lectures with and without using robot. To clearly recognize the effect of using robots in the class, we designed the questionnaires and clustering the group study with and without robot to perform statistic and experiment. Based on this study, we also made several observations about factors that we believe contributed to the success of the education. Finally, we present some results in terms of robotic and some results from questionnaire study conducted with the robot and the comparison of the two groups is made to validation. As a primal results, the using of the robot as the teaching assistance has improved the percentage of correct answers of roughly 15%, level of attention of roughly 12% and the comprehension of roughly 11.2%.

7.1 Questionnaires

We invited two classes coming from two deferent high schools of Rome with different skills. We first mixed two classes, then we divided in two groups randomly by name alphabet to attend two deferent lessons: informatics as Hanoi Tower and physics as Elastic Collision with and without the youBot as show in Figure 7.1. In this way both groups of students attended a lesson with the youBot and a lesson without it.

At the end of the lesson we asked the students to compile a questionnaire that was divided in three different sections: a first part about the liking of the experience itself using the youBot (increasing motivation, easiness of the topic, etc), a second part that aimed to understand the level of attention during the lesson (questions about rules, the game inventor, etc) and finally a third part about the comprehension of the topic (the recursive problem, the exponential solution, etc).

Experiments	Sturtents	Hanoi Tower	Elastic Collision
GROUP A	19	Study without YouBot	Study with YouBot
GROUP B	20	Study with YouBot	Study without YouBot

Figure 7.1: Experimental configuration with two groups of students.

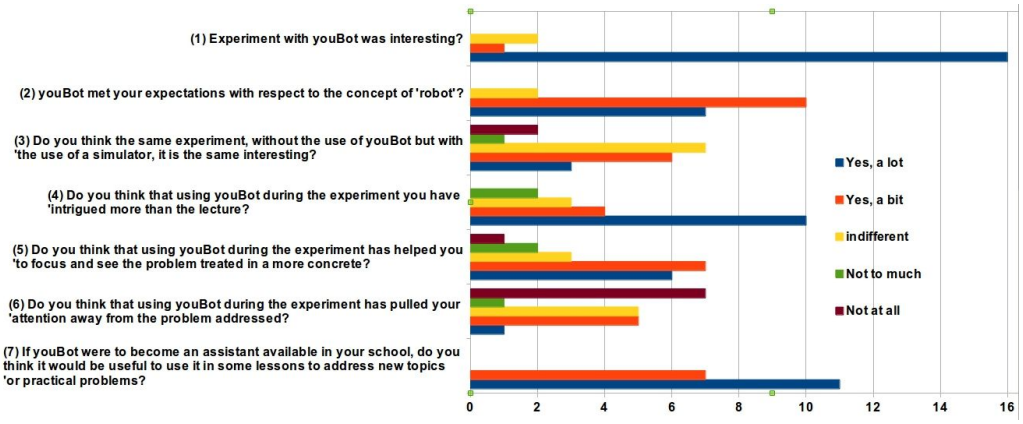


Figure 7.2: Results of questionnaire liking part for group A that studied Hanoi Tower without the youBot but studied elastic collision with the youBot.

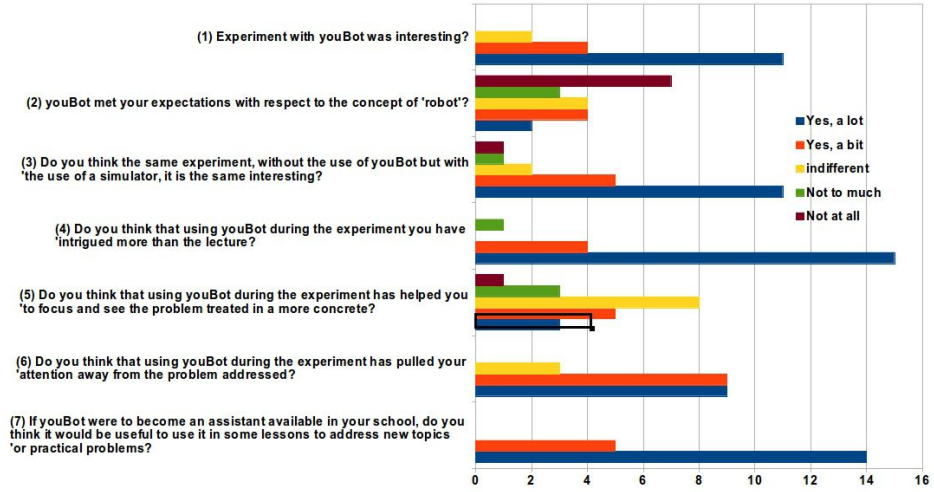


Figure 7.3: Results of questionnaire liking part for group B that studied Hanoi Tower with the youBot but studied elastic collision without the youBot.

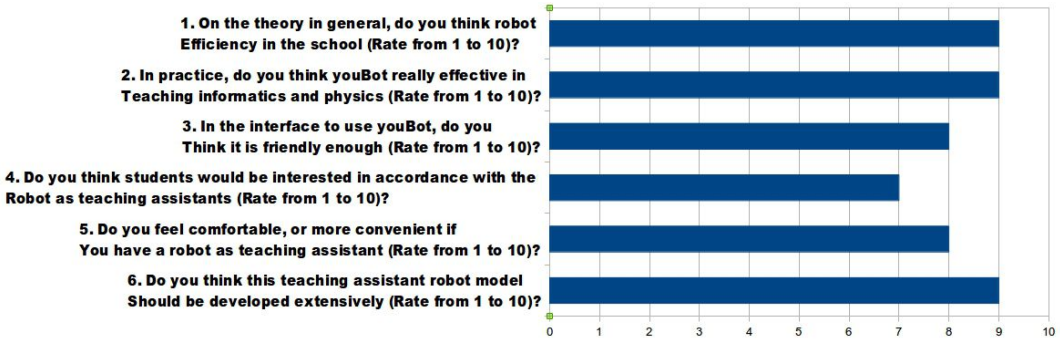


Figure 7.4: A questionnaire for teachers on effective education and using youBot.

The questionnaire where dispensed to two groups of students that followed the lesson: Group A has 19 students that follow the Hanoi lesson without the youBot and Group B has 20 students that followed the same lesson with the youBot as the teaching assistance. Otherwise, Group B follow the Elastic Collision lesson with the youBot as the teaching assistance and Group A followed the same lesson without the youBot.

For the layer of the teacher who is able to use youBot as teaching assistances, we have one program guide and help the teacher uses youBot a quick and convenient way. It only takes about 2-3 hours of exposure and control the youBot, teachers will be able to use the youBot without additional technicians in the class. The following steps can help teachers use youBot:
Step 1: Connect the laptop and youBot via LAN network.
Step 2: Connect the phone and laptop with the same Wifi.
Step 3: Run the program on the laptop.
Step 4: Control youBot by the interface on the phone (see Figure 5.4).

We also have one questionnaire for teachers on effective education and using youBot (See Figure 7.4). We asked 6 questions for the teacher to talk about the impact of the use of robots in schools. In general, teachers have very good reviews from this proposal. Specifically, 90% of them support the robot in the classes, 90% were rated good for this proposal, 80% of them feel friendly to teach with robot. However, the interface is not so reasonable, somewhat it quite hard to use (70%).

7.2 Results

We developed and implemented a system to allow for arm movement, navigation, and object manipulation for the youBot platform. To do this, we took ad-

vantage of existing ROS architecture that dealt with manipulation, and tailored it to the youBot's physical setup. By using these open source architectures, we were able to reduce our development time and focus on fine-tuning the robot's performance.

The arm movement code is robust, taking into account both self-collision and environmental hazards to avoid damaging the arm. The system has also been optimized towards the youBot's specific manipulator, by constraining the system to be consistent with the arm's own 5 degrees of freedom constraints. Several methods of controlling the arm were implemented, from a simple prompt for a desired pose, to a virtual interactive marker, to an automated system that moves the arm towards detected objects.

Finally, a set of often-used arm configurations was created, to allow for easy movement of the arm to positions that provide the Kinect with a useful field of view, or minimize the chance that the arm will block the overhead detection's view of the corner markers.

The navigation system has been updated with an improved interface, in order to move based on the robot's local coordinates. Several systems were implemented to convert destinations detected locally by the Kinect into a drivable location in the world frame, despite major differences between the two systems. The lighting was also updated to be more consistent, improving system reliability, and the overhead cameras were calibrated accordingly. The navigation has also been connected with the object detection code, in order to provide for automated movement to a detected object.

Object manipulation code was implemented to pick up and place objects. The final system can pick up objects both on the floor and on table surfaces that the arm can reach. Once it picks up an object, the youBot can place it in any reachable location on the floor, in the same orientation that it was picked up in. A simple prompt-based interface was also created to allow for quick access to this functionality.

All of these components were designed to function as a low-level control interface for the youBot as this project goes forward. While these functions are not at the point where they are easily accessible to a common user, they provide a solid framework on which to build a more complex system. This project took advantage of open source code and customized it to the youBot system, focusing on providing robust and consistent performance in these low level interfaces.

In terms of education, our model, using robot as teaching assistance has

Table 7.1: Tower of Hanoi lesson questionnaire: Level of attention and comprehension

Level of attention	
1	When was it invented the game of the Hanoi Tower? Who invented the game?
2	List the rules of the game that you remember?
3	Can you explain what recursion is?
4	How long it will take to the monks to solve the riddle?
Level of comprehension	
5	What is the link between the number of discs and the minimum number of moves?
6	For n = 6 which is the minimum number of moves?
7	If n increases, the way you solve the problem changes?
8	And time spent to solve the game changes?

received good evaluation from the teachers. (See Figure 7.4) Our case study also revealed remarkable consensus of opinion amongst the students: in fact the liking part of the questionnaire showed that 74% of the students enjoyed the experienced very much and the remaining 26% was satisfied about the experience. There were no negative perceptions about the youBot. The questions and the answers proposed in the questionnaire are reported in Figure 7.2 and Figure 7.3.

When students was asked about the interesting effects when learning with the youBot, group A that studied Hanoi Tower without the youBot but studied Elastic Collision with the youBot, they rated 16/19 answers of "yes, a lot", 2/19 answers of "yes, a bit", 1/19 answers of "indifferent". group B that studied Hanoi Tower with the youBot but studied elastic collision without the youBot, they rated 12/20 answers of "yes, a lot", 4/20 answers of "yes, a bit", 2/20 answers of "indifferent".

When students was asked about the intrigued effects in experiment with the youBot more than the lecture, the group A rated 10/19 answers of "yes, a lot", 4/19 answers of "yes, a bit", 3/19 answers of "indifferent", 2/19 answers of "Not to much", the group B rated 15/20 answers of "yes, a lot", 4/20 answers of "yes, a bit", 1/20 answers of "indifferent".

When students was asked about the youBot as a assistance to teach practical problem, the group A rated 12/19 answers of "yes, a lot", 7/19 answers of "yes, a bit", the group B rated 15/20 answers of "yes, a lot", 5/20 answers of "yes, a bit".

Observing the lesson we have noticed that every time the youBot executed actions to solve the game, students were very curious to see which was the next step and to understand how the youBot could "think" and act in an "intelligent" way. In addition students engaged discussion about the solution trying to

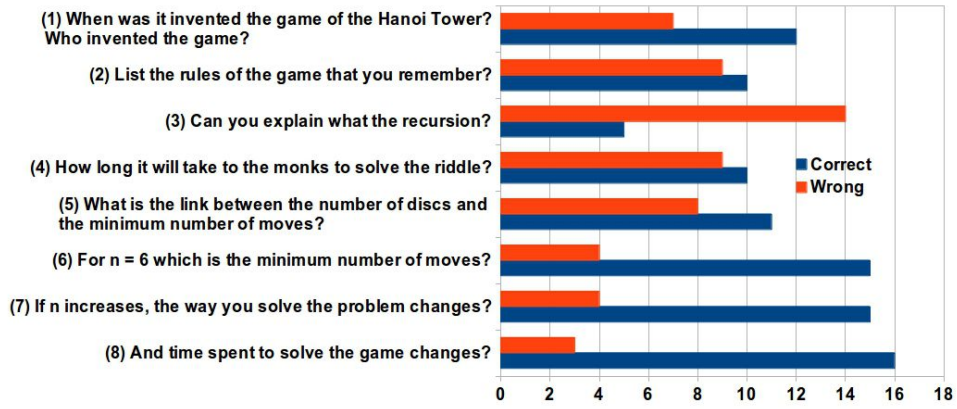


Figure 7.5: Level of attention and level of comprehension part for group A that studied Hanoi Tower without the youBot

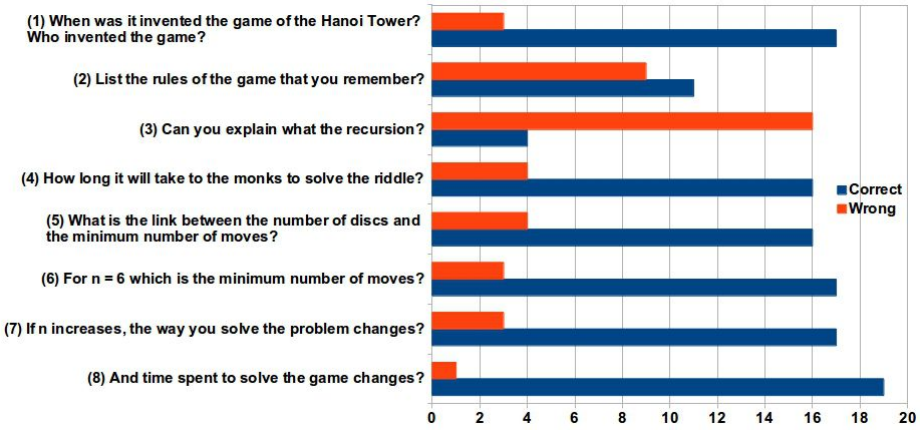


Figure 7.6: Level of attention and level of comprehension part for group B that studied Hanoi Tower with the youBot

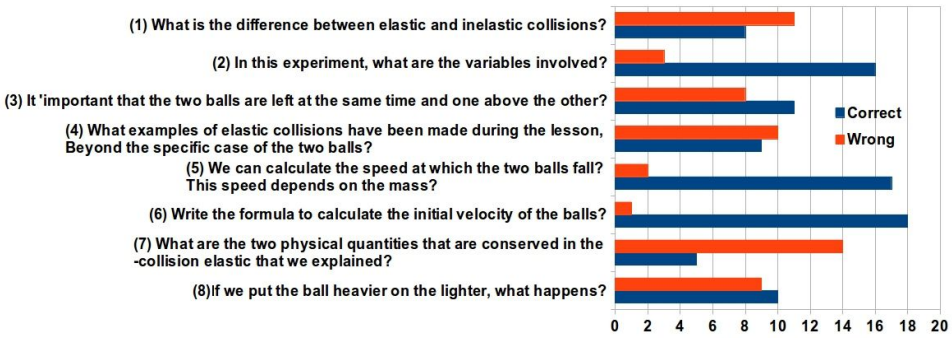


Figure 7.7: Level of attention and level of comprehension part for group A that studied Elastic Collision with the youBot

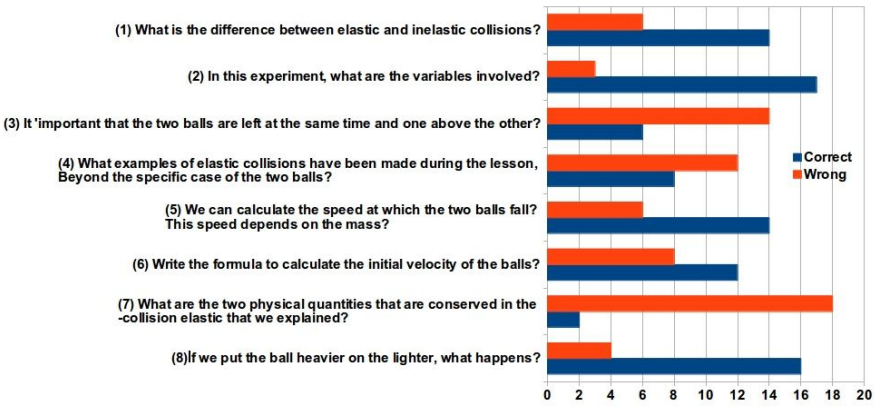


Figure 7.8: Level of attention and level of comprehension part for group B that studied Elastic Collision without the youBot

Table 7.2: Hanoi Lesson with and without the youBot: Questionnaires results.

Contents	**Percentage of correct answers with the youBot**	**Percentage of correct answers without the youBot**
The lesson in general	73,1%	58,7%
Level of attention	60%	48,6%
Level of comprehension	86,2%	75%

anticipate the next move of the robot.

The second and third part of the questionnaire, respectively, have been built to understand which was the level of attention during the lesson and if students learn the topic explained. In each section of the questionnaire we asked 4 questions. The questions are reported in table 7.1. We made a comparison between the results achieved in the two groups in order to understand if there was an added value in the lesson given with the youBot as the teaching assistance. The results of the questionnaire are shown in the table 7.2.

Comparing the results of Tower of Hanoi lesson in Figure 7.5 and in Figure 7.6, we can see their correct answers indicated by blue much more than red one indicated their wrong answers. In detail, when using the youBot as the teaching assistance, the percentage of correct answers was increasing about 16%.

Comparing the results of Elastic Collision lesson in Figure 7.7 and in Figure 7.8, we can see we can see their correct answers indicated by blue more than red one indicated their wrong answers. In detail, when using the youBot as the teaching assistance, the percentage of correct answers was increasing about

14%.

In terms of success, responses indicated that using the youBot as the teaching assistance kept the students interest and it was effective. From the table 7.1 and table 7.2, we can see how the use of the youBot as the teaching assistance has improved the percentage of correct answers of roughly 15%, the level of attention of roughly 12% and the comprehension of roughly 11.2%. This is an encouraging results since we want to refine and repeat the experiment with other groups of student.

Chapter 8

Conclusions

This chapter presents the limitations and discussions, then the future works and the conclusions are shown.

8.1 Limitations and discussions

For user interface, the system is currently operated almost exclusively from the command line, which is effective for development purposes, but needs to be made user-friendly for online use. For this reason, an interface between the online component and the code developed is an important future work. Before the portions that we created can be used as part of the final product, a simplified user interface is required, one that is easily controllable over the web interface.

For grasping, the standard the youBot gripper is extremely limited in its versatility, which greatly hampers number of applications for the system as a whole. With only 2.3 centimeters of travel, there is very little margin for error in grasping objects. Given the maximum size of 4.3 centimeters, the gripper cannot grab any large objects, and many of the small objects that it can pick up can be difficult to detect with the Kinect. There is also no force feedback within the system, making every grasp attempt a blind reach, and precluding the attempt to grab fragile objects. For this reason, it would be beneficial to redesign or otherwise improve the gripper mechanism.

In terms of robotic in education, for the lesson plan, the design of our lesson plans was created favorable conditions for teachers when teaching with the youBot. But the lesson plan was remains rigid, not flexible for teachers to have the time to talk and answer more questions from students. Lesson plans should also consider more elaborate for students 'communication' with the robot.

For the questionnaire, the design of our questionnaire is not really effective in statistical analysis. There are still many questions for students to write long answers and teachers must consider the dots instead of just marking answers on the severity by multiple-choice questions. To save time and easily to analyse the results, we will design the questionnaire to increase the number of multiple-choice questions, and therefore, the results will be easily evaluated for statistic.

8.2 Future works

In terms of robotic for user interface, we will try to be made user-friendly for online use. For navigation, we will try to make you not provide which direction of the robot will be facing once it gets to that location. For grasping, we will try to improve the successful rate and try to use other grippers with size of 4.3 centimeters or bigger to grasp other bigger objects. For object recognition, a possible improvement to the youBotś pickup and place functionality would be adding the ability to perform object recognition.

In terms of robotic in education as a primal results, the using of the robot as the teaching assistance has improved the percentage of correct answers of roughly 15%, level of attention of roughly 12% and the comprehension of roughly 11.2%., but we will refine and repeat more experimental evaluations to confirm that using the youBot as the teaching assistance would be more efficient.

8.3 Conclusions

In our work, we are questioning the relationship between robotics and educational outcomes.

We first begin by outlining the history of educational robotic in practice, then we find out the problems, motivations and propose our solutions to identify and quantify in educational benefits.

In terms of robotic as our first main contribution, we used the youBot platform which has an arm with 5 degrees of freedom and 4 wheel base. Using invert kinematic-Cartesian mode and arm motion planning, we specifies the desired target position of the gripper in Cartesian space as (x,y,z) to solve many problems such as "Writing", "Cleaning", "Elastic Collision", "Tower of Hanoi" problem. To solve "Tower of Hanoi" for example, we focus on inverse kinematics solution to find parameters for the youBot approach to objects and calculate all

joints of position variables.

In terms of education as our second main contribution, we designed a lesson plan in particular to teach "Tower of Hanoi" problem conducted with the youBot as a teaching assistance. Looking at the results of the questionnaire and at the experience done with the students we can state that the youBot fits in with the existing robotics teaching challenging: it is highly motivating for participants and advances both academic and personal development skills.

Finally, we present some results from questionnaire study conducted with the robot and the comparison of the two groups is made for validation. As a primal results, the using of the robot as the teaching assistance has improved the percentage of correct answers of roughly 15%, level of attention of roughly 12% and the comprehension of roughly 11.2%.

For the future works, we will refine and repeat more experiments with other groups of student and let robot also solve more problems in difference subjects.

Bibliography

[1] P. Seymour, *Mindstorms: Children, computers, and powerful ideas.* Basic Books, 1980.

[2] P. Seymour, *Constructionism: A new opportunity for elementary science education. Massachusetts Institute of Technology.* Media Laboratory, Epistemology and Learning Group, 1986.

[3] P. Seymour, *The children's machine: Rethinking school in the age of the computer.* Basic Books, 1993.

[4] A. Eguchi, "Educational robotics theories and practice: Tips for how to do it right," *Robots in K-12 Education: A New Technology for Learning, B. Barker, G. Nugent, N. Grandgenett, & V. Adamchuk, Eds. Hershey, PA, IGI Global*, pp. 1–30, 2012.

[5] N. Grandgenett, E. Ostler, N. Topp, and R. Goeman, "Robotics and problem-based learning in stem formal educational environments," *Robots in K-12 Education: A New Technology for Learning*, p. 94, 2012.

[6] B. S. Barker and I. Global, *Robots in K-12 Education: A New Technology for Learning.* Citeseer, 2012.

[7] R. Singh, F. Maire, J. Sitte, A. Tickle, T. Naniwa, E. U. S. H. Robot, F. Yamasaki, and Y. Nakagawa, "Robots for education," in *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, p. 230, Springer, 2005.

[8] D. P. Miller, I. R. Nourbakhsh, and R. Siegwart, "Robots for education," *Springer handbook of robotics*, pp. 1283–1301, 2008.

[9] M. Resnick, F. Martin, R. Sargent, and B. Silverman, "Programmable bricks: Toys to think with," *IBM Systems journal*, vol. 35, no. 3.4, pp. 443–452, 1996.

[10] I. R. Nourbakhsh *et al.*, "Robots and education in the classroom and in the museum: On the study of robots, and robots for study," 2000.

[11] A. Druin and J. A. Hendler, *Robots for kids: exploring new technologies for learning*. Morgan Kaufmann, 2000.

[12] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1, pp. 59–65, IPCB: Instituto Politécnico de Castelo Branco, 2009.

[13] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1400–1405, IEEE, 2007.

[14] K. P. Morgan, "Educational robotics as leadership development for youth," 2013.

[15] P. Dillenbourg, "What do you mean by collaborative learning?," *Collaborative-learning: Cognitive and Computational Approaches.*, pp. 1–19, 1999.

[16] J. B. Weinberg, W. W. White, C. Karacal, G. Engel, and A.-P. Hu, "Multidisciplinary teamwork in a robotics course," *ACM SIGCSE Bulletin*, vol. 37, no. 1, pp. 446–450, 2005.

[17] M. Umaschi Bers, "The role of new technologies to foster positive youth development," *Applied Developmental Science*, vol. 10, no. 4, pp. 200–219, 2006.

[18] C. Dede, "Transforming education for the 21st century: New pedagogies that help all students attain sophisticated learning outcomes," *Commissioned by the NCSU Friday Institute, February*, 2007.

[19] H. A. Samani, J. T. K. V. Koh, E. Saadatian, and D. Polydorou, "Towards robotics leadership: An analysis of leadership characteristics and the roles robots will inherit in future human society," in *Intelligent Information and Database Systems*, pp. 158–165, Springer, 2012.

[20] B. Mutlu, J. Forlizzi, and J. Hodgins, "A storytelling robot: Modeling and evaluation of human-like gaze behavior," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pp. 518–523, IEEE, 2006.

[21] E. Cejka, C. Rogers, and M. Portsmore, "Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school," *International Journal of Engineering Education*, vol. 22, no. 4, p. 711, 2006.

[22] A. Eguchi, M. Kandlhofer, Emanuele Menegatti, G. Steinbauer, S. Hirschmugl-Gaisch, Eck, *et al.*, "Educational robotics for promoting 21st century skills," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 8, 2014.

[23] B. Fagin and L. Merkle, "Measuring the effectiveness of robots in teaching computer science," in *ACM SIGCSE Bulletin*, vol. 35, pp. 307–311, ACM, 2003.

[24] P. Gonzales, J. C. Guzmán, L. Partelow, E. Pahlke, L. Jocelyn, D. Kastberg, and T. Williams, "Highlights from the trends in international mathematics and science study (timss)," *National Center for Education Statistics, US Department of Education*, pp. 1–104, 2004.

[25] M. Lemke, A. Sen, E. Pahlke, L. Partelow, D. Miller, T. Williams, D. Kastberg, and L. Jocelyn, "Outcomes of learning in mathematics literacy and problem solving: Pisa 2003 results from the," 2004.

[26] S. Waddell and K. L. Doty, "Why tearch robotics?," *Tech Directions*, vol. 58, no. 7, 1999.

[27] I. R. Nourbakhsh, K. Crowley, A. Bhave, E. Hamner, T. Hsiu, A. Perez-Bergquist, S. Richards, and K. Wilkinson, "The robotic autonomy mobile robotics course: Robot design, curriculum design and educational assessment," *Autonomous Robots*, vol. 18, no. 1, pp. 103–127, 2005.

[28] R. D. Beer, H. J. Chiel, and R. F. Drushel, "Using autonomous robotics to teach science and engineering," *Communications of the ACM*, vol. 42, no. 6, pp. 85–92, 1999.

[29] E. Mauch, "Using technological innovation to improve the problem-solving skills of middle school students: Educators' experiences with the lego mindstorms robotic invention system," *The Clearing House*, vol. 74, no. 4, pp. 211–213, 2001.

[30] M. Portsmore and C. Rogers, "Bringing engineering to elementary school," *Journal of STEM education*, vol. 5, 2004.

[31] D. J. Barnes, "Teaching introductory java through lego mindstorms models," in *ACM SIGCSE Bulletin*, vol. 34, pp. 147–151, ACM, 2002.

[32] V. S. Moore, "Robotics: Design through geometry.," *Technology Teacher*, vol. 59, no. 3, pp. 17–22, 1999.

[33] J. Geissler, P. J. Knott, M. R. Vazquez, and J. R. Wright Jr, "Virtual reality robotic programming software in the technology classroom.," *Technology Teacher*, vol. 63, no. 6, p. 6, 2004.

[34] M. Robinson, "Robotics-driven activities: Can they improve middle school science learning?," *Bulletin of Science, Technology & Society*, vol. 25, no. 1, pp. 73–84, 2005.

[35] S. P. Wagner, "Robotics and children: Science achievement and problem solving.," *Information Technology in Childhood Education Annual*, vol. 101, p. 45, 1999.

[36] E. Müggler, M. Fässler, D. Scaramuzza, S. Huck, and J. Lygeros, "Torque control of a kuka youbot arm," 2013.

[37] S.-F. Chen and I. Kao, "Conservative congruence transformation for joint and cartesian stiffness matrices of robotic hands and fingers," *The International Journal of Robotics Research*, vol. 19, no. 9, pp. 835–847, 2000.

[38] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[39] S. Sharma, G. K. Kraetzschmar, C. Scheurer, and R. Bischoff, "Unified closed form inverse kinematics for the kuka youbot," in *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pp. 1–6, VDE, 2012.

[40] T. Jenkel, *Mobile Manipulation for the KUKA youBot Platform*. PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, 2013.

[41] P. Malmsten, *Object discovery with a microsoft kinect*. PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, 2012.

[42] R. Bischoff, U. Huggenberger, and E. Prassler, "Kuka youbot-a mobile manipulator for research and education," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.

[43] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

[44] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2149–2154, IEEE, 2004.

[45] J. Smisek, M. Jancosek, and T. Pajdla, "3d with kinect," in *Consumer Depth Cameras for Computer Vision*, pp. 3–25, Springer, 2013.

[46] Y. Okubo, C. Ye, and J. Borenstein, "Characterization of the hokuyo urg-04lx laser rangefinder for mobile robot obstacle negotiation," in *SPIE Defense, Security, and Sensing*, pp. 733212–733212, International Society for Optics and Photonics, 2009.

[47] G. Kraetzschmar, W. Nowak, N. Hochgeschwender, R. Bischoff, D. Kaczor, and F. Hegger, "Robocup@ work rulebook," 2013.

[48] O. Michel, "Webots: Symbiosis between virtual and real mobile robots," in *Virtual Worlds*, pp. 254–263, Springer, 1998.

[49] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Point cloud library," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/12, 2012.

[50] J. S. Hott, S. M. Papadopoulos, N. Theodore, C. A. Dickman, and V. K. Sonntag, "Intraoperative iso-c c-arm navigation in cervical spinal surgery: review of the first 52 cases," *Spine*, vol. 29, no. 24, pp. 2856–2860, 2004.

[51] M. R. AbuQassem, *Simulation and Interfacing of 5 DOF Educational Robot Arm*. PhD thesis, Islamic University of Gaza, 2010.

[52] J. Wang, F. Adib, R. Knepper, D. Katabi, and D. Rus, "Rf-compass: robot object manipulation using rfids," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, pp. 3–14, ACM, 2013.