ULB UNIVERSITÉ LIBRE DE BRUXELLES

Bruface Brussels Faculty of Engineering

Vrije Universiteit Brussel

# Development of a human-robot interaction manufacturing task using the Baxter coworker.

Yuri Durodié

Master thesis submitted under the supervision of
Prof. Dr. Ir. Bram Vanderborght
Prof. Dr. Ir. Dirk Lefeber

The co-supervision of
Ir. Ilias El Makrini

In order to be awarded the Master's Degree in
Applied Sciences and Engineering, Electro-Mechanical Engineering, Mechanics-Construction Engineering

Academic year
2014-2015

## 0.1 Acknowledgement

I would like to take the opportunity to thank several people. Without their valued help this master thesis would not have been finished within the deadlines.

First and foremost, I would like to thank Bram, Ilias, Carlos and Dirk for having given me the unique opportunity to work with Baxter and for their academic inputs to my work. It has been a great experience.

Many thanks also to my studentmates, Tom, Jan, Robin and Gert, with whom the morning, the midday and the afternoon breaks were spent drinking coffee and joking around, keeping the morale high.

Special thanks further to Tom, whom served as my first source of reasoning and was also used as a guinea pig. (Outstanding soundboard.)

I would like to also thank my close friends, Nicolas and Thibault, whom kept telling me that I would succeed in finishing my thesis in time. (Close call guys!)

I further would like to give a special appreciation to my father for his multiple and constructive suggestions to my work and for his challenging questions.

My mother finally deserves a special acknowledgement for the many hours she spent reading my thesis and trying to understand everything I wrote, so she could help me in writing it better and clearer in English. It must be hard being a mom.

Thank you all for your support !

Yuri Durodié
May 2015.

## 0.2 Abstract


### Abstract English

The use of collaborative robots in the industry is raising. The ability to work closely together with a human gives the advantage of being able to use both the problem solving capabilities of the human as well as the repeatability and accuracy of the robot.
This master thesis aims to built and test a toolbox using the existing robot Baxter, a dual arm collaborative robot of Rethink Robotics.

The algorithms used to interact with the objects are elaborated and some limitations of the iterative inverse kinematics of Baxter where shown. Because of these limitations a pragmatic analytical inverse kinematic solver was developed and tested.
The performance of the algorithms were measured and show encouraging results.

A wizard based configuration tool was developed on Baxter to interact with humans. The wizard aims to intuitively guide a human collaborator through the configuration of sub-tasks, such that the robot could perform a task in collaboration with a human operator.
The usability of this wizard was also tested and the results are quite promising.

### Abstract Dutch

Het gebruik van collaboratieve robots in de indutsrie is aan het groeien. De mogelijkheid om voor de mens en de robot om nauw samen te werken, geeft het voordeel dat zowel de probleem oplossende capaciteiten van de mens als de reproduceerbaarheid en nauwkeurigheid van de robot kunnen gebruikt worden.
Deze master thesis heeft als doel om een toolbox te bouwen en te testen voor de bestaande robot Baxter. Baxter is een twee-armige robot van Rethink Robotics.

De algoritmes gebruikt om met objecten te interageren worden besproken en sommige beperkingen van de iterative inverse kinematic solver van Baxter worden aangetoond. Omwille van deze beperkingen werd een pragmatische analytische inverse kinematic solver ontwikkeld en getest.
De prestaties van deze algoritmes werden getest en geven aanmoedigende resultaten.

Er werd ook een wizard gebaseerde configuratie module gemaakt op Baxter om met de mens te interageren. Doel van deze wizard is om de operator intuitief te begeleiden bij het configureren van de deeltaken, de welke samen een totale taak vormen die de robot kan uitvoeren in samenwerking met de operator.
De bruikbaarheid van deze wizard werd ook getest en de resultaten zijn veel belovend.

### Abstract French

L'utilisation des robots collaborants avec l'agent humain dans l'industrie est en hausse. La capacité de travailler en étroite collaboration donne l'avantage de pouvoir utiliser à la fois la capacité humaine de résoudre des problèmes ainsi que la capacité d'accomplir des tàches répétitives avec la précision du robot.

Ce traité vise à construire et tester une boîte à outils à l'aide du robot Baxter existant, un robot de collaboration à deux "bras" de Rethink Robotics.

Les algorithmes utilisés pour interagir avec les objets sont élaborés et certaines limitations du logiciel de solution de cinématique inverse iterative du robot Baxter inclus d'origine, sont montrées dans le câdre de ce travail. En raison de ces limitations, une version analytique simplifiée du logiciel de solution de cinématique inverse a été développée et testée.
La performance des algorithmes a été testé et a montré des résultats encourageants.

Un logiciel de configuration assistant Wizard pour l'intéraction avec l'humain a été développé pour le robot Baxter. Cet assistant vise à guider un collaborateur humain de façon intuitive pour la configuration des sous-tâches, de sorte que le robot pourrait accomplir une tâche en collaboration avec un opérateur humain.
La facilité d'utilisation de cet assistant a également été testé et ces résultats sont étonnants.

# Contents


0.1 Acknowledgement . . . 1
0.2 Abstract . . . 2

**1 Motivation** **9**
1.1 Goal of thesis . . . 9
1.2 Lay-out of thesis . . . 10

**2 State of the art** **11**
2.1 Collaboration between humans and robots in the automotive . . . 11
2.1.1 Levels of collaboration . . . 11
2.1.2 High level cooperation . . . 12
2.2 Safety . . . 14
2.2.1 Post-Collision . . . 14
2.2.2 Pre-Collision . . . 15
2.3 Communication . . . 16
2.4 Usability . . . 17
2.5 The Baxter robot . . . 19
2.5.1 Hardware . . . 19
2.5.2 Software . . . 20
2.5.3 State of art Baxter Research version . . . 21
2.6 State of art of Baxter alike robots . . . 22
2.7 Conclusion . . . 22

**3 Tools** **23**
3.1 Discrimination and Artificial Intelligence . . . 23
3.1.1 Definitions . . . 23
3.1.2 The discrimination problem and classification . . . 25
3.1.3 Classification algorithm . . . 25
3.1.4 Reinforcement . . . 26
3.1.5 The naming game . . . 26
3.2 Computer Vision . . . 27
3.2.1 Finding shapes . . . 27
3.2.2 Shape recognition . . . 28
3.2.3 Hu Moments . . . 31
3.2.4 Matching R($\theta$) representation . . . 31
3.3 Inverse kinematic solvers . . . 32
3.3.1 Inverse kinematic solver on Baxter . . . 32
3.3.2 Analytical inverse kinematic solver . . . 33
3.3.3 Testing the inverse kinematic solver . . . 38
3.3.4 Conclusion of the inverse kinematic solver . . . 45
3.4 Conclusion . . . 45

**4 The program** **46**
4.1 Concept . . . 46
4.2 The wizard . . . 47
4.3 Learn objects procedure . . . 48

4.3.1 Implementation . . . 49
4.4 Learn task procedure . . . 49
4.4.1 Implementation . . . 50
4.5 Execute task procedure . . . 52
4.6 conclusion . . . 53

**5 The experiments** **54**
5.1 Validating the gripping location and the orientation algorithms . . . 54
5.1.1 Experimental setup . . . 55
5.1.2 Analysis of the data . . . 59
5.1.3 Conclusion on the algorithms experiment . . . 63
5.2 Validating the usability . . . 63
5.2.1 Experimental setup . . . 64
5.2.2 Analysis of the data . . . 65
5.2.3 Conclusion on the usability experiment . . . 67
5.3 Conclusion of the experiments . . . 67

**6 Conclusion** **68**

**7 Further work** **69**

**8 Bibliography** **70**

**9 Annex** **73**
9.1 Proof of equations 3 and 4 . . . 73
9.2 Programs . . . 74
9.2.1 Baxter_Controll.py . . . 74
9.2.2 camera_control_movement.py . . . 75
9.2.3 Learn_new_object3.py . . . 75
9.2.4 menu.py . . . 75
9.2.5 objects.py . . . 76
9.2.6 task.py . . . 76
9.2.7 Thesis.py . . . 77
9.2.8 JointCommander.py . . . 77
9.3 Mail with Rethink Robotics . . . 77
9.4 Wizard screens . . . 80
9.5 Experiment on algorithms . . . 80
9.6 Experiment on usability . . . 80

# List of Figures


1 Example of a low level HRC [1]. . . . . 11
2 Simplified scheme of behaviour [17]. . . . . 14
3 Non restrictive summary flow chart for safety strategies in HRC. . . . . 14
4 Example of gesture based communication from the robot towards the human collaborator [21]. . . . . 17
5 Example of gaze based communication from the robot towards the human collaborator [6]. . . . . 17
6 Example of Labview for robot programming [23]. . . . . 18
7 Example of CAD-based programming RobotStudio of ABB [25]. . . . . 19
8 The hardware of Baxter [21]. . . . . 20
9 Hybrid work cell with Baxter where the human collaborator points to the object with a laser [5]. . . . . 21
10 YuMI, the dual arm collaborative robot of ABB [31]. . . . . 22
11 Examples of the definitions of AI in table 2. . . . . 24
12 Objects used in experimental set-up. . . . . 28
13 Example of R($\theta$) plot [37]. . . . . 29
14 Example of S($\phi$) plot [37]. . . . . 30
15 R($\theta$) representations. . . . . 32
16 References and frames of Baxter (Image adapted from [44]). . . . . 33
17 Transformation of EE to EE'. On the left the projection in the xy-frame of O, on the right the projection on the Rz-frame of O. . . . . 34
18 Transformation of O to O'. On the left the projection in the xy-frame of O, on the right the projection on the Rz-frame of O. . . . . 35
19 Transformation of O' to O". On the left the projection in the xy-frame of O', on the right the projection on the Rz-frame of O'. . . . . 35
20 Projection on z"x"-frame of O" of the offsets. . . . . 36
21 Error map for the analytical IK. The error is shown in the top graph and the trajectory in the bottom graph. . . . . 39
22 Error of the iterative IK solver. . . . . 40
23 Zoom on errors of errormaps for analytical and iterative IK. . . . . 41
24 Errormaps of the analytical IK solver on straight lines. . . . . 42
25 Errormaps of the analytical IK solver on straight lines. . . . . 42
26 Maximum errors on position of Baxter . . . . . 43
27 Errors on the positions of the joints. . . . . 44
28 Step responses of joint $S_1$. . . . . 44
29 General flow chart. . . . . 46
30 Shape game [48]. . . . . 47
31 First screen of the wizard. . . . . 47
32 Tree architecture of the wizard. . . . . 47
33 Flowchart of the learn objects procedure. . . . . 48
34 Flow chart of the learn task procedure. . . . . 50
35 Flow chart of a possible task. . . . . 51
36 Special screen to help supervisor: Case of placing an object when no object has been gripped. . . . . 51
37 Flow chart of the task execution procedure. . . . . 52

38 Time line of the measured times. . . . . 55
39 Set-up for experiment 1. . . . . 56
40 Gripping success . . . . . 56
41 Types of frequent failures. . . . . 57
42 Flow chart of program for the experiment of the validation of the gripping location and orientation algorithms. . . . . 58
43 Adaptation of object 2 to have one distinct maximum. . . . . 61
44 Ideal flowchart for the configuration of Baxter for the usability experiment. 64
45 Experimental set-up for the validation of the usability. . . . . 65

# List of Tables


1 Joint naming convention in Baxter based on [26]. . . . . 20
2 Table of AI definitions. . . . . 24
3 Mean, std dev and maximum error of the errors shown in figure 23. . . . . 41
4 Conclusive table on the inverse kinematic solvers. . . . . 45
5 Overview of measured parameters in the validation of the location, orientation and gripping algorithms. . . . . 55
6 Measured data for object 1. . . . . 59
7 Measured data for object 2. . . . . 62
8 Overview of measured parameters in the validation of usability. . . . . 64
9 Measured data during the usability experiment. . . . . 65

# 1 Motivation

Most of the industrial robots used today in manufacturing are very large machines which are, for safety reasons, fenced off from humans and are very difficult to reprogram. Yet the last decade a lot of research has been done in the use of small and easy reprogrammable robots which cooperate with a human partner. There are multiple reasons for this evolution.

The main reasons are :

1. Today's industrial robots are heavy independent machines, which take highly trained specialists to program for a certain task. The reprogramming of the robots usually takes a lot of time, from several days to some months [1][2][3]. While nowadays the product changeovers are measured in months [2][3] or even days or less depending on the application.
2. Besides this, those industrial robots need a lot of security systems to prevent them from harming humans. These safety systems are bulky and inflexible [1].
3. Humans and robots share complementary skills. Humans are intelligent and dexterous, while robots have more precision, strength and are able to do repetitive tasks without losing attention [3][4]. Therefore the concept of hybrid cells, where a robot and a human work together, is being developed [5].

This evolution will mostly affect the small and medium enterprises (SMEs) [4]. Many of these SMEs cannot afford the large industrial robots. However a lot of tasks in SMEs, which don't require the speed en precision of the large industrial robots, could be automated [6].

The cost of these small, easy-reprogrammable robots would be competitive with the labour cost in low-cost countries [3][4]. This would have several impacts, which would benefit the economies of North America and Europe [3][4]:

- Retain intellectual skills that would otherwise go offshore.
- Make Western companies more competitive.
- Provide jobs for developing, producing, maintaining and training robots.
- Improve working conditions.
- Reduce manufacturing lead-time for finished goods.
- Allow operation by lower skilled workers.
- ...

## 1.1 Goal of thesis

The goal of this thesis is to program the Baxter robot in such a way that it can easily be taught to perform a manufacturing task by personnel that is not trained in programming. The program development on Baxter should be done in such a way that the interaction between the operator is (safe), intuitive and easy. It should also make use of the complementary skills as explained in the motivation.

## 1.2 Lay-out of thesis

The master thesis is made out of 4 major parts.

The first part, section 2, will elaborate the state of the art for collaborative robots. In this part all major aspects of human robot collaboration are described. Those aspects are safety, communication and usability. At the end of the first part the Baxter robot is described and also a small state of the art is given for the research on Baxter.

The second part, section 3, explains in more detail the algorithms that are needed to interact with objects. The first part of this section will explain how to discriminate objects one from another. This will be followed by an explanation of Computer Vision and a discussion on some representations of shapes. This part ends with the study of the inverse kinematic solver and the inherent limitations of Baxter.

The third part, section 4, will explain in more detail the control architecture of the wizard. The architecture is divided in 3 procedures, the first is the learn object procedure, the second one is the learn task procedure and the last one is the execute task procedure.

In the last and major part, section 5, the experiments are presented. Two experiments were done. The first one consisted in testing the performance of the used algorithms, described in section 3. The second experiment tested the usability of the wizard developed in section 4.

Afterwards a conclusion was written in section 6 and several proposals for future work were made in section 7.

# 2 State of the art

The idea of the hybrid work cell, where a robot and a human work together to achieve a common goal, has grown over the past decade. The idea is not only pushed by industry, but also by NASA and other non profit organisations [7].

The first part of this state of the art will elaborate the different levels in human robot collaboration and touch the safety and communication in human robot collaboration. Given the prime importance of safety in any manufacturing environment, it's good practise to keep this aspect in mind when working on human robot collaboration.

In the second part the Baxter robot will be explained in more detail and the state of the art of easy and intuitive programming of robots to perform a manufacturing task will be summarised.

## 2.1 Collaboration between humans and robots in the automotive

### 2.1.1 Levels of collaboration

There are different levels of Human Robot Collaboration (HRC) currently implemented in automotive manufacturing, which is a trend setting industry for robotics.

*Low level*
In the low level collaboration model the human and the robot do not interact directly with each other [1][5].When loading parts there is a transfer device that makes sure no direct contact is needed [1][5]. An example of this is a typical welding cell, where the operator places the parts on a rotary table, while the welding robot is welding these parts into a sub-assembly and a material handling robot removes the sub-assembly. Once the parts are placed on the rotary table the operator has to clear the area and push a button to notify that the area is clear, so the process can start [1]. A schematic of this example is shown in figure 1.

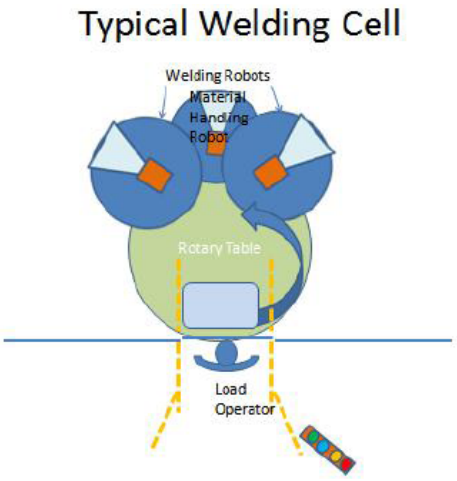



**Figure 1:** Example of a low level HRC [1].

This is how most of the large industrial robots work today.

*Medium level*
In the medium level collaboration the operator comes in direct contact with a robot. The important part is the operational state of the robot when the human operator is in the work space of the robot. The operator and the robot work sequentially, this means that the robot cannot begin the next task before the operator has initiated it with an external input away from the robot's workspace [1][5]. It's important that the actuators of the robot at that moment are de-energized and the robot is positioned at the edge of its working range [1]. For example the low level welding collaboration of the previous paragraph would change in a medium level collaboration if the operator places the parts directly in front of the robot.
In the future the intend is to have the robot actuators continuously energized while the operator does his work. Also the input of the operator, needed to start the next task of the robot, has to be eliminated such that the robot is able to automatically recognise when the operator has finished his job. A further development is to split the workspace of the robot in a restricted zone, where the robot has to stop when a human is in that zone, and the rest of the workspace where the robot is still working on his task while paying attention to the human operator [1].

*High level*
In current high level human robot collaboration the robot's actuators are energized during the whole process and the robot still works while a human is in any part of his workspace. The speed and/or motion of the robot may be modified [1][5]. Currently such high level human robot collaborations are scarce in automotive manufacturing [1].
In the future the speed and motion of the robot have to be synchronized with the human activity to form a partnership working towards the same goal [1]. This means that the robot has to become aware of situations and be able to anticipate and adapt to dynamic situations using natural human communication mechanisms [1].

The thesis will focus on the high level collaboration.

### 2.1.2 High level cooperation

When humans work in team they have to develop a fluent team behaviour, the same holds true for robots [8][7][9]. Safety [10][11][12] and communication [7][9] are key aspects for a successful collaboration.

Besides safety and communication, the robots also have to be easily taught how to perform a task [13][14][15]. This aspect will be called usability as was done in [16].

All 3 aspects :
1. Safety
2. Communication
3. Usability

dependent on the hardware and software of the robot.

The hardware of a robot can be seen as a collection of joints, links, sensors and indicators:

- Joint: Joints provide the degrees of freedom of a robot. A joint can be actuated or not.
- Link: Connects the joints with each other.
- Sensor: Device that measures physical properties. For example: camera's, distance sensors, buttons, etc.
- Indicator: Device that provides information for the user. For example: lights, screens, etc.

The software of the robot is the collection of programs running on the robot:

- Operating systems, IDKs, etc. are important for the development of the programs on the robot.
- Behaviour: Set of programs that will run in operation, which are important for the end user (not highly trained personnel).

The end user will mostly be confronted with the behaviour and hardware of the robot. In behavioural psychology and artificial intelligence, it is widely accepted that behaviour is organized in three layers [17]:

- The reactive layer constitutes of the lowest-level-processes which do not evolve with learning [17]. State information, coming from sensory inputs, is immediately acted upon with appropriate motor actions [17]. In this layer the most basic safety systems will be incorporated. The reactive layer could also enhance other systems, such as preventing the robot's collision with obstacles or keeping the robot's equilibrium while walking.
- The deliberative layer corresponds to complex and skilled behaviours and guides most motor actions. At this level, a work plan is conceived based on the sensory inputs, the inputs of the reactive layer and of the reflective layer, to fulfil a goal [17]. Most of the tools for the end user will be placed in this layer.
- Reflective layer is the most complex form of the conscience in which the mind deliberates itself. Unlike the lower layers, the reflective layer is not linked with sensory and motor systems. This layer evaluates the plans (formed by the deliberative layer) and may decide to change these plans in order to better attain the goal. Therefore, its function is 'to think', not 'to act' [17].

In figure 2 a simplified scheme of the behaviour as explained previously can be found.

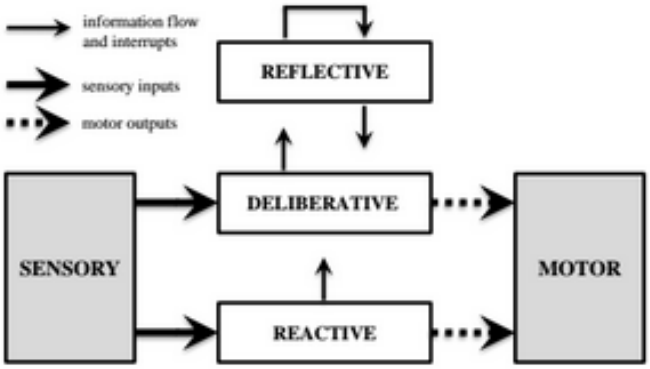



**Figure 2:** Simplified scheme of behaviour [17].

## 2.2 Safety

The most important challenge encountered when introducing robots into an anthropic environment is safety [10][11][12].The previous industrial robots were completely fenced off and had a lot of safety systems to ensure nobody could get hurt [13][10][11][12]. In human robot collaboration those robust safety system are not possible any more, but the safety of the human collaborators has to be ensured anyway.

There are 2 distinct strategies to obtain safe human robot collaboration: Pre-collision and Post-collision [10][11][16]. These strategies can have multiple approaches [11]. Most robotic systems aim to use at least one approach for each strategy [11].

In figure 3 a summarizing flowchart is given of the safety strategies in human robot collaboration. The flowchart is not restrictive.

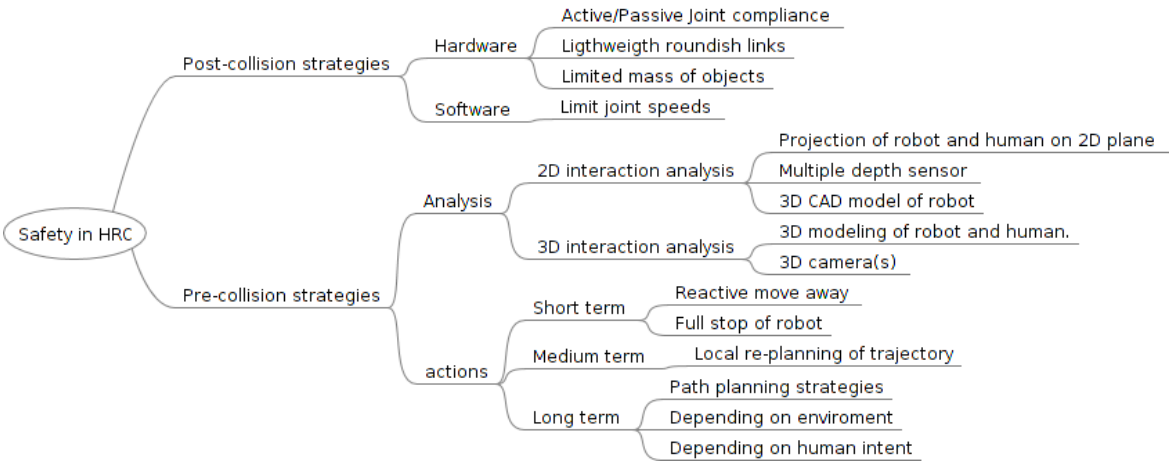



**Figure 3:** Non restrictive summary flow chart for safety strategies in HRC.

### 2.2.1 Post-Collision

The Post-collision strategy aims to reduce the impact and injuries after the occurrence of an unexpected collision between a robot and a human [10][12]. There are no universal

criteria yet to define the accepted severity of such a collision. The most commonly used standard is the ISO 10218-1-"Safety requirements for robots in manufacturing industry" [12].

Most post collision strategies are built in the hardware of the robot:

- Compliant actuators or joints: Compliant joints submit themselves to an external force [10][13]. The robot will move away, or in the direction of the force so to avoid an injury [13].
- Design of the links.
- Limiting speed.

Joint compliance is one of the main targets to increase post-collision safety [16][18]. There is active and passive compliance [16][18]. In passive compliance the actuator has an elastic element while in active compliance the actuator tries to mimic the elastic behaviour [18]. Though using active compliance control has shown a delay between the impact and the compliance [16]. This is because the main impact phase is over before the controller can react based on the sensor input [16].In passive compliant joints the motor and link inertia are physically decoupled resulting in a delay-free compliant behaviour in case of collision [16]. An example of such an actuator is the Series Elastic Actuator (SEA) which is used in the Baxter robot and the BioRob Robot Arm [4][16]. A downside of this type of actuator is the reduced accuracy on the end effector [12][16].Therefore research has been done towards variable damping actuators [19]. This type of actuators are semi-active because the level of damping can be regulated on demand [19].

An important parameter in a collision is the speed and weight of the objects colliding. By making the links as light as possible the impact of a collision at a certain speed will be small [4][16][13]. Another aspect is the form of the link. Links that are roundish will cause less damage in collision because the area of impact will be bigger than with joints having edges [13].

By limiting the speed at which the robot can operate, the impact can be reduced. When the robot allows it, the speed can be adapted to the presence of a human [4]. This is done in Baxter where the sonar senses the presence of a human [4]. Baxter will then move slower [4].

Impact reduction can also be achieved by reducing the payload of the robot [4][16]. The most important cause of death and critical injuries in HRC is clamping [20].

### 2.2.2 Pre-Collision

In pre-collision strategies one tries to avoid an impact. Most methods exist in calculating the physical separation between the robot and the human, tracking this separation and taking preventive actions when this distance gets below a threshold [10][11]. Depending on the time before collision occurs, 3 different action strategies can be used [11]:

1. Long term: Path planning strategies can be implemented in order to avoid any collision [11]. This strategy would be situated in the deliberative layer and the reflective layer.

2. Medium term: Local re-planning and trajectory modification can be used to modify the robot's movement and avoid a hazard [11]. This strategy would be situated in the the deliberative layer.

3. Short term: In the case of a hazard, reactive strategies could be used to move the robot away [11]. This strategy would be situated in the reactive layer.

Without going much more in detail, the pre-collision strategies can be divided in 2 families based on their approach to analyse the situation [10]. The first type treats the problem in a 2D euclidean space by projecting the robot and the human on the same plane [10]. The second type analyses the problem directly in a 3D euclidean space by using the explicit 3D model of the human and the robot [10].

## 2.3 Communication

Communication can be seen as the total set of interactions between the human and the robot. Communication is a key factor for success in human robot collaboration [21][7][21][22].

When humans collaborate with each other they use speech, gesture, gaze and non-verbal cues to communicate [7][14]. Also the environment is used to communicate with on another [7]. Objects can serve as frame of reference [7] and can even introduce possible manipulations [14]. In human human collaboration gaze has a very important role [7]. The gaze provides visual feedback [7], for example by showing if the communication is clear or not. It regulates the flow of conversation [7]. In human human collaboration, communication can be modelled by 3 types [7]:

1. Audio: Communication that can be heard, like speaking or screaming.

2. Visual: Communication that can be seen, like posture, gaze or gestures.

3. Environmental: Interactions with the surrounding world, using objects as reference.

Those types of communication can be affected by the working environment and by technology [7]. In an industrial environment there is a lot of noise, making reliable verbal communication difficult [22]. Therefore human robot collaboration would depend heavily on clear non-verbal communication [22]. This mostly includes gestures and gaze. But also environmental cues are important. When a robot needs to learn a position, taking the robot by the arm and move it to the position would be much easier than try to gesture it. Humans use this also when they communicate with each other: for example when learning to tennis or golf it is not unusual that the trainer shows the correct movement by taking the pupil by the arms. This is also done with Baxter [15] and the Biorob [16].

Communication can be split in 3 parts:

1. Communication from the human to the robot, which implies the correct interpretation of human cues [7][22]. This has been studied in [22].

2. Communication from the robot to the human, which implies the correct implementation of understandable communication cues towards the human cooperator [21][7]. In [21] a robot arm has been programmed to communicate with a human collaborator by gestures. This is visualized in figure 4. In the commercial version of Baxter the robot is equipped with a face, which it uses to look at where it is going to move its arms [15]. This face and gaze can be seen in figure 5.

3. Interaction with the environment and in particular with objects [14][7].

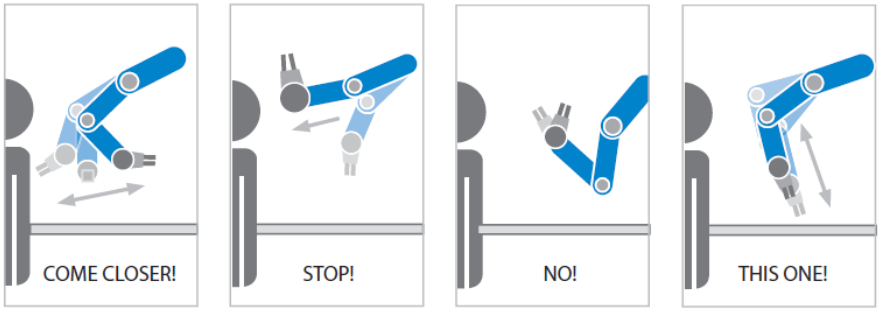



**Figure 4:** Example of gesture based communication from the robot towards the human collaborator [21].

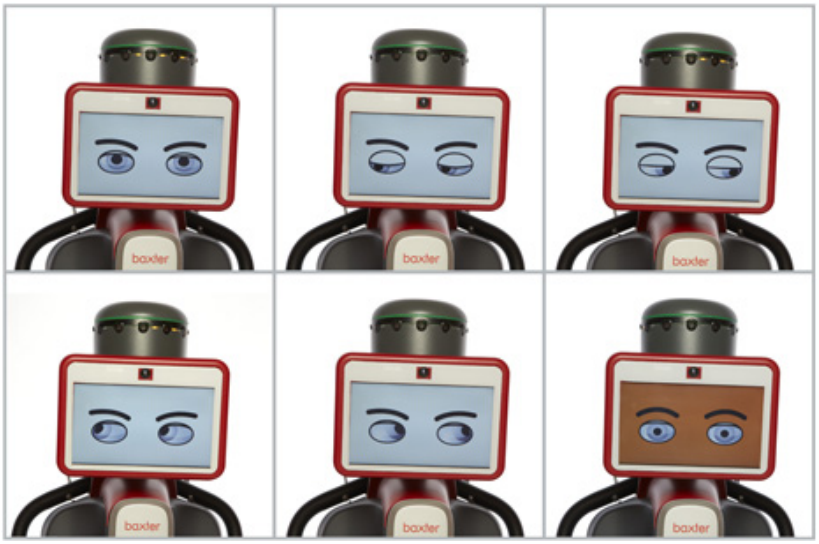


**Figure 5:** Example of gaze based communication from the robot towards the human collaborator [6].

## 2.4 Usability

Programming a robot today is expensive, takes highly trained personnel and takes a lot of time [2]. For untrained personnel to be able to program a robot to perform a task implies the use of simple and intuitive programming. This idea is called the usability by [2].
In general a lot of work has been put in making programming easier by making the programming languages more powerful and intelligent [2]. A good example of this is Labview,

which is already widely used in the industry [2]. Labview provides functional blocks which are connected by data flow lines as is shown in figure 6. This kind of programming eliminates the possibility of making syntax errors and is very visual, making it easy for a novice to use [2].

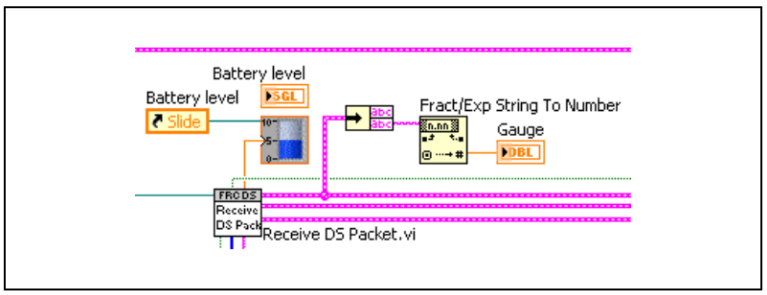



**Figure 6:** Example of Labview for robot programming [23].

Programming is made easier for a user if he can work with high level constructs [2]. In robot programming, low level programming would be to set physical positions or specific movements, while in high level programming one focusses on groups of operations or functional tasks. That's why high level programming in robotics is sometimes also called task level programming [2]. For example in low level programming the angles of the joints to move down can be programmed, while in high level programming one will have tasks like "pick object", "place object", etc. [2].

High level or task level programming requires the user to configure certain parameters of a task. For example in a pick object task the object that needs to be picked, has to be configured. In blind picking also the location of the picking has to be configured.

This configuration can have multiple approaches [2]:

- Icon-based programming: In Icon-based programming the user can drag and drop icons, which represent tasks, and symbols to program a robot. An example of this is the MORPHA, anthropoMORPHic Assistance systems, research initiative [24]. This level of programming is relatively low and leaves a lot of configuration open to the user [2], which makes it less suited for novice level programmers.

- Data Flow Diagrams: In Data Flow Diagrams the user wires functional blocks together and configures parameters of the blocks. The robotic module of Labview as shown in figure 6 is an example.

- CAD-based programming: In CAD-based programming the environment of the robot is recreated virtually and the robot can be moved by dragging the arm with a mouse or clicking on locations. Of every object a CAD model has to be made. It is very important that the virtual model is an accurate representation of the real world, because the data that is sent virtually at first will later be sent to the real world robot. An example of a CAD-based programming is ABB's RobotStudio, shown if figure 7.

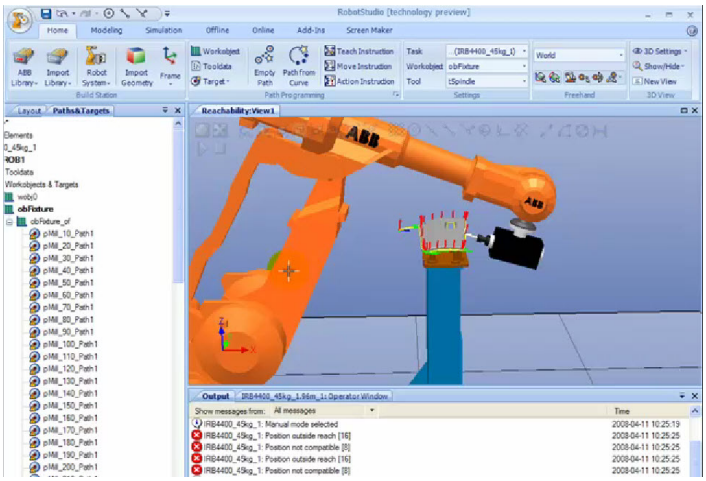


**Figure 7:** Example of CAD-based programming RobotStudio of ABB [25].

- Wizard based programming: In Wizard based programming the user is guided by a wizard to configure the parameters of the tasks. This greatly assists a novice and intermediate level programmers. The commercial version of Baxter uses a wizard to guide the user through the configuration of tasks [15].

- Lead-Through programming: In Lead-Through programming the robot is manually moved to positions by the operator. This can be a very useful and easy tool to configure spatial parameters. It requires the robot to run in a safe operating mode, where the robot is compensating for the weight of its arms. This is used in the commercial version of Baxter [15] and in the Biorob [16].

- Programming by imitation [14]: This is based on the fact that humans learn by imitation and a lot of objects imply certain manipulations [14]. For example a screwdriver implies a torsion manipulation while a hammer implies a hitting manipulation. The possibility to learn by imitation is explored in [14].

The list above is not restrictive [2]. In most robots multiple approaches are used in function of the hardware of the robot and it's application field [2]. In Baxter a combination of Lead-Through programming and Wizard based programming are used [15].

## 2.5 The Baxter robot

The Baxter robot is a robot designed by Rethink Robotics to provide in the needs explained in the motivation under section 1. Baxter was first unveiled in September 2012 by Rethink Robotics [4]. The screen on which the gaze of the robot is shown, makes this robot different from other similar robots designed for human robot collaboration in manufacturing [13].

### 2.5.1 Hardware

"Baxter consists of two, 7-degree-of-freedom arms with Series Elastic Actuators at each joint, incorporating full position and force sensing. Three integrated cameras, along with sonar, accelerometers and range-finding sensors." cited from [26].

On figure 8 the hardware of Baxter is shown. In black are all the joints of Baxter, each point of rotation is indicated with the orange arrows and these joints are all Series Elastic Actuators. The sensors are indicated in blue and the indicators are shown in green. The 2 arms are identical.

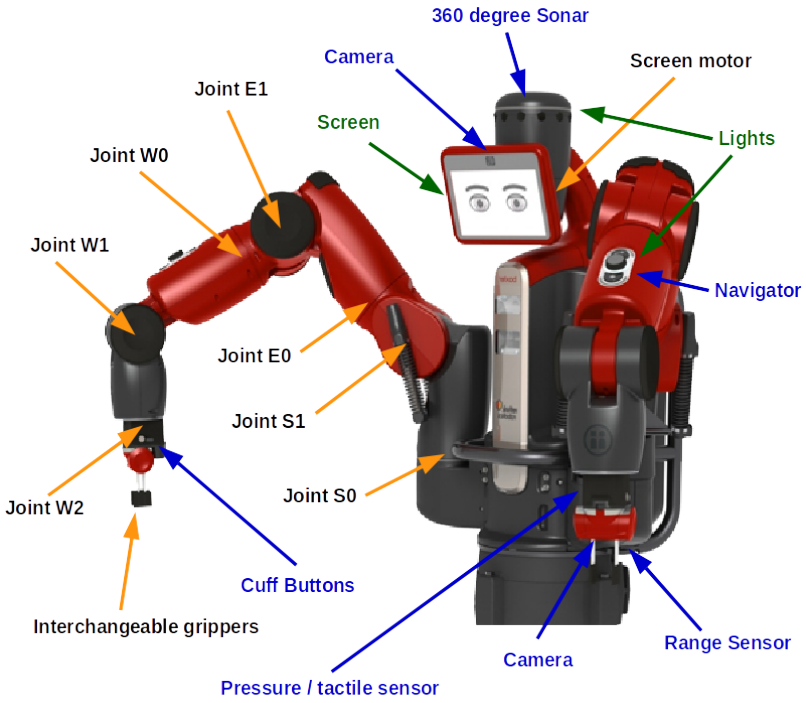



**Figure 8:** The hardware of Baxter [21].

In table 1 the naming convention of the joints is given.

| | |
|---|---|
| **S0** | Shoulder Roll |
| **S1** | Shoulder Pitch |
| **E0** | Elbow Roll |
| **E1** | Elbow Pitch |
| **W0** | Wrist Roll |
| **W1** | Wrist Pitch |
| **W2** | Wrist Roll |

**Table 1:** Joint naming convention in Baxter based on [26].

### 2.5.2 Software

Rethink Robotics delivers Baxter in 2 versions: The research version and the commercial version.

As mentioned in section 2.4, the commercial version offers a Wizard based program to the operator. It has several tasks from blind picking to camera use picking. The commercial Baxter robot has been successfully employed in manufacturing environments [15]. In

those environments a typical job was to pick items from a conveyor and place them in a container or to pick items out of a container and place them on a conveyor [15].

The research version of Baxter comes with an Software Developing Kit (SDK). The current SDK is version 1.1 which consist of the operating system Ubuntu 14.04 [26] and ROS Indigo (Robot Operating System [27]). ROS is used to form the link between all the hardware drivers of Baxter. Rethink Robotics offers on the SDK amongst others a build-in wrapping interface to ROS [28]. The programs can be written in Python or C++.

### 2.5.3 State of art Baxter Research version

Baxter is a fairly new robot and only limited research has been done on it so far. A summery of this research is made below.

*Advanced bin picking Baxter*
In [5] a bin picking task is worked out to sort parts for a simple assembly in collaboration with a human. In [5] the human, the robot and the room are monitored with 6 kinects, a 3 dimensional camera. The detection of the parts is also done with those kinects. A computer generates a plan based on the CAD of the assembly and the CAD of the parts, for the human and robot to follow. The instructions for the human are shown on a remote screen. The interaction lies in helping Baxter to detect parts and their orientation. The human uses a laser to point things to Baxter based on what is asked on the remote screen. In figure 9 the hybrid cell is shown.

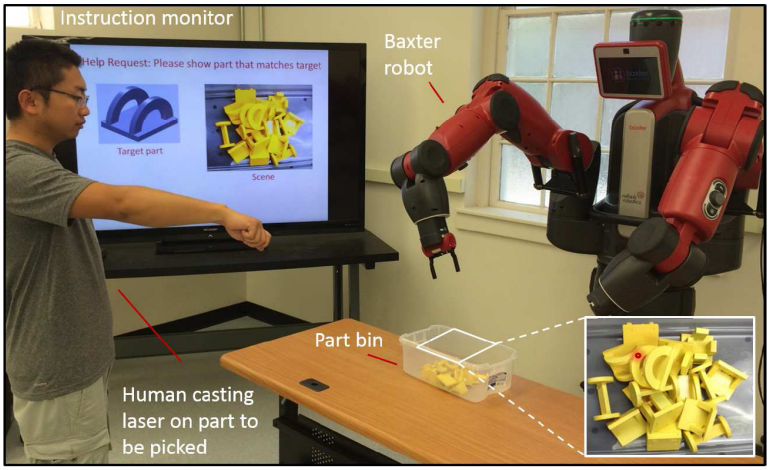



**Figure 9:** Hybrid work cell with Baxter where the human collaborator points to the object with a laser [5].

*Surgical Baxter*
In [29] the Baxter robot is used to identify surgical tools and to deliver them to a surgeon on demand. The research is done because a lot of hospitals face nurse shortages [29]. The end effector of Baxter is adapted in this scenario with a magnetic end to easily pick up the metallic tools [29]. The research in [29] is intended for small surgical operations like a mock abdominal incision surgery.

*Control officer Baxter*
In [30] the Baxter robot is used to check luggage in airports for objects that are prohibited. The Baxter robot is used for the manipulation of the luggage and the removal of the suspected object in luggage. The objects are detected with an overhead RGB-D sensor.

## 2.6 State of art of Baxter alike robots

Not a lot of collaborative robots exist today. The existing ones can be classified in several ways:

- Dual or mono arm: Baxter is a dual arm robot. YuMI of ABB is also a dual arm robot and is designed specifically for electronic assembly [13]. The iiwa of kuka is also a dual arm robot with active compliance. The Biorob is a monoarm robot. Rethink Robotics recently also came with a monoarm robot called Sawyer. Sawyer is special again because of the screen, which no other collaborative robot seems to have.
- Fix or mobile: Baxter is a fix robot, meaning it cannot move by itself. APAS of Bosh is a mono-arm collaborative robot which can move around the work floor.

The robot that is closest to Baxter is YuMI from ABB, shown in figure 10. It only got released in April 2015 and at present no papers have been written about it, that could be found. They claim that it can reach a precision of 0.02 mm for repetitive tasks [31]. Note that YuMI is designed to handle electronic components. Baxter's published precision is only 0.5 mm in its limited envelope and 5 mm over the whole range.


**Figure 10:** YuMI, the dual arm collaborative robot of ABB [31].

## 2.7 Conclusion

Human robot collaboration is a fairly recent field of research and covers many aspects. In the research done on Baxter no research has been done sofar on the usability. The goal of the thesis will be to make a behaviour framework for Baxter such that a novice could teach it to do something. For this a Lead-Through and Wizard based program will be used. It's the most user friendly and intuitive for a novice. Also the Baxter robot is ideal for this with its screen and navigator buttons on its arm.

# 3 Tools

As part of the experiment, some tools have to be made. The first tool is a classifier, able to discriminate different objects, which will be explained in section 3.1. In section 3.2 the tools will be made which are needed to find objects in camera images and to find their orientation in the image. The last section, 3.3, will deal with the inverse kinematic problem which is needed to determine the angles between the elements of the arms of Baxter to reach a desired position in space.

## 3.1 Discrimination and Artificial Intelligence

For the robot to be able to discriminate objects one from another, is an important aspect in human robot collaboration [5][14]. Without going too deep into the details of Artificial Intelligence (aka AI), the discrimination problem is explained in this section. Also a simple implementation, the one used in the experiment, is explained. This section ends with a short explanation of the naming game, which will be useful later on.

### 3.1.1 Definitions

Before explaining the discrimination problem some definitions have to be established. The definitions in table 2 are based on the lectures of [32].

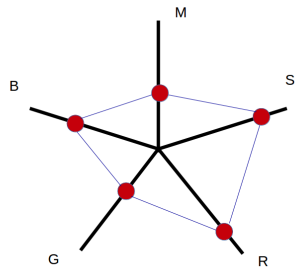
| | |
|---|---|
| **Object** | A real physical object to which a set of properties can be attributed. For example: a red disk of 5 g, a green cube of 10 g. |
| **Measurable Property** | A property of which the value can be measured with a sensor. For example: the mass of an object with a scale, the color (Red Green Blue or RGB values) with a camera, the shape with a camera, ... |
| **Object Representation** | A mathematical representation of an object. The representation is defined by a set of measurable properties in a vector space, called the vector space of properties. For example : an object with mass 4.8 g, an RGB value of [219, 101, 98] with its vector representation: (4.8, 219, 101, 98). |
| **Class** | In general the mathematical representation of an idea. In this case the representation of the idea of a certain object. This idea can be as detailed as wanted. For example: simply a reddish object or more detailed a reddish triangle-shaped object with a mass of about 3 g. |
| **Class Properties** | The class also has a set of properties, but these properties don't have fixed values. It can be assumed that the value of each property is a normal distribution [33], with a mean value, $\eta$, and a variance, $\sigma$. The properties of the class also represent a vector space of properties. |

| | |
|---|---|
| **Discriminating** | The act of finding a difference between 2 objects or classes. |
| **Classifying** | The act of finding the best match for the representation of the object at hand within the existing classes. |
| **Reinforcing** | The act of using the properties of an object to refine the properties of the class it belongs to. |

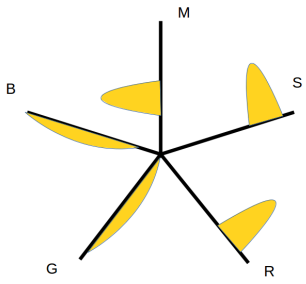
**Table 2:** Table of AI definitions.

These definitions are illustrated in figure 11. Illustration 11a shows a real object, a red disk with a certain mass.
The values measured for the different properties, are represented in a spider diagram 11b by the red circle on each of the different the axes. In the spider diagram all axes begin in 0 in the mid of the spider diagram and the extreme value at the end is 1. Therefore the values represented in it are normalized. M is the mass axis, S the shape axis, R the red value axis, G the green value axis and B the blue value axis.
In image 11c the class of a red disk with a certain mass is shown. The properties now don't have fixed values any more, but are a distribution shown in yellow. The height of the distribution is the intensity of the occurrence of that value for that property. In figure 11c this distribution is shown. The smaller the variance of a class property the more important that property is for an object to be classified under this class.
In image 11d the class of a more generic red object is shown.

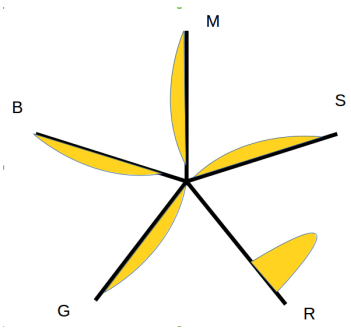
**(a)** Example of an object: A red disk [34].




**(b)** Object repesentation of the disk in figure 11a.




**(c)** Example of a class: red disk with certain mass.




**(d)** Example of another class: a red object.

**Figure 11:** Examples of the definitions of AI in table 2.

### 3.1.2 The discrimination problem and classification

The discrimination problem aims to identify different classes in a random population of objects [32]. The problem is approached by representing all objects into their vector space of properties [32]. All representations which are in the neighbourhood of each other form a cluster [32]. All the objects of one cluster can then be used to make a class [32]. The difficulty in the discrimination problem is the definition of the neighbourhood - when defined too strictly each object forms its own class and when defined too large all objects could be classified under one class [32]. This neighbourhood also depends on what one wants to achieve with it. Fortunately, in the case of human robot collaboration addressed here, the objects (or better the classes) which will be worked with, are known.

Therefore the only problem that remains to be solved is the classification. Once a classification has been made a special kind of discrimination can be done on the known classes. In this case the classes have to be kept separated, but by finding the property for which the classes are the most different, the number of properties that have to be checked to classify an object to a class can be reduced.

### 3.1.3 Classification algorithm

To classify an object into a class, the distance in the vector space of its properties to the class is calculated for all the classes. The class with the lowest distance is chosen to be the class the object corresponds to. Because the properties don't have the same units nor have the same range of values, some properties might influence this distance more than other properties. Therefore the properties will first be normalised by following formula :

$$value_{normalised} = \frac{value_{measured} - minimum}{maximum - minimum} \tag{1}$$

In this formula the minimum and maximum are defined by the range of the measuring sensor. This formula will turn every property's value into a value laying between 0 and 1.

As explained before some properties are more important for certain classes than others. Therefore the distance between the values of the object property and the mean value of the class property is first subjected to a weighing. In order to keep things simple the weighing will be chosen to be (1 - the variance of the distribution of the normalised value). Consider $dis$, being the distance between the class and the object in the normalised space of properties, $n$, the number of properties in the space of properties, $\eta_{i,class}$, the normalised mean value of property i, $val_{i,norm,object}$, the normalised value of property i of the object, and $\sigma_{i,class}$, the standard deviation of the normalised value of property i of the class, then $dis$ is defined as:

$$dis = \sqrt{\sum_{i=1}^{n}[(\eta_{i,class} - val_{i,norm,object})(1 - \sigma_{i,class})]^2} \tag{2}$$

This is ideal for calculating the distance for properties which only have one measured value, but sometimes a functional property is encountered. An example of a functional property could be the shape of an object with an R($\theta$) representation (see section 3.2). If (x,R(x)) represents the points of the functional property of an object representation and (u,R(u))

represents the points of the functional property of the class, then for every point in (x,R(x)) the distance between the point to (u,R(u)) which is the closest is chosen. A vector with all those distances is made and afterwards the total distance is taken as the norm of the vector.

### 3.1.4 Reinforcement

Reinforcing an object to a class gives the ability to determine if a property is important for a class or not. By reinforcing an object to a class one changes the mean value and the variance of the properties of the class. To avoid the need to keep track of all previous values ever to recalculate the mean value and the variance of the distribution, the new mean value en new variance are best calculated from their old values and the value of the object as given below.

Assume that the mean value, $\eta_n$, and variance, $\sigma_n^2$ of a property of the class are known for n objects and we want to reinforce object number n+1 to this class. Let $val_{norm}$ be the normalised value of the object property calculated by use of formula 1, then the following formulas are true for n $> 1$:

$$\eta_{n+1} = \frac{n.\eta_n + val_{norm}}{n+1} \tag{3}$$

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2 + (n-1)\sigma_n^2 + n(\eta_n - \eta_{n+1})^2}{n} \tag{4}$$

### 3.1.5 The naming game

The naming game is a virtual experiment used in AI to research the origin and evolution of language [32][35]. In a naming game a group of agents classify a set of objects and give each class a name independently from each other [32]. Once this is done the agents play a game with each other.

Two agents play the naming game, one is the listener and one is the speaker [32]. First the speaker names a class he knows. Then the listener will try to find the class he thinks is mentioned with that name and point to an object that could be classified under this class. The speaker classifies the object pointed by the listener. If the class, the object classifies to, is the same as the one initially named, the game is a success. If not, the speaker will now point to an object that classifies under the first named class. The listener will then reinforce the class closest to the object with this new object [32]. The aim of the game is to see how the names and the classes changes over the number of played games and if they converge [32].

The game can have many alternatives [32], one of which is interesting for human robot collaboration: The special case were there are only 2 agents and one is always the speaker. Taking the human as the speaker and the robot as the listener the intelligence of the human is used to define the classes of the objects used.

## 3.2 Computer Vision

Making a computer see and recognise shapes on an image is a difficult task [36][37]. The field of Computer Vision has emerged as a discipline in itself and is strongly connected to the fields of mathematics and computing sciences [36]. It also has connections to the field of physics, psychology of perception and neuro sciences [36]. The field of Computer Vision has evolved in trying to recreate the way animals see [36]. This has 2 reasons:

1. To better understand how biological vision works [36]. The idea of trying to better understand the biological side is a general idea in Artificial Intelligence [32].

2. The attempts to ignore biological vision and reinvent an own vision have not been very successful [36].

Despite this difficulty of making a computer see, there has been a lot of success in the field of Computer Vision [36]. An example of this are the cars which can ride on regular roads and rough terrain through the use of camera's and the analysis of real-time three-dimensional dynamic scenes [36]. Also the tracking of humans in hybrid work cells by use of 3D cameras like done in [10] is a good example of the success of computer vision.

The first huge task is detecting shapes in a camera image [36][37], afterwards the system has to be able to match the shape found to an object or class.

### 3.2.1 Finding shapes

There are several ways to find shapes in an image, one is by grey scale, another one is by oriented line segments [36][37]. The use of oriented line segments has several advantages over the others [37]. Oriented line segments are reasonably invariant for rotation of the object and therefore can reliably reconstruct an object [37]. It's also thought that the human visual system uses oriented line segments to process images [37].

In this project the canny edge detector has been used. It produces a grey scale image [38]. On this image a set of transformations are implemented, the most important one is in taking the first derivative [38]. Afterwards a threshold with hysteresis is used to determine the edges [38].Though oriented line segments are better, the canny edge detector is widely used and readily available in the Opencv library [39], which is an open source library for Computer Vision algorithms [39].

Baxter adjusts it's camera to the light intensity. Using a black worktable for instance will make the camera very sensitive resulting in less contrast. To be independent from the colour of the worktable, the objects used will be white and have a pronounced black border. This will make sure that the contrast to detect the objects is maximal. Consequently the edge detector will work more reliably.

In figure 12 the objects used in the experimental set-up are shown. On figure 12a object 1 is shown and on figure 12b object 2 is shown. The objects are designed to be easily graspable by Baxter.

**(a)** Object 1.

**(b)** Object 2.

**Figure 12:** Objects used in experimental set-up.

### 3.2.2 Shape recognition

Once the shapes in an image are detected the recognition of the shape can start. An important aspect of Computer Vision is to find a representation of geometrical shapes in such a way that they are invariant when seen from different view points [36][37]. This means that the representation should be invariant for translation, rotation and scale [37]. This would prevent the need to describe all relevant views of an object [37]. Besides the invariant properties the representation should also be compact but have enough descriptive power to allow discrimination between dissimilar objects [37]. The representation should also be robust to all types of noise expected from the input data [37]. In [37] several of those representations have been described. The main advantages and disadvantages of some of the representations described in [37] are summarised below.

| **Shape Skeleton:** The skeleton shape is obtained by repeatedly thinning until it becomes a unit pixel width network. Leaving a purely topological description of the shape. | |
|---|---|
| **Advantages:** | - Skeleton shapes are robust to random noise.<br>- Skeleton shapes possess the translation and scale invariant properties. |
| **Disadvantages:** | - Purely topological descriptions are too ambiguous. For example a square and a circle are both described by a point.<br>- Sensitive to occlusion. Occlusion is the phenomenon where boundaries of two different objects can not be distinguished from one another. |

Shape skeleton can be a good representation when observing humans with a 3D camera.

| | |
|---|---|
| **Moment Invariants (or Hu moments):** The moment of a shape in a binary M by N image is defined as : $$u_{pq} = \sum_{j=0}^{M-1} \sum_{i=0}^{N-1} i^p j^q f(i,j) \quad (5)$$ Where f(x,y) is the pixel intensity at coordinate (x,y) and p+q is the order of the moment. To make the representation independent of translation the moments are taken with respect to the centre of the area of the shape. Moments on their own do not uniquely represent an arbitrary shape, but a set of functions based on the moments can determine such. | |
| **Advantages:** | - Possesses the translation and rotation invariant properties.<br>- Only a number of scalars are needed to describe the object. |
| **Disadvantages:** | - Calculation of the moments involves finding the centre of the shape which is sensitive to noise, occlusion and poor segmentation.<br>- Doesn't possess the scale invariant, but this can be added by normalisation. |

| | |
|---|---|
| **R($\theta$) plots:** R($\theta$) plots are formed by plotting the distance of each point of the boundary of the shape with respect to the centre of the shape in function of the angular displacement around the boundary. The invariance in orientation can be found by shifting the profile over the theta axis and the invariance in scale by shifting it over the r axis. An example of an R($\theta$) plot can be seen in figure 13. | |
| **Advantages:** | - The orientation of the object can be found.<br>- Invariant for orientation and scale.<br>- Simple concept. |
| **Disadvantages:** | - Not invariant for translation.<br>- R and $\theta$ are calculated with respect to the centre of the shape, which is sensitive for errors.<br>- For all but the most simple shapes the plot becomes multivalued, for some $\theta$ there could exist multiple R values. |

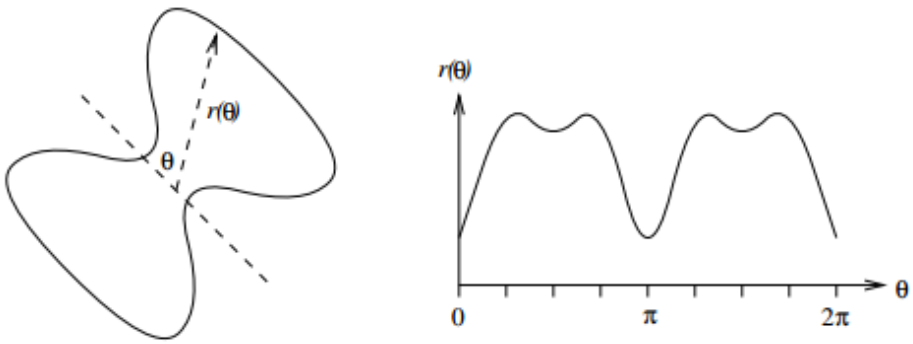



**Figure 13:** Example of R($\theta$) plot [37].

| **S($\phi$) plots:** On S($\phi$) plots the tangential orientation of each boundary point in function of the distance, s, travelled on the boundary is shown starting from an arbitrary boundary point. In figure 14 an example of an S($\phi$) representation can be seen. | |
|---|---|
| **Advantages:** | - Can cope with occlusion.<br>- Not multivalued.<br>- Invariant for rotation. |
| **Disadvantages:** | - Not invariant for scale (but can be normalized).<br>- Not invariant for translation.<br>- Difficult to accurately determine the distance travelled along the boundary. |

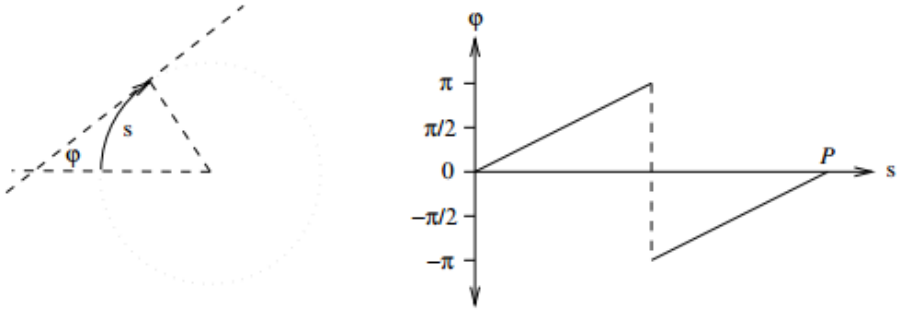



**Figure 14:** Example of S($\phi$) plot [37].

| **2D Projective Invariants:** Based on 2 principles :<br>1. Preservation of points of a tangency on a 2D planar object under different projections.<br>2. The mapping of any 4 points from one plane to another is sufficient to determine the transformation matrix T which fully defines the transformation. | |
|---|---|
| **Advantages:** | - Invariant for scale, rotation and translation. |
| **Disadvantages:** | - Very complex to implement. |

Sometimes one representation is not sufficient to completely describe an object [37]. Combinations of multiple representations can be used [37].

To quickly recognise an object it is better to use a representation which has all 3 invariant properties : invariant for translation, rotation and scale. When working with robots however, the orientation of the object with respect to the grippers is an important property. Therefore 2 representations will be used here during the project :

1. Invariant moments, in specific Hu moments. The invariant moments are used because of their invariance to rotation and translation. Because of this, they can quickly tell from an image if a shape is worth inspecting or not.

2. Once an object has been recognised, the R($\theta$) representation will be used to determine the orientation of the object. The R($\theta$) representation is used mostly because of it's simplicity. Using R($\theta$) representation makes it very easy to find the orientation of the object.

### 3.2.3 Hu Moments

Hu derived 7 invariant rotational moment functions which form a suitable shape representation vector [37] :

$$M_1 = (u_{20} + u_{02}) \tag{6}$$

$$M_2 = (u_{20} - u_{02})^2 + 4u_{11}^2 \tag{7}$$

$$M_3 = (u_{30} - 3u_{12})^2 + (3u_{21} - u_{30})^2 \tag{8}$$

$$M_4 = (u_{30} + u_{12})^2 + (u_{21} + u_{03})^2 \tag{9}$$

$$\begin{aligned} M_5 = (u_{30} - 3u_{12})(u_{30} + u_{12})((u_{30} + 3u_{12})^2 - 3(u_{21} + u_{03})^2) \\ +(3u_{21} - u_{03})(u_{21} + u_{03})(3(u_{30} + 3u_{12})^2 - (u_{21} + u_{03})^2) \end{aligned} \tag{10}$$

$$\begin{aligned} M_6 = (u_{20} - u_{02})((u_{30} + u_{12})^2 - (u_{21} + u_{03})^2) \\ +4u_{11}(u_{30} + 3u_{12})(u_{21} + u_{03}) \end{aligned} \tag{11}$$

$$\begin{aligned} M_7 = (3u_{21} - u_{03})(u_{30} + u_{12})((u_{30} + u_{12})^2 - 3(u_{21} + u_{03})^2) \\ -(u_{30} - 3u_{12})(u_{21} + u_{03})(3(u_{30} + u_{12})^2 - (u_{21} + u_{03})^2) \end{aligned} \tag{12}$$

where $u_{pq}$ are defined in formula 5.

Matching between different shapes is done by matching these 7 Hu moments [37]. OpenCv has a function to calculate the Hu moments [39].

### 3.2.4 Matching R($\theta$) representation

To find the orientation of an object, once it has been recognised, involves matching 2 R($\theta$) representations to one another. The 2 shapes are the ones of the recognised object, the measured shape, and the one of the class the object was classified under, the reference shape. The best shape is found by calculating the distance between the reference shape and all the shapes obtained by shifting the measured shape over every boundary point. The distance between the shapes is calculated as explained in section 3.1.3.

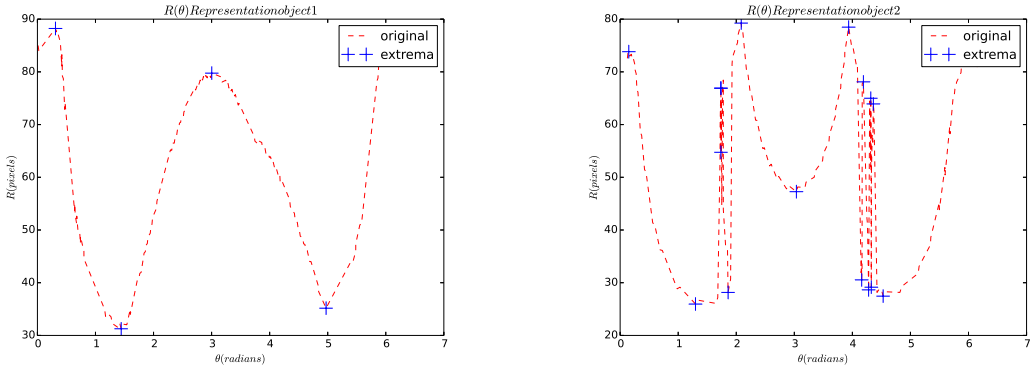
Matching 2 profiles to one another can take much time. The time is roughly equivalent to the number of points in the profile squared because the function has to shift the shape over every point in the boundary, calculate the distance between all the points in the boundary and afterwards calculate the global distance between the measured shape and the reference shape.

Therefore in this project only the local minimum and maximum of the shapes will be used to find a match. Reducing the points the shape has to be shifted over and the amount of distances to be calculated drastically, allows to gain calculation time.

Unfortunately there is always noise on measured data making it difficult to find the local extrema [40]. In [40] an algorithm is described to find the local extrema based on

the knowledge of the accuracy of the measurement. Because the accuracy of the edge detection is not easily known, the accuracy has been taken as 5% of the difference in the maximum value of R and the minimum value of R, with R being the distance to the center of the object.

Using this algorithm on the shapes of figure 12 produces the shapes in figure 15. The red line is the shape with all the points, the blue crosses represent the local extrema. As can be seen from the drawing the number of points is reduced drastically. This is of course paid for with more ambiguity. Yet, this is not a problem because the recognition of the shape is done by Hu moments. We can also see the multivalued problem with the R($\theta$) representation in object 2. Around $\theta = 2$ and around $\theta = 4$ it can be seen in image 15b that the shape is submitted to heavy vibration. This is because in fact there are 2 boundaries at this angle.




**(a)** Object 1. **(b)** Object 2.

**Figure 15:** R($\theta$) representations.

Further reduction of the calculation is done by orientating the training shape to have the maximum in 0 and then only shifting the measured shape over the local maxima which values are close to the one of the maximum of the measured shape.

## 3.3 Inverse kinematic solvers

Inverse kinematic is a common problem in robotics [41]. It consists in finding the right angles of the joints for the end effector to have a known position in space [41]. A position in space is defined by 6 parameters: the 3 spacial coordinates and the 3 orientation coordinates. Most of the time, several solutions are possible and in most cases no analytical solution nor a straight forward way to find the right angles exist [41]. The problem is therefore often solved by iteration.

### 3.3.1 Inverse kinematic solver on Baxter

There is an inverse kinematic solver which can be used with Baxter [42]. This inverse kinematic solver is an iterative solver [42]. The solver searches for an optimised solution where the action space of the end effector is as large as possible.

When moving from one point to another point nearby, unfortunately some joints have to rotate a lot to have the best optimised solution. This results in a very unpredictable trajectory, which is not very desirable. For example: Consider the situation where the end effector is positioned right above the object it wants to grip. The only movement that it still has to do, is the movement downward. If during this movement the orientation of the end effector changes completely, there might be a chance that the end effector pushes the object away or changes the orientation of the object. Sometimes the iterative inverse kinematic solver of Baxter doesn't find a solution for a position that should be reachable. That's why an alternative inverse kinematic solver was searched for.

Openrave [43] also offers a state of the art iterative inverse kinematic solver that could work. However, the actual version of Openrave, that is 0.8.2, was not compatible with Ubuntu 14.04. Therefore a simple analytical inverse kinematic solver was developed.

### 3.3.2 Analytical inverse kinematic solver

In figure 16 the right side of Baxter is shown with every frame of reference used in this part. The red axes are the local x-axes, the blue are the local z-axes en the green are the local y-axes. The z-axes in the joints are defined such that the links connected by the joint rotate around these local z-axis and the x-axes are defined such that when the angle of the joint is zero the link would align with it. The end effector is indicated with EE.

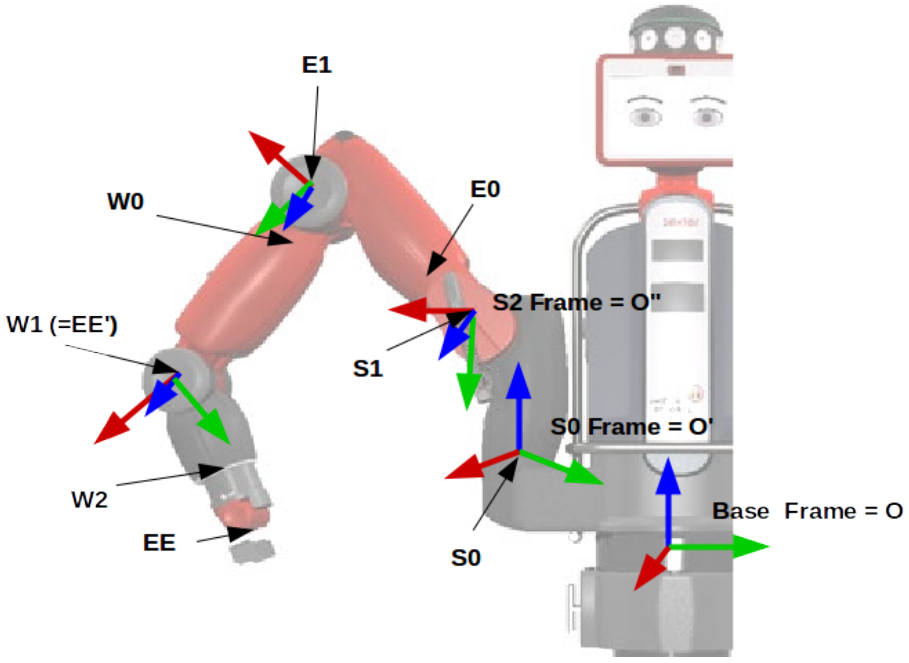



**Figure 16:** References and frames of Baxter (Image adapted from [44]).

Baxter has a 7 degrees of freedom arm. But when looking at an object one can best maintain a position perpendicular to the plane of interest, which fixes 2 degrees of freedom. To simplify things, this plane has been taken horizontally because most of the time as well as in this experiment, Baxter will work on a worktable placed horizontally in front of him.

Besides this, one also wants the camera in it's arm, close to the end effector, to be faced to the outside, so that the shadow of Baxter itself doesn't affect the view. This takes away another degree of freedom.
Those 3 degrees of freedom will be located in the wrist, joints $W_0$, $W_1$ and $W_2$, and determine the orientation of the end effector.

Since from $W_1$ the links are fixed downwards to be perpendicular to the horizontal plane, the coordinates of EE' are fixed with respect to the coordinates of EE in the reference frame O and are given by following formulas:

$$\begin{aligned} x_{EE'} &= x_{EE} \\ y_{EE'} &= y_{EE} \\ z_{EE'} &= z_{EE} + L \end{aligned} \tag{13}$$

with L being the length of the last link, between the joint $W_1$ and the location of the end effector, EE.

This transformation is shown in figure 17, where the joints are represented by yellow circles and the links by orange lines It schematically visualises on the left the projection on the xy-frame of the local reference frame O and on the right the projection on the Rz-frame of the local reference frame O. R is the axis going through the local reference the projection of the end effector on the xy-frame.

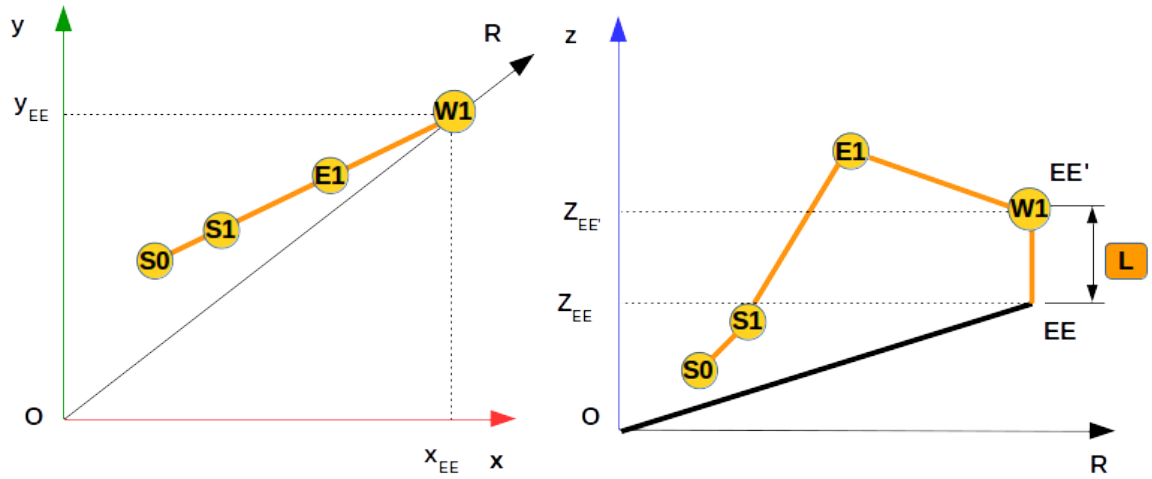



**Figure 17:** Transformation of EE to EE'. On the left the projection in the xy-frame of O, on the right the projection on the Rz-frame of O.

Fixing joint $E_0$ to a value such that the $E_1$ only rotates around an axis perpendicular to the z-axis of the base frame, consumes a 4th degree of freedom.

The 3 remaining degrees of freedom will be used to control the position of the end effector. The position of the end effector is expressed in the reference frame O located in the base.
Let $X(= [x,y,z])$ be the location of the end effector, EE, in the reference frame of the base. Transforming it to the reference frame O', the reference frame in which the value

of $S_0$ is defined, includes a fixed translation, $X_{trans}(=[x_{trans}\ ,\ y_{trans},\ z_{trans}])$, and a fixed rotation, $Rot$.
Let $X'(=[x',\ y',\ z'])$ be the location of the end effector, EE', in the frame O', then:

$$X' = Rot.(X - X_{trans}) \tag{14}$$

In this reference frame O', the rotation angle $S_0$ can be found by following formula:

$$S_0 = \arctan 2(y', x') \tag{15}$$

This transformation is shown in figure 18 .

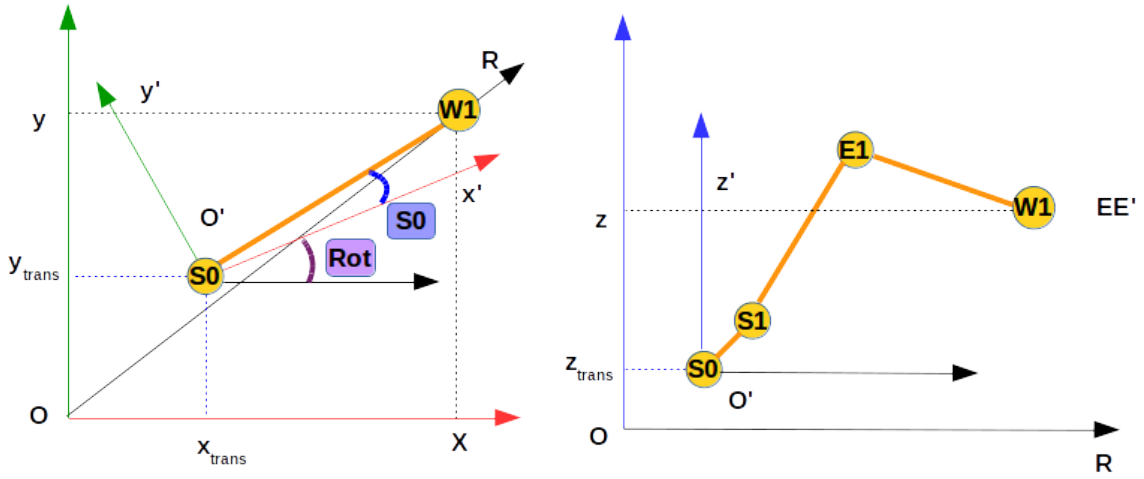



**Figure 18:** Transformation of O to O'. On the left the projection in the xy-frame of O, on the right the projection on the Rz-frame of O.

Assuming the joint $S_0$ has turned over the angle $S_0$, expressing the point of the end effector, EE', in the reference frame O", the reference frame where joint $S_1$ is defined, includes again a fixed translation, $X'_{trans}$, and rotation, $Rot'$. Using a similar formula as 14 the coordinates $X''$ of EE' can be found in the reference frame O", this is shown in figure 19.

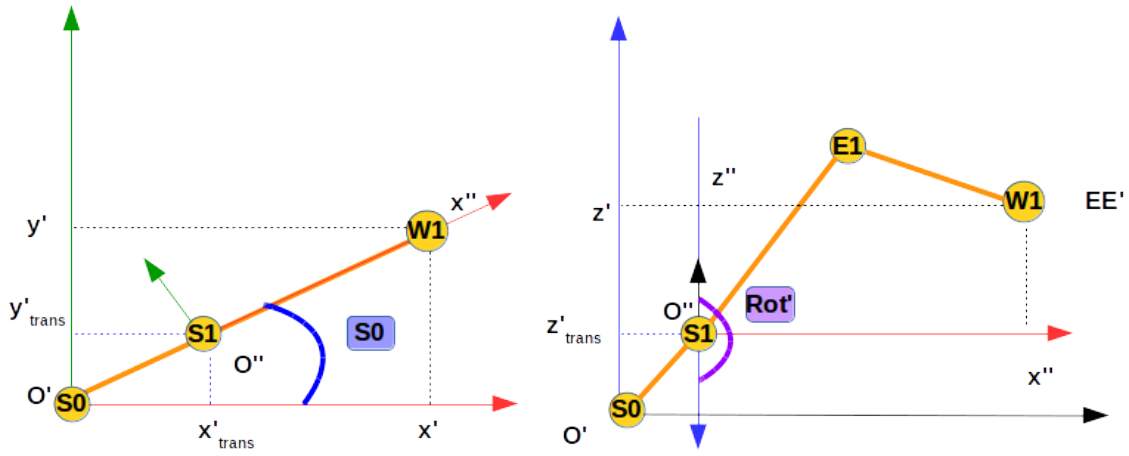



**Figure 19:** Transformation of O' to O". On the left the projection in the xy-frame of O', on the right the projection on the Rz-frame of O'.

The problem is now reduced to the known 2 degrees of freedom arm in a plane for which the solutions are widely described. From [41] the general geometrical solution for the position is given by following angles:

$$E = \arctan 2(s_2, c_2) \tag{16}$$

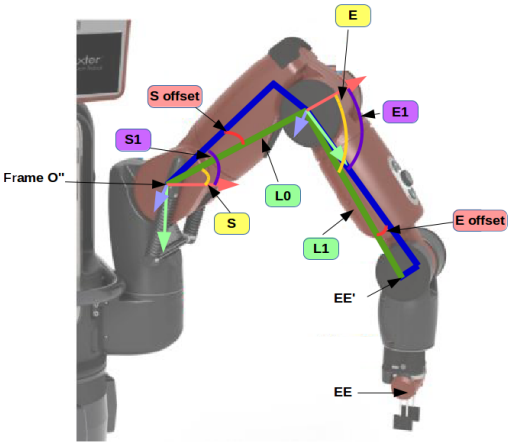
$$S = \arctan 2(y'', x'') - \arctan 2(L_1 s_2, L_0 + L_1 c_2) \tag{17}$$

with $x''$ en $y''$ being the position of joint $W_2$ with respect to the reference frame shown on figure 20a and with $c_2$ and $s_2$ defined as follows:

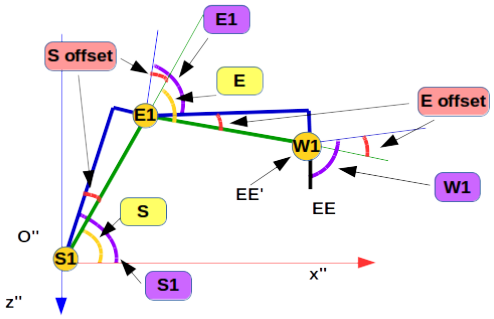
$$c_2 = \frac{x''^2 + y''^2 - L_0^2 - L_1^2}{2L_0 L_1} \tag{18}$$

$$s_2 = \pm\sqrt{1 - c_2^2} \tag{19}$$

The parameters of this formula are shown in figure 20a. In formula 19 $s_2$ is taken positive to have the arm in the position like shown in figure 20a. The green line shows the total length of the links. But the robot's architecture follows the blue lines. Therefore the angles E and S as calculated above, in yellow on the figure, have a fixed offset, shown in red, bringing them to angles $E_1$ and $S_1$.




**(a)** Definitions of the parameters of the IK on Baxter. (Image adapted from [26])




**(b)** Scheme of the offsets in the arm of Baxter.

**Figure 20:** Projection on z"x"-frame of O" of the offsets.

The formulas for the angles $S_1$ en $E_1$, in purple on image 20a are then:

$$S_1 = S + S_{offset} \tag{20}$$

$$E_1 = E - E_{offset} + S_{offset} \tag{21}$$

Because the way Baxter is assembled, the values given in the specifications [26] are not

exact [45]. After the assembly of Baxter the real values are measured and stored in Baxter's URDF file. An URDF file or Unified Robot Description Format file is XML format for representing a robot model [46] and is used in ROS to have a unified way of representing robots. Many ROS packages use this file to work with. The exact translations and rotations between the different reference frames are found by the use of TF. TF is a package of ROS which keeps track of the different reference frames of the robot in function of the time.[47].

All angles are then defined by following formulas:

$$S_0 = \arctan 2(y', x') \tag{22}$$

$$S_1 = S + S_{offset} \tag{23}$$

$$E_0 = 0 \tag{24}$$

$$E_1 = E - E_{offset} + S_{offset} \tag{25}$$

$$W_0 = 0 \tag{26}$$

$$W_1 = \frac{\pi}{2} - E_1 + S_1 \tag{27}$$

$$W_2 = \pi - S_0 - Rot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{28}$$

with $S_0$, $S_1$ and $E_0$ as calculated above. To keep the joint of $W_1$ at an angle of $\frac{\pi}{2}$, the angles of $E_1$ and $S_1$ have to be compensated. For the camera to be faced outward, the joint $W_2$ has to be turned over an angle of $\pi$. $W_2$ has to be compensated for $S_0$ too and the last term comes from the rotation between the frame of reference O and O'. Thus if $S_0 = 0$ there is still a rotation.

In order not to harm the robot and to always find the best possible solution the calculated angles are limited to the angles described in [26].

In case a position is not reachable the analytical inverse kinematic solver will give a solution for which the height has been adapted to reach the desired position.

### 3.3.3 Testing the inverse kinematic solver

To test the inverse kinematic solver several programs have been made. One program consists in manually moving all the joints to the desired location through the use of a GUI (Graphical User Interface). Another program is to change the position of the end effector through the use of a GUI. The last program consists in testing the inverse kinematic solver over a large area.

The first two programs were made for safety issues. If the inverse kinematic solver would not work correctly and the robot would be directly connected to this inverse kinematic solver, the robot could make unexpected movements and this could be harmful for the robot or even worse somebody nearby the robot could be hurt. Therefore the angles were calculated through the use of the inverse kinematic solver and then the calculated angles would be manually set in order to always have control over the robot. Once this was successful, meaning that the end effector was more or less at the right position, the second program could be used to test several positions manually.

The last program consisted in automatically moving through a cube in the workspace of Baxter while providing an error map. An example of such an error map is shown in figure 21. On this figure the trajectory is shown in the bottom graph and the error on the end position is shown in the top graph. The colors of the coordinates are conform to the colors of the figures in section 3.3.1, x in red, y in green and z in blue. The coordinates are measured in the reference frame O of figure 16. On the top graph of figure 21 the error in z shows large spikes. Those spikes in the beginning of the graph are due to the fact that the joint $S_1$ is not able to reach the angle asked for. The spikes at the end are due to the fact that the desired position is not reachable and the analytical inverse kinematic solver than first tries to reach the best position for x" in reference frame O" of 16. Except for those spikes, the errors are generally centred around zero.

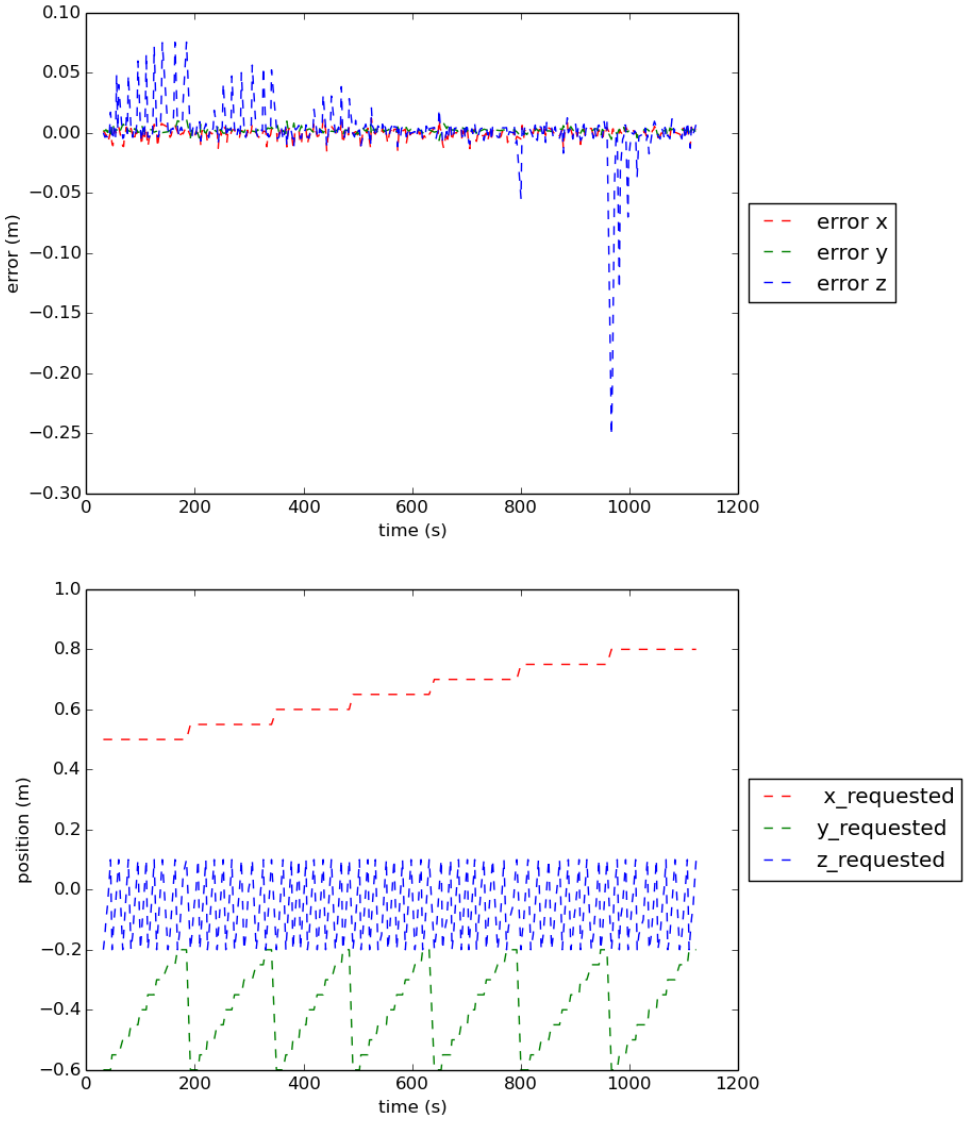



**Figure 21:** Error map for the analytical IK. The error is shown in the top graph and the trajectory in the bottom graph.

In figure 22 the same trajectory has been used to make an errormap of the iterative inverse kinematic solver which comes with Baxter. In the top graph of figure 22 one can see at the end large errors in position. Those are due to the fact that if the iterative inverse kinematic solver doesn't find a solution it doesn't move. In the rest of the trajectory the error also stays well centred around zero.

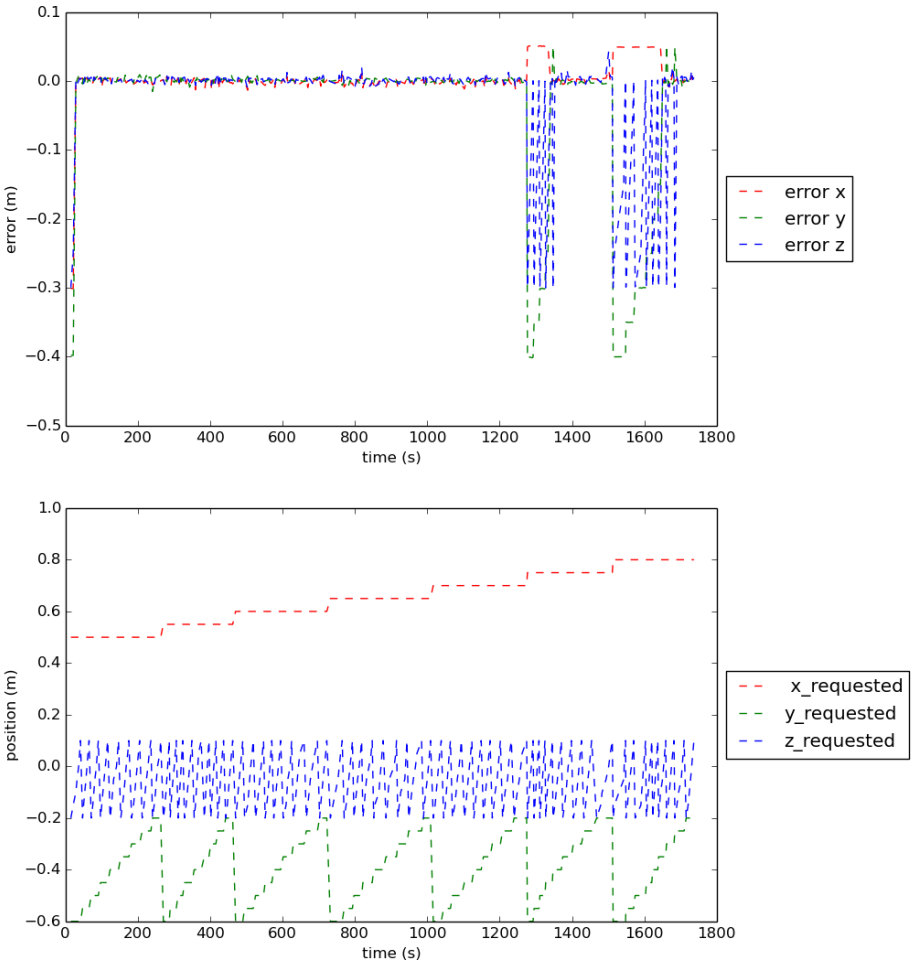
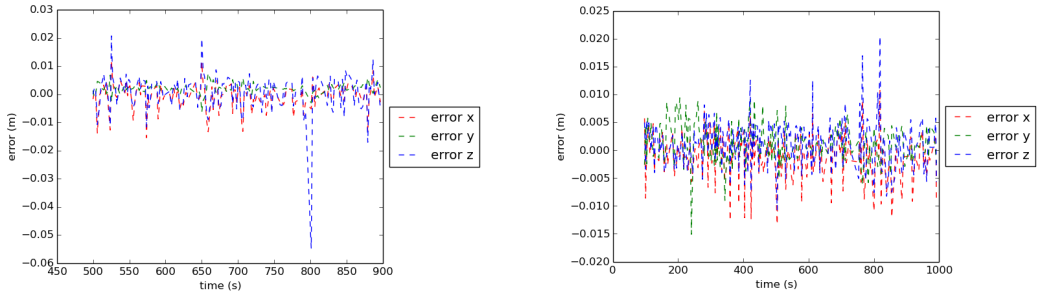



**Figure 22:** Error of the iterative IK solver.

From the top graphs of figures 21 and 22 one can see that the analytical inverse kinematic solver has a greatly reduced working space (till 1200 s) than the iterative one (till 1800 s). The analytical one is faster though, reaching all the points in 1122 s, while the iterative solver needs 1734 s to reach all the points.

Figure 23 zooms in on the errors of both solvers in the part where the error is centred around 0. For the analytical solver this is from 500 s to 900 s in figure 23a, going for x from 0.65 m to 0.75 m , for y from - 0.6 m to -0.2 m and for z from -0.2 m to 0.1 m. For the iterative solver it is from 100 s to 1000 s in figure 23b, going for x from 0.55 m to 0.65 m , for y from -0.6 to -0.2 and for z from -0.2 m to 0.1 m.




**(a)** Zoom on errors of errormap for analytical IK in figure 21

**(b)** Zoom on errors of errormap for iterative IK in figure 22

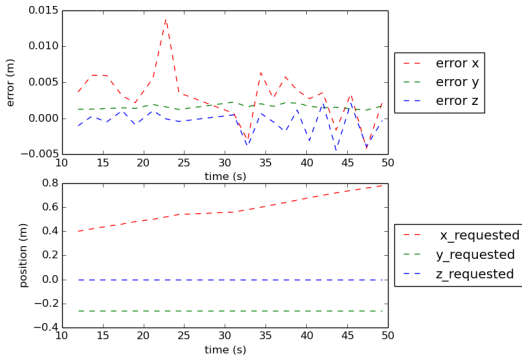
**Figure 23:** Zoom on errors of errormaps for analytical and iterative IK.

For both solvers one can see in figure 23 that the error is mostly contained between $\pm$ 10 mm with sometimes a spike going to 20 mm and even one going to 55 mm for the analytical one. This last spike is most probably coming from a non-reachable position for the analytical solver. In table 3 the mean, the standard deviation and the maximum are summarized for the errors shown in figure 23. For the analytical solver the spike of 55 mm was ignored, since coming from a known cause (non-reachable position).

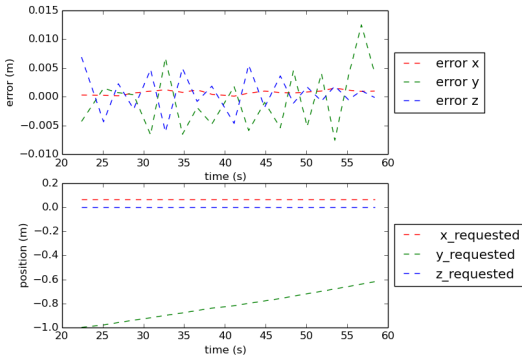
| | Analytical | | Iterative | |
|---|---|---|---|---|
| | Error (mm) | Max error (mm) | Error (mm) | Max error (mm) |
| Error x | -0.6 $\pm$ 4.3 | 15 | -1.1 $\pm$ 3.9 | 13 |
| Error y | 2.0 $\pm$ 1.7 | 7.5 | 1.1 $\pm$ 3.3 | 15 |
| Error z | 0.8 $\pm$ 7.0 | 21 | 1.3 $\pm$ 4.1 | 20 |

**Table 3:** Mean, std dev and maximum error of the errors shown in figure 23.

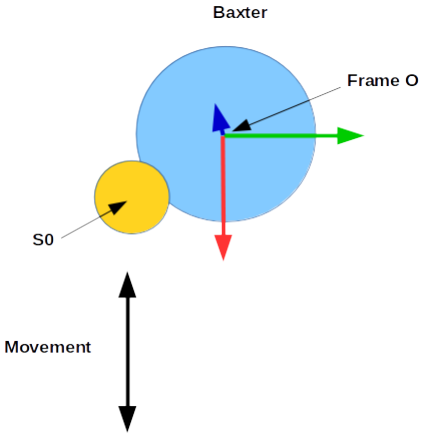
From table 3 one can conclude that for both solvers the standard deviation of the error is in several mm (3 to 4 mm), while the max error goes up to 20+ mm. The mean error itself fluctuates around 1 to 2 mm for both solvers. The 2 mm for the mean error on y for the analytical solver could come from a systematic error in the calculation. This could be another offset in the angles not taken into account in the analytics. To find the cause of the errors for the analytical solver, further test have been done here below.
In figure 24 the errormaps of the analytical inverse kinematic solver are shown for movements on a straight line. In figure 24a the trajectory is chosen such that the robot would move his arm perpendicular to Baxter at the position of the joint $S_0$ as shown in figure 24c - x movement. In this figure one can see that the error in y, in green, is more or less stable while the errors in x and z (red and blue) fluctuate around 0. When doing a similar test, but then with a movement like shown in 24d - y-movement - a similar result can be seen in figure 24b, but with the error of x and y switched (x more or less stable now).



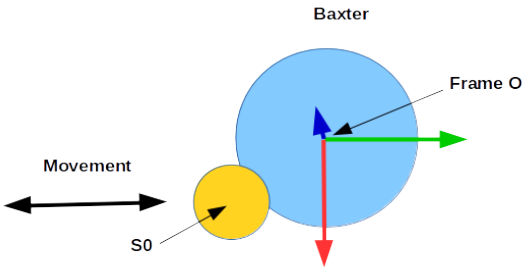

**(a)** Errormap of a movement in the x direction as shown in 24c.




**(b)** Errormap of a movement in the y direction as shown in 24d



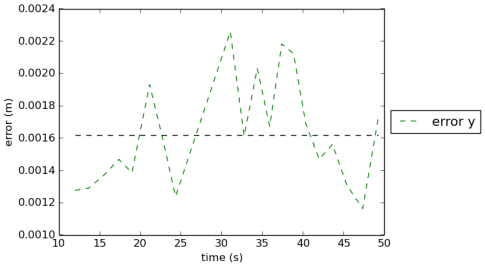

**(c)** Movement in the x direction.



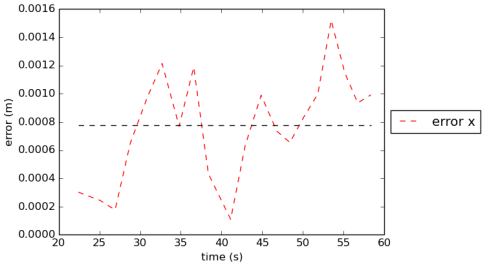

**(d)** Similar movement in y direction.

**Figure 24:** Errormaps of the analytical IK solver on straight lines.

In figure 25a the error of y in figure 24a is zoomed in and the error of x in figure 24b is zoomed in in figure 25b. One can see that the error in y fluctuates around 1.6 mm $\pm$ 0.6 mm (maximum error) and the error in x fluctuates around 0.8 mm $\pm$ 0.7 mm (maximum error). From both offsets on can conclude that there is still a systematic error in the calculation of $S_0$. This systematic error of 1.6 for y and 0.8 mm for x could come from an offset in the angles in the z"y"-plane of the reference frame O", which was not taken into account.



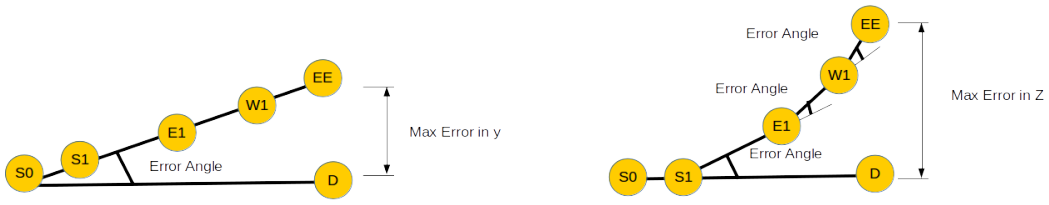

**(a)** Error of y in figure 24a.




**(b)** Error of y in figure 24b.

**Figure 25:** Errormaps of the analytical IK solver on straight lines.

These maximum error fluctuations of 0.6 mm for y and 0.7 mm for x around their systematic offsets are within the precision of Baxter, which is calculated here below. The makers of Baxter claim that the error on the angles is 0.1° [26]. The worst error on the y position in the case of the movement in figure 24c, is obtained by stretching the arm completely and multiplying it by this angle of 0.1°, which is visualised in figure 26a. This gives a calculated maximum error of 1.6 mm on the y position. Taking into account that the last joint $W_1$ is bending down in the project set-up, the resulting effective maximum error to compare against, is 1.4 mm. In the drawings of figure 26 EE is the maximum position of the end effector and D is the desired position for the end effector.




**(a)** Maximum error on y.

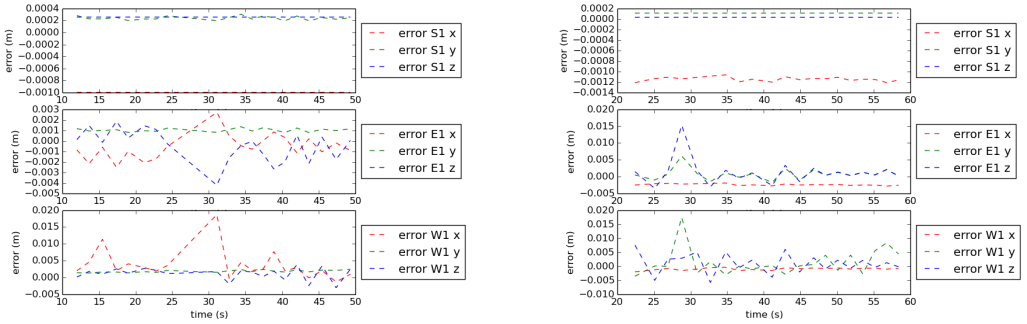
**(b)** Maximum error on z.

**Figure 26:** Maximum errors on position of Baxter

The maximum error for the z position can be seen in figure 26b and is equal to 3.0 mm, while the maximum error on the x position will be the vector combination of the errors in y and z, which is 3.3 mm.
Clearly those errors are not achieved in practise with the analytical IK solver.

In figure 27 the error on the positions of the joints $S_1$, $E_1$ and $W_1$ (= EE') are shown for the trajectories shown in figure 24. In the top graph of figure 27a and figure 27b one can see that the position of joint $S_1$ is constant for both movements. The error in x is constantly around 1 mm, which could come from an error of the measured value in the URDF (Unified Robot Description Format) of Baxter. This also explains why the error of x in the movement shown in figure 24d was 1 mm smaller than the error of y in the movement shown in figure 24c. It also indicates that the offsets of the errors in x and y are coming from further down in the arm.

On the middle graph of figure 24a the error of the position of joint $E_1$ is shown. This error can only be caused by joints $S_1$ and $S_0$. Since the top graph shows that the error caused by joint $S_0$ is fixed in space and fluctuates very little, the error on the position of joint $E_1$ is mostly caused by joint $S_1$. In this graph the error in the z positions is 0 $\pm$ 2 mm with a maximum error of 4 mm and the error in x is -1 $\pm$ 2 mm again with a maximum error of 4 mm. The offset in x clearly comes from the error on the position of joint $S_1$ shown in the top graph. Taking this error to calculate the error in the angle of $S_1$ one can find that the error in the angle is 0.3° for 2 mm and even 0.6° if taking the maximum error measured. Calculating this back to the end position the error in joint $S_1$ is responsible for 4 to 8 mm of the total error.




**(a)** Errormap of the positions of joints $S_1$, $E_1$ and $W_1$ (= EE') for the trajectory as shown in figure 24a.

**(b)** Errormap of the positions of joints $S_1$, $E_1$ and $W_1$ (= EE') for the trajectory as shown in figure 24b.

**Figure 27:** Errors on the positions of the joints.

The error of the position in joint $W_1$ is given in the bottom graph of figure 24a. From this already 4 to 8 mm are caused by the error in $S_1$. The rest of the error is caused by joint $E_1$. Though difficult to estimate from this graph since the error of joint $S_1$ influences the error a lot. When ignoring the 2 larger spikes the error seems to be contained within 5 mm. This means that the error caused by $E_1$ is roughly 1 to 2 mm, leading to a similar error for the angle as for joint $S_1$ given the links between $S_1$ and $E_1$ as well as between $E_1$ and $W_1$ have more or less the same length.

The error in the end position is mostly caused by the error in joint $S_1$. In figure 28 the precision of the joints $S_1$ in function of the time is shown and several steps. The precision is only measured down to 10 seconds and the dashed black line indicates the error of $\pm$ 0.1° . In this graph one can see that the settling time of the controllers is very large before reaching the claimed error of 0.1° for steps of 5 cm. Depending on the step demanded it takes between 1 second, for the black curve, to 3 seconds for the purple curve, to even 7 seconds for the red and yellow curves. Notice that a deviation of 0.2 degrees in joint $S_1$ cause an error for z of 6 mm.

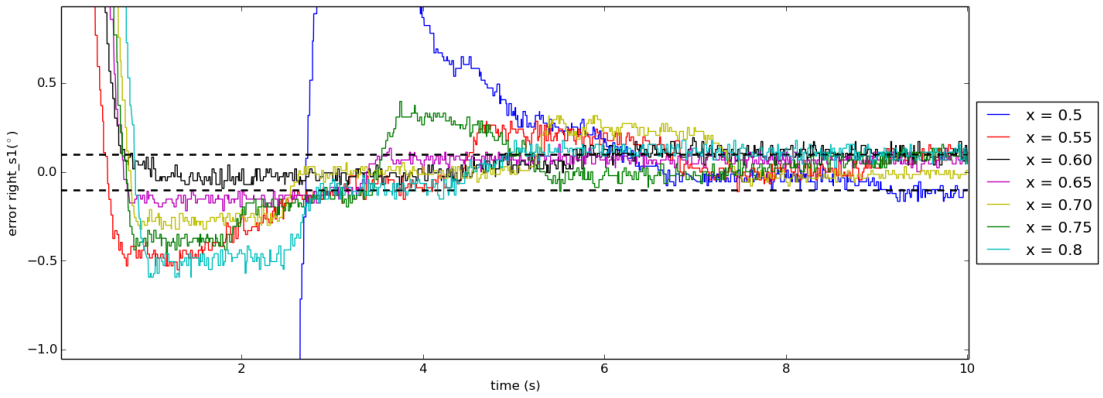



**Figure 28:** Step responses of joint $S_1$.

Similar results could be obtained for joint $E_1$ and $W_1$. In those results it can be seen that

$E_1$ has the longest time to settle. But it has less influence on the end position then joint $S_1$.

The movement shown in figure 28 can be seen physically. When the robot has to hover over an object 5 mm above the worktable he first crashes onto the table, after 5 seconds the arm starts to lift from the table and after another 5 seconds the robot is hovering 5 mm above the table.
The robot also shows a visible hysteresis movement. This movement is probably caused by the springs in the SEAs.

This means that no inverse kinematic solver will perform better then the given errors in an acceptable time if the controllers of the joints are not adapted. It could help if instead of setting the positions of the joints and then moving at maximum speed to these positions (as is done here in this project), a predefined velocity profile would be given where the torque in the joint changes continuously.

### 3.3.4 Conclusion of the inverse kinematic solver

The errormaps on the position of the analytical inverse kinematic solver and the iterative inverse kinematic solver show similar errors and error fluctuations. Those are caused by the performance of the controllers in the joints.
Both solvers have their advantages and drawbacks. Those are listed in table 4.

| **Analytical inverse kinematic solver** | **Iterative inverse kinematic solver** |
|---|---|
| + Faster calculation | ++ Greater workspace |
| + Finds the "best" solution for not reachable positions | - Strange movements |
| - - Systematic error for the calculation of $S_0$ giving a constant error of 2 mm. | - - Does not always find a solution for reachable positions. (special case: when only having to turn the last joint $W_2$, the iterative solver fails regularly) |

**Table 4:** Conclusive table on the inverse kinematic solvers.

The last disadvantage of the iterative solver weighs a lot in an application where the orientation is important. Therefore the analytical inverse kinematic solver will be used during the rest of the project.

## 3.4 Conclusion

Several tools have been built in this chapter which are needed for the project. In section 3.1 a classifier was made to be able to discriminate different objects one from another on the basis of the measurable properties. In section 3.2 tools have been made to detect objects from images and to determine their orientation. In section 3.3 an analytical inverse kinematic solver was built and was chosen above the iterative inverse kinematic solver for the rest of the project because the iterative one does not always find a solution for all reachable positions.

# 4 The program

With the tools of section 3 in hand the main task of exploring a HRC can be started. First the concept that is aimed for, will be elaborated in section 4.1. In this section the main control architecture will be defined. In section 4.2 the architecture of the wizard used, will be explained. This will be followed by an explanation of the different sub-procedures to come to the proof of concept in sections 4.3, 4.4, 4.5.

## 4.1 Concept

The goal of this project is to give a proof of concept for a human robot cooperation cell. In this concept the robot would ultimately be able to independently perform a given manufacturing task. The cooperation between human and robot lies in the human learning the task to the robot and in returning signals from the robot to the human to provide feedback.

In such a set-up, a human could supervise a cell of multiple robots performing a task and this could be several robots performing the same task or multiple robots working together on one task.

The flowchart in figure 29 describes the procedure to work with the robot and establishes the cooperation cell. The full procedure has 3 smaller sub-procedures, in light blue. Every sub-procedure has a deliverable, in dark blue, which is necessary to go to the next sub-procedure.

The first procedure is to learn the robot all objects necessary to complete the task and to turn them into classes. The second procedure is to learn the robot the task it has to execute. The final procedure is the main operation where to robot executes his task. This will be done independently except when the robot runs into trouble, then the robot will have to ask the human supervisor to help him out.

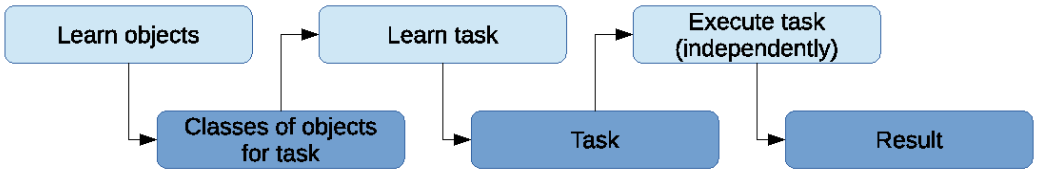



**Figure 29:** General flow chart.

A combination of a lead-through and wizard based approach will be used as is the case on the commercial version of Baxter. This seems the most intuitive approach for non trained personnel.

Several manufacturing tasks can be selected as the task to be performed to this proof the concept. As described in 2.5.3 most tasks today are pick and place actions, where the orientation of the object to pick is not important (except for [5]) and the place where the object had to be set does not have to be precise.

The task chosen as case in this project would be for the robot to play a shape game, as can be seen in figure 30. This would be a first step towards an assembly.

**Figure 30:** Shape game [48].

## 4.2 The wizard

The wizard will be a GUI on the screen of Baxter and will be navigable through the use of the navigator buttons on the arms of Baxter (figure 8). The first screen is shown in figure 31.

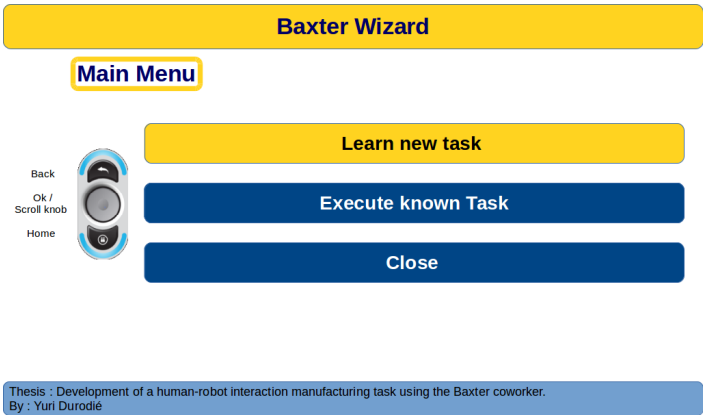



**Figure 31:** First screen of the wizard.

The wizard will have a tree architecture. This tree is shown in figure 32. The wizard will guide the user through all the steps and help him with the correct configurations.

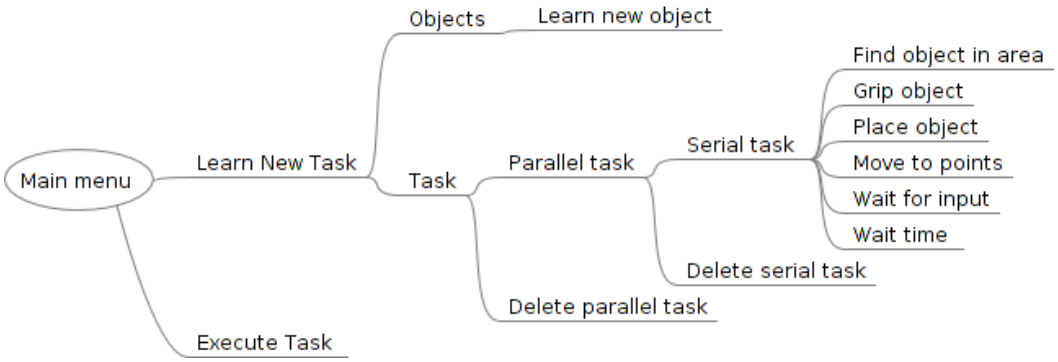



**Figure 32:** Tree architecture of the wizard.

First the user can choose between a predefined task to be executed or to learn a new task. When choosing the new task option the user will be guided through the learn objects procedure, see section 4.3, and afterwards through the learn task procedure, see section 4.4. When choosing the execute task procedure the user will have to choose from previously defined tasks. Then the robot goes into automatic mode.

## 4.3 Learn objects procedure

The learn objects procedure, shown in figure 33, asks the human supervisor if all classes of objects needed for the task have been learned. If all objects are already known, they can be loaded from a database, otherwise the robot will have to learn the objects. In this case, the supervisor will first learn the robot where to grab the object. Humans can easily see where to place their fingers/hand to take an object. Also the strength of the gripping will be determined by the human. The intelligence and insights of the human will be conveyed to the robot in the robot human collaboration model. Afterwards the robot will examine the object and map the relevant and measurable properties. Finally the robot will keep the class of the object. If all the classes are known the robot will discriminate the classes to find the properties that are the most relevant for the classes.

As a last step, a naming game could be played with the robot as described in section 3.1. In this game the supervisor will place different objects sequentially before the robot upon which the robot will identify the object and tell the supervisor which class it belongs to. If the answer is correct the robot will reinforce the class with the object. If the answer is wrong the supervisor will pick the right class upon which the robot will reinforce this new class with that object. If no mistakes are made the discrimination was successful. If the naming fails a new discrimination should be made. At the end of the learn object procedure a library with the classes of all necessary objects should be obtained.

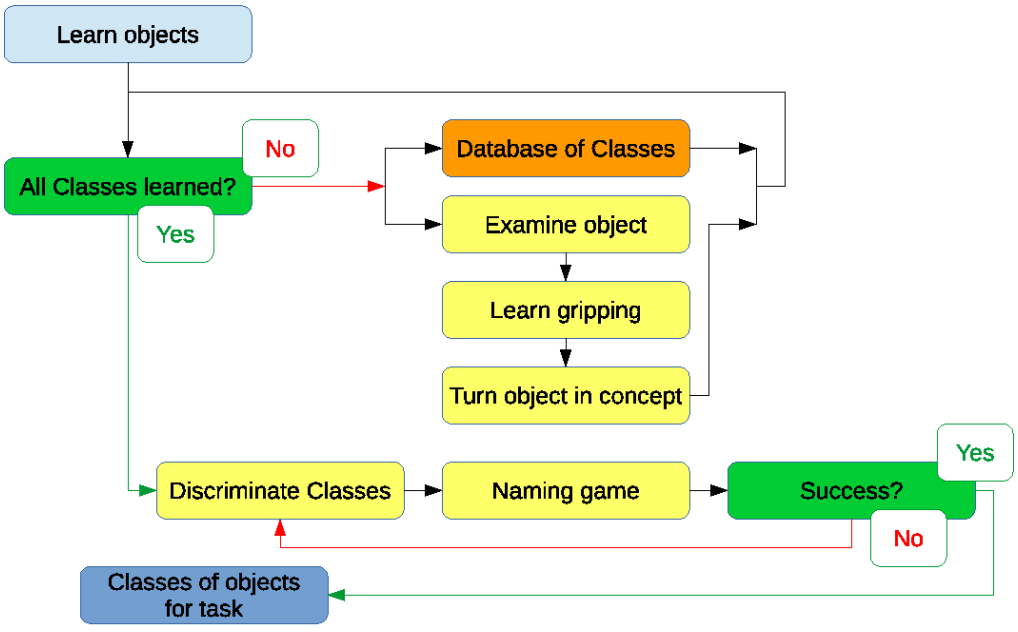



**Figure 33:** Flowchart of the learn objects procedure.

### 4.3.1 Implementation

Baxter will start by asking the supervisor to help him to grip the object. Gripping an object is not a straight forward task for a robot. By using the buttons on the cuff the gripping force can be configured by the supervisor

When the supervisor is happy with the force and location to grip the object, he presses on the ok button on the arm of Baxter. Baxter will move the object to his starting position and place it on the table. At this point Baxter will examine the orientation of the object as well as the location of gripping. Then Baxter will add the gripping location, the gripping orientation and the gripping force as a property to the class.

The gripping orientation is determined by the polar representation of the object, as explained in section 3.2. The gripping angle of the class is determined by the angle at which the highest radius is found with respect to the angle of the gripper of Baxter.

For this orientation of the object, the gripping location is determined as the difference of current location of the gripper while Baxter is holding the object and the location of the gripper once Baxter has released the object, moved slightly upwards and centred its camera view on the centre area of the object.

Afterwards Baxter will measure all properties of the object and start to reinforce the class with those properties. For the implementation in this project only 2 objects were used, the objects shown in figure 12 in section 3.2. The properties measured where the 7 invariant Hu moments and the mean radius of the object in polar representation.

The discrimination will determine if the properties chosen are enough to determine the class of the object. This is not yet implemented.

## 4.4 Learn task procedure

In the learn task procedure, shown in figure 34, the supervisor can choose a known task from a database or define a new task. Defining a new task is done by using predefined subtasks, which can be seen as the toolbox. A task can be defined by placing several subtasks behind one another in a given sequence.

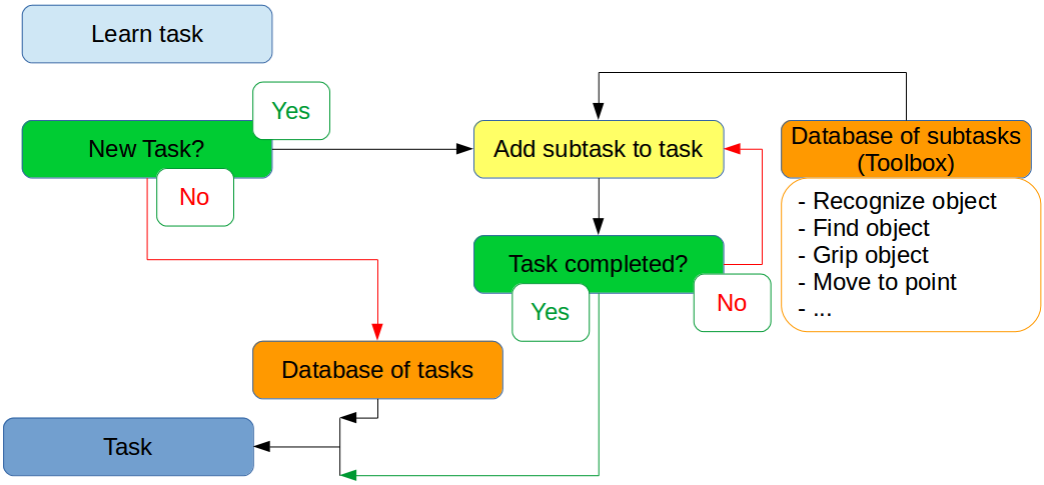



**Figure 34:** Flow chart of the learn task procedure.

A task existing of a sequence of subtasks is called a serial task in this project. To introduce a more complex task, parallel tasks have been introduced. A parallel task exists of several serial tasks.

The usefulness of this configuration is illustrated by following example: When 2 objects have to be sorted, object 1 in box A and object 2 in box B, it doesn't matter if first object 1 is found and placed in box A and then object 2 is found and placed in box B. For this a parallel task could be made with the search and placing of object 1 in box A and a second parallel task with the search and placing of object 2 in box B. But for instance when a gift box is filled with one of each of the two objects, it is important that after object 1 has been found and placed in the box, object 2 is found and placed in the same box. Therefore a serial task would be better suited. First finding and placing object 1 and afterwards finding and placing object 2.

### 4.4.1 Implementation

Only the tools necessary to fulfil the task of the experiment have been developed. The task at hand could look like in figure 35. First Baxter will search in a given area for an object. If it finds the object, it will pick up the object. Once the object has been picked up it will move to a set of defined points and finally place the object at its end location. Afterwards the task is repeated.

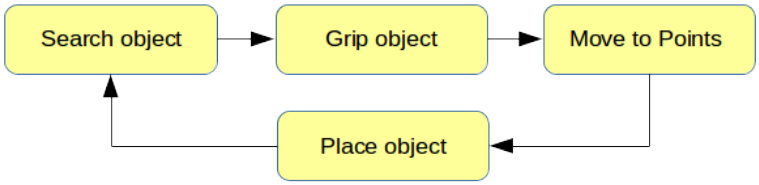



**Figure 35:** Flow chart of a possible task.

In the next subsections the configuration of the subtasks of Baxter are explained. The sentence *'when the user is ok with something'* indicates that the user confirmed the configuration by pressing the ok button on the arm of Baxter.

*Find object*
In the find object subtask the supervisor will be asked to define the search area in which he wants to find a certain object. The configuration wizard will first ask to show Baxter 2 opposite corners of the rectangular search area. Afterwards Baxter will ask the supervisor to place the object he has to search for underneath the camera.

*Grip object*
Baxter will ask the supervisor to place the object he has to grab underneath his camera. When the supervisor is ok with the object, Baxter will proceed to pick up the object targeted for.

*Place object*
Baxter will place the object at the specified location. When Baxter has no object gripped, a special screen will tell the supervisor to first grip an object. This screen is shown in figure 36. Several of such pop-up screens are placed to help the supervisor and to prevent him from making logical mistakes.

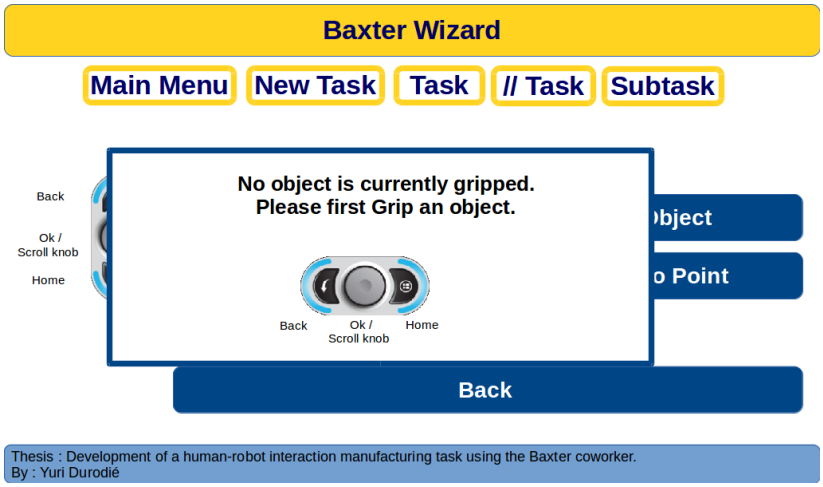



**Figure 36:** Special screen to help supervisor: Case of placing an object when no object has been gripped.

*Move to points*
In the move to points subtask, the supervisor is asked to move the arm of Baxter to the

location he wants the arm to be. The location and orientation of the end effector are saved. The number of positions are unlimited and can simply be added by pushing a button on the cuff. When the supervisor is ok with the points, the robot will replay the movement by going to the different points.

*Wait for input*
In this configuration a button on Baxter can be chosen as an input needed before continuing. This can be useful when an action of the human operator is needed before proceeding with the rest of a sequence of manipulations.

This subtask has been added in order to do the experiments in section 5.1.

*Wait time*
The number of seconds Baxter has to wait at a position can be configured using the scroll knob. This can be useful when for example a bar code on an object has to be scanned. The robot needs to hover an object over a laser and stand still for several seconds in order for the scanner to read the bar code.

## 4.5 Execute task procedure

Once the task has been configured, Baxter can start the execution of the task. The flowchart of this procedure is shown in figure 37. Baxter will start by executing subtask by subtask. If there is no problem with the execution of the subtask, Baxter will proceed to the next subtask. If there is a problem, Baxter will stop and ask the human operator for help.

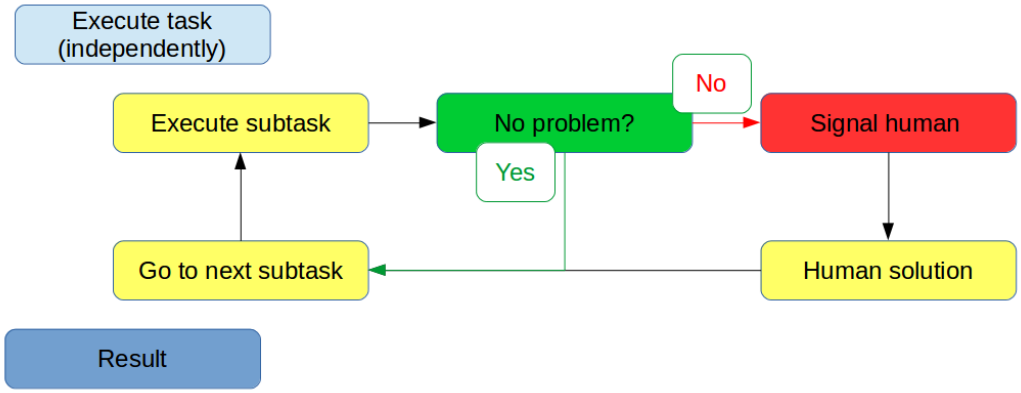



**Figure 37:** Flow chart of the task execution procedure.

In the subtask the success or failure of this subtask was implemented. But there was not enough time to define a procedure to signal the human collaborator of the problem and interact with him to solve it.

The current implementation of the execute task procedure runs through all the tasks and when an error occurs it simply stops.

## 4.6 conclusion

In this section the wizard based configuration tool was developed.

In the learn object procedure the most important aspect is that Baxter learns from the operator how to grip an object. This consists in the orientation of the object during gripping, the position the grippers have on the object as well as the force exerted by the grippers on the object. This is a good example of how the human problem solving skills can be used in combination with the repeatability of the robot co-worker.

In the learn task procedure, 6 subtasks have been defined that can be placed in a serial or parallel configuration. The 6 tasks are :

- Find object: Search of an object in an area.
- Grip object: Grip an object.
- Place object: Place an object at a given location and orientation.
- Move to points: Define a path the end effector has to follow.
- Wait for input: Define a button on Baxter for which Baxter has to wait before continuing with the next task.
- Wait time: Wait a given number of seconds before continuing with the next subtask.

In the execute task procedure an architecture has been proposed to signal the human cooperator when there is a problem, but no time was left to implement this procedure.

# 5 The experiments

Making roughly 4200 lines of code, of which about 2300 are involved in the end program of this project, work together takes quite some time to debug, with unfortunately no guarantee all bugs have been 'killed'. Nevertheless several experiments have been done, however regularly showing new bugs. In this section only the last results are presented, those for which can be assumed no major bugs were still 'alive'.

The first experiment consists in validating the gripping location and the orientation algorithms. The second experiment consists in validating the usability.

Both experiments were done with the analytical inverse kinematic solver.

## 5.1 Validating the gripping location and the orientation algorithms

The experiment to validate the gripping location and the orientation algorithms aims to measure several parameters which can give an idea of the performance of the algorithms used.

The parameters that have been measured during this experiment are shown in table 5 :

| **Measured parameter** | **Explanation** |
|---|---|
| $\Delta x$ (mm) $(= x_{learned} - x_{gripped})$ | The difference between the location Baxter was learned to grip, $x_{learned}$, and the location it did grip the object, $x_{gripped}$. This gives a measure of the performance for learning the gripping position. This parameter is subject to a lot of external errors, one of the most important ones is the performance of the inverse kinematic solver. As explained in section 3.3.3 one can expect already an error of $\pm$ 5 mm with maximum errors going to 2 cm for the positioning. |
| $\Delta\theta$ (°) $(= \theta_{actual} - \theta_{calculated})$ | The difference in the actual orientation of the object, $\theta_{actual}$, and the orientation Baxter calculated for it, $\theta_{calculated}$. This gives a measure of the performance of the R($\theta$) plots and the algorithms to detect the orientation from those plots as explained in section 3.2. |
| $t_{DO}$ (s) (= time Detecting Object) | The time it takes to find an object when the object is underneath the camera, which gives a measure for the detection and discrimination algorithms described in section 3.1 and 3.2. |
| $t_{FO}$ (s) ( = time Finding Orientation ) | The time it takes to make and analyse an R($\theta$) plot of an object, and find the orientation of the object using the properties of the class of the object. This gives a measure of the performance or the speed of the algorithms to detect the orientation explained in section 3.1. |

| | |
|---|---|
| $t_G$ (s)<br>(= time Gripping) | The time it takes to grip the object successfully, meaning gripping the object somewhere that it is clamped between the grippers without changing its orientation significantly (less then 5°) as shown in figure 40, given the object has been detected and classified. $t_G$ gives a measure of performance of the total gripping algorithm. |
| $t_T$ (s)<br>(= time Total) | The total time it takes to find and classify an object, to find the orientation of the object and to successfully grip the object, given the object was placed in front of the camera. $t_T$ gives a measure of the performance of the total algorithm. |
| S (%)<br>(= success rate) | The rate of success of effectively gripping an object over all the attempts, which is a measure for the overall performance to grip an object. |

**Table 5:** Overview of measured parameters in the validation of the location, orientation and gripping algorithms.

All the times are represented in the time line of figure 38

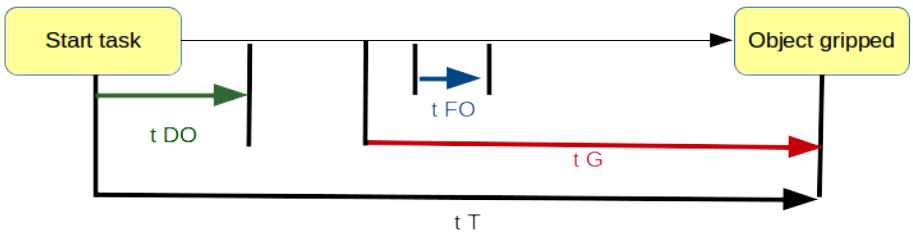


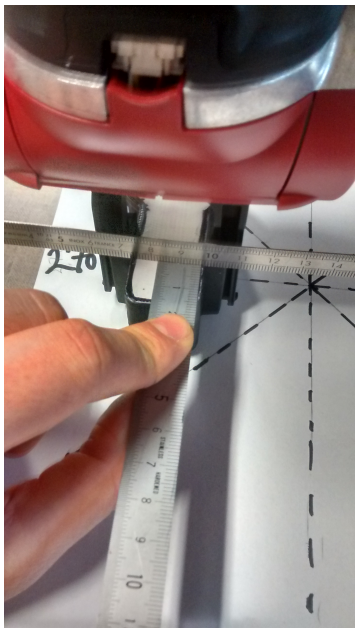

**Figure 38:** Time line of the measured times.

### 5.1.1 Experimental setup

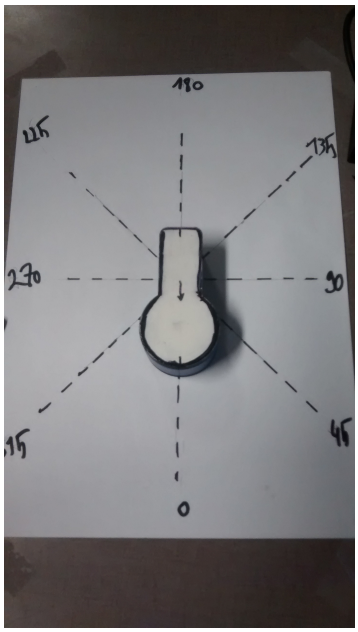
To measure all the times t given in table 5 as precisely as possible the measurement was hard coded in the programs. Given the time precision of the computer is much higher then the expected times measured during the experiment, being in the order of magnitude of several milliseconds, one can assume that the error of the measuring device is not significant.

To measure $\Delta x$, the distance from one end of the object to the beginning of the gripper is measured, for both $x_{learned}$ and $x_{gripped}$, as can be seen in figure 39a.

$\theta_{actual}$ was measured by placing the object on a $360°$ map such that the orientation of the object is known. This can be seen in figure 39b. Assuming that the objects are placed carefully and that the error of this placement is not significant in comparison to $\Delta\theta$, $\theta_{calculated}$ will be an output of the program. This implies that an orientation for which the joint $W_2$ is not able to turn (range from -3.04 to + 3.04 radians) will not be taken into account.



**(a)** Set-up for measuring $\Delta x$



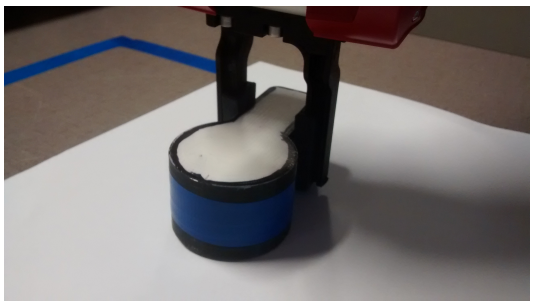

**(b)** Set-up for measuring $\Delta\theta$

**Figure 39:** Set-up for experiment 1.

The orientations $\theta$ for which the parameters in table 5 will be measured, are 0°, 45°, 90°, 135°, 180°, 225°, 270° and 315°. For every orientation a number of attempts are made to grip the object, and this for both objects 1 and 2 shown in figure 12. For every successful attempt all parameters are measured, while for failed attempts only some parameters were measured depending the type of failure that occurred.

### Successful attempts

A successful attempt is defined as an attempt where the object was gripped, without intervention, in such a way that if the object would be placed, the right orientation would be kept. This is shown in figure 40. In figure 40a a typical successful attempt is shown and in figure 40b also shows a successful attempt because the orientation of the object is not deviating a lot. A deviation of less than $\pm 5°$ on the end orientation was still categorised as a success.



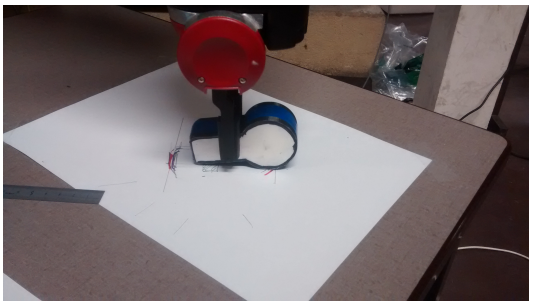
**(a)** Example of a typical successful gripping.

**(b)** Successful: Objects end orientation deviated less than $5°$.

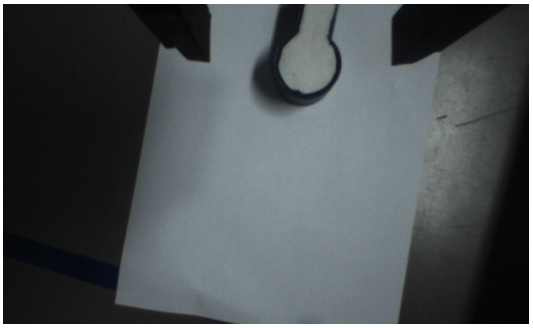
**Figure 40:** Gripping success

### Failed attempts

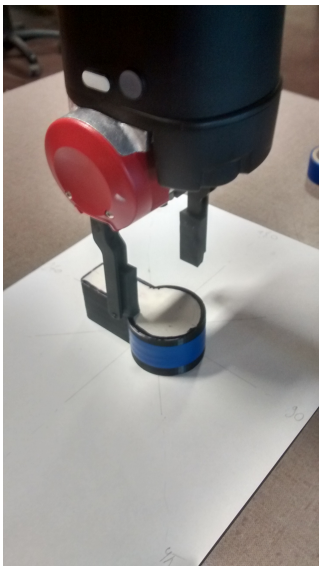
During the test different kinds of failures could be distinguished. Those failures are shown in figure 41. Depending the type of failure some parameters were still measured and kept.



**(a)** Failure type 1: Gripped object, but object flipped while gripping.



**(b)** Failure type 2: Object out of field of view after orientation.



**(c)** Failure type 3: Gripper blocked against object.

**(d)** Failure type 4: End orientation deviated more than $5°$.

**Figure 41:** Types of frequent failures.

In figure 41a a failure (type 1) is shown where the robot, while moving down to the object, hits the object in such a way that the object flips. For this type of failure all the times and $\Delta\theta$ have still been measured.

In figure 41b a failure (type 2) is shown where after the robot oriented its arm to grab the object, the object is no longer in the field of view of its camera. This comes from the offset of the camera with respect to to the center of rotation and this is maximal when the camera has to turn over $180°$. Though this error could be easily solved by changing the orientation algorithm, such that the position is also adjusted with respect to the orientation and the offsets of the camera, this has not been done during this project. During the tests, the object was slightly moved manually such that it got back in de field of view of Baxter. For this type of error all measurements were kept. Notice that the human intervention implies a small time loss, that is also captured in the measurements.

In figure 41c a failure (type 3) is shown where the gripper gets blocked against the object. For this type of failure only $\Delta\theta$ , $t_{DO}$ and $t_{FO}$ were measured.

In figure 41d a failure (type 4) is shown where the object was gripped, but the end orientation deviates too much to be seen as successful. A deviation of more than $\pm 5^{\circ}$ on the end orientation was seen as a failure. For those failures only the measurement of $\Delta x$ was not made.

During the experiments a 5th type of failure occurred sometimes. The failure was that from time to time the gripper failed to open 100% resulting in less clearance than normal. The clearance obtained was most of the time just enough to release the object. Knowing that the difference in distance between the closed and open gripper is only 3 cm and that the object itself has a thickness of 2.5 cm this could cause problems with the 5 mm accuracy of the inverse kinematic solvers.

### Program

For this experiment the control architecture described in section 4 has been used. The flow of the tasks is given in figure 42. In figure 42 the yellow blocks represent the subtasks of the wizard as explained in section 4.4.1, the green blocks represent the human inputs. The task itself consisted in waiting for an input, so that the human operator had enough time to collect all the data and correctly place the object. Once the input had been given (pushing the ok button on the arm) the robot would then search for an object in the area defined by the same points and finally proceed to grip the object. When the object was successfully gripped, the robot would move the object, such that the side that needed to be measured ($x_{gripped}$), is facing the human and that the object is resting on the worktable, but still gripped firmly. The program would wait again for external input before placing the object. This was done to give time to measure $x_{gripped}$.

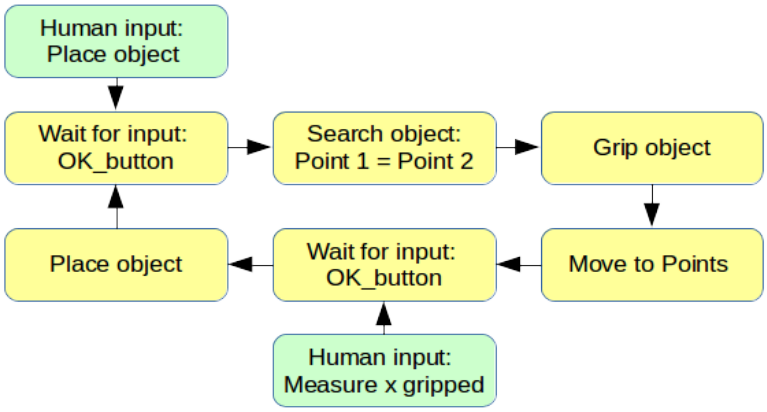



**Figure 42:** Flow chart of program for the experiment of the validation of the gripping location and orientation algorithms.

### 5.1.2 Analysis of the data

#### Object 1

In table 6 the measured data is shown for object 1. 10 attempts were made per orientation. The explanation of the data follows after the table.

| **Mean values for measurements of Object 1** | | | | | | | |
|---|---|---|---|---|---|---|---|
| | $\eta \pm \sigma$ | | | | | | |
| $\theta_{act}$ (°) | S (%) | $\Delta x$ (mm) | $\Delta\theta$(°) | $t_{DO}$ (s) | $t_{FO}$ (s) | $t_G$ (s) | $t_T$ (s) |
| **0°** | 100 | $-1.4 \pm 1.7$ | $6.5 \pm 10.3$ | $0.8 \pm 0.7$ | $0.019 \pm 0.005$ | $12 \pm 4$ | $20 \pm 6$ |
| **45°** | 90 | $1.7 \pm 2.4$ | $6.7 \pm 8.1$ | $0.5 \pm 0.5$ | $0.039 \pm 0.020$ | $17 \pm 3$ | $24 \pm 3$ |
| **90°** | 90 | $-0.2 \pm 3.1$ | $4.5 \pm 11.0$ | $1.1 \pm 0.7$ | $0.020 \pm 0.005$ | $22 \pm 4$ | $31 \pm 6$ |
| **135°** | 90 | $-0.5 \pm 2.1$ | $-13.4 \pm 10.8$ | $1.2 \pm 0.97$ | $0.024 \pm 0.003$ | $29 \pm 7$ | $38 \pm 10$ |
| **180°** | 30 | $-1.3 \pm 1.4$ | $-0.5 \pm 7.0$ | $0.6 \pm 0.4$ | $0.023 \pm 0.006$ | $35 \pm 13$ | $42 \pm 14$ |
| **225°** | 60 | $-3.5 \pm 1.7$ | $5.9 \pm 10.1$ | $1.0 \pm 0.6$ | $0.020 \pm 0.003$ | $28 \pm 9$ | $34 \pm 9$ |
| **270°** | 80 | $-0.3 \pm 1.3$ | $6.9 \pm 19.6$ | $0.6 \pm 0.5$ | $0.133 \pm 0.197$ | $23 \pm 4$ | $31 \pm 2$ |
| **315°** | 70 | $1.1 \pm 1.7$ | $12.0 \pm 10.8$ | $0.9 \pm 0.9$ | $0.022 \pm 0.004$ | $24 \pm 4$ | $32 \pm 6$ |
| **total** | 77 | $-0.7 \pm 2.2$ | $2.6 \pm 11.4$ | $0.8 \pm 0.6$ | $0.04 \pm 0.038$ | $24 \pm 8$ | $31 \pm 9$ |

**Table 6:** Measured data for object 1.

In total 77% of the attempts were successful. In other words, 19 out of the 80 attempts failed.

*Failures*
The failed attempt at $45°$ was due to the gripper that was not completely opened. In that same orientation, 2 other attempts were successful despite the fact that the grippers were not fully open. The experiment was then restarted because the occurrence of gripper failure was too high.

The failed attempt at $90°$ was due to a completely wrongly calculated orientation ($63°$), resulting in the gripper getting stuck in a type 3 failure as in figure 41c. In the same orientation another gripper failure occurred, but the attempt was successful even with not fully opened grippers.

The failure at $135°$ was also of type 3, 41c.

Of the 7 failures at orientation $180°$, 5 were of type 2, figure 41b. The 2 others were errors of type 3, figure 41c, which were the result of a bad orientation.

The 4 failures of the orientation at $225°$ were 3 of the type 3, figure 41c, and one of the type 1, figure 41a, also a result of bad orientation.

The 2 failures of orientation $270°$ were again a failure of type 3.

it they 2 of the failures of orientation $315^\circ$ were a failure of type 3. The other one was due to a type 5 failure of not fully opened grippers.

*Overall performance*
The result for $\Delta x$ is very good. Almost zero offset and a standard deviation that is lower than expected, given the error on the position of the inverse kinematic solver showed a standard deviation for the error around 4.6 mm (4.3 mm for x and 1.7 mm for y, table 3, thus 4.6 ($= \sqrt{4.3^2 + 1.7^2}$)). This is not surprising because a too large error on the position resulted in the majority of the failures: type 1, figure 41a, type 3, figure 41c, and type 4, figure 41d. This type of failures prevented the measuring of $\Delta x$. Consequently the experiment gives a slight misrepresentation of this error which explains why the deviation is better then expected.

When looking at $\Delta\theta$ one can observe that a standard deviation of $11.4^\circ$ is very large. When looking at the data attempt by attempt, it can be deducted that there is almost a 1 in 2 chance that the orientation is almost exact, within some degrees, or that it is immediately $20^\circ$ off. This comes from the way the R($\theta$) plots are compressed to only the maximum and minimum values, as explained in section 3.2.4. If you look closely at object 1 the maximum value for R is in the tail. There are 2 equal maximum values, that are the 2 corners of the tail, which are separated by exactly $20^\circ$. The part in between is more or less straight in the R($\theta$) plot and because of this only one of the 2 points is chosen as the maximum by the algorithm. The chosen maximum is the one that is the biggest for that R($\theta$) plot, but unfortunately since both normally have the same value this choice is purely subject to the noise of the measurement, resulting in a 50/50 chance of choosing one or the other.

This can be prevented by changing the algorithm to take both points at the end of a "straight" line in the R($\theta$) plots or maybe by using another representation.

Most of the attempts done with a wrong orientation still resulted in successes. This is because the angle variation was not that large. Though most of the failures of type 3 (and type 1) can be attributed to a combination of a wrongly calculated orientation and the inaccuracy of the inverse kinematic solver.

Looking at $t_{FO}$ and $t_{DO}$ one can conclude that both are very low and only represent a small fraction of the total time needed to pick up an object. Most of the time $t_G$ was spent waiting for the joints to reach enough precision to continue to the next movement. This proves that the algorithms are fast and work well, except for the error in compressing the data, but this can be solved easily and will not make the algorithm much slower.

It can be observed that finding the orientation $t_{FO}$ has a constant good performance, with a small standard deviation with respect to the mean value. But the detecting object time algorithm $t_{DO}$ is not as constant, with a standard deviation almost equal to the mean value. The time was very dependent on how fast after the last movement, back to the initial position, the ok button was pressed to start the next search. The variation could be a result of the movement. When the robot is moving the image is very blur and mostly

no contours can be detected on it, certainly not the ones that have to match to a certain contour. So it took roughly 0.02 seconds to detect the object when the camera, thus the arm, was stopped completely, while it took 2 seconds to detect an object when he was still adjusting his joints after a precision of $0.5^\circ$ was met. This movement has been discussed in section 3.3.3 and is shown in figure 28. $0.5^\circ$ was the precision asked for to be achieved within an acceptable time without losing too much accuracy, before moving on to the next task,

In general the time needed to complete the task increases with the orientation. The times given for the orientation of $180^\circ$ are high because a lot of the times the object was not found after the orientation of the camera and needed intervention of the human operator to bring the object, which increased the time measured because of the reaction time of the operator to interfere - operator needs to pay attention - and it takes longer to turn the camera over the orientation. But it is not the only factor that determines the difference in time. Sometimes it took more time to center the camera exactly over the object. This could go faster when a better controller would be made for the feedback of the position of the object in the camera image to the desired position of the arm.

This concludes the observation of the experiment with object 1.

### Object 2

To prevent the calculation of the wrong orientation on object 2, as has been seen and explained with object 1, the object was slightly changed to have only 1 distinct maximum. This is shown in figure 43.

**Figure 43:** Adaptation of object 2 to have one distinct maximum.

In table 7 the measured data is shown for the adapted object 2. Only 5 attempts were made per orientation for object 2. The explanation of the data follows after the table.

| Mean values for measurements of Object 2 | | | | | | | |
|---|---|---|---|---|---|---|---|
| | $\eta \pm \sigma$ | | | | | | |
| $\theta_{act}$ (°) | S (%) | $\Delta x$ (mm) | $\Delta\theta$(°) | $t_{DO}$ (s) | $t_{FO}$ (s) | $t_G$ (s) | $t_T$ (s) |
| **0°** | 100 | $4.7 \pm 0.1$ | $1.8 \pm 1.3$ | $0.7 \pm 0.5$ | $0.023 \pm 0.006$ | $20 \pm 6$ | $27 \pm 4$ |
| **45°** | 100 | $4.0 \pm 1.4$ | $3.1 \pm 0.8$ | $1.4 \pm 0.6$ | $0.030 \pm 0.005$ | $18 \pm 4$ | $25 \pm 5$ |
| **90°** | 100 | $4.0 \pm 6.4$ | $0.7 \pm 0.6$ | $1.1 \pm 0.7$ | $0.022 \pm 0.005$ | $17 \pm 2$ | $25 \pm 2$ |
| **135°** | 80 | $-3.0 \pm 4.0$ | $1.5 \pm 1.9$ | $1.5 \pm 1.0$ | $0.021 \pm 0.003$ | $22 \pm 3$ | $30 \pm 3$ |
| **180°** | 80 | $-1.3 \pm 3.4$ | $2.8 \pm 0.7$ | $1.5 \pm 0.7$ | $0.020 \pm 0.003$ | $27 \pm 6$ | $33 \pm 7$ |
| **225°** | 100 | $-4.6 \pm 5.1$ | $-1.1 \pm 1.6$ | $0.6 \pm 0.8$ | $0.031 \pm 0.008$ | $27 \pm 9$ | $35 \pm 8$ |
| **270°** | 100 | $-2.3 \pm 2.6$ | $1.6 \pm 1.1$ | $0.5 \pm 0.6$ | $0.022 \pm 0.004$ | $28 \pm 10$ | $35 \pm 10$ |
| **315°** | 100 | $2.0 \pm 1.8$ | $-1.6 \pm 1.0$ | $0.3 \pm 0.3$ | $0.023 \pm 0.005$ | $27 \pm 8$ | $32 \pm 10$ |
| **total** | 94 | $0.3 \pm 4.5$ | $1.5 \pm 1.4$ | $1.0 \pm 0.8$ | $0.024 \pm 0.006$ | $23 \pm 6$ | $30 \pm 6$ |

**Table 7:** Measured data for object 2.

In total 94% of the attempts were successful. In other words, only 2 out of the 40 attempts failed.

*Failures*
The failure at the orientation of $135°$ was a failure of type 3, figure 41c, and was probably caused by an extreme value or the inaccuracy of the inverse kinematic solver. With the same orientation a success, as shown in 40b, was also observed. This is again a result of an extreme value or the inaccuracy of the inverse kinematic solver.

The failure at the orientation of $180°$ is one of type 2, figure 41b. Thus the object was out of the field of view after the rotation.

2 successes as shown in figure 40b were observed during the attempts for the orientation of $225°$.

*Overall performance*
The result for $\Delta x$ is very good. Since this time almost all experiments were a success, all the data of the measured distance could be used. Hence the 4.6 mm in deviation is found as was to be expected. This also shows that the algorithm to learn a location is pretty good, since the error on the location is due to the intrinsic errors of the accuracy of the robot itself.

The results for $\Delta\theta$ have also drastically improved now since the ambiguity of the location of the maximum in the R($\theta$) plots have been bypassed with the modified object 2. This confirms the theory of the error found with object 1.

Times were not really affected by object 2 compared to object 1.

This concludes the observations of object 2.

### 5.1.3 Conclusion on the algorithms experiment

In general one can state that the algorithm to detect an object and to find the orientation of the object take very little time with respect to the total time needed to pick up the object.

It has been shown that the algorithm to compress the R($\theta$) plots needs some adaptation, since problems arise when 2 equal maximum values are connected by a more or less "straight" line in the R($\theta$) plots.

When the problems of the compression algorithm are bypassed, the results are much better. With an error in the gripping location mostly coming from the inaccuracy of the robot itself. The error in the angles are also very small, at least small enough to not interfere in the gripping location or the end result of the gripping action.

It has been demonstrated that when rotating the camera, the position of the arm should be changed to compensate for the offset of the camera with respect to the centre of rotation.

Also a better controller for the feedback of the position of the object in the camera image could be made to improve the overall speed.

## 5.2 Validating the usability

The experiment for validating the usability of the robot, as explained in section 2.4, using a wizard based control architecture to configure the robot, as explained in section 4, is meant to show that people with little or no experience in programming can program the robot to perform a simple task in an acceptable time. The experiment is in no way statistically significant, for which a much larger experiment needs to be set up. However it gives a first indication.

The parameters measured during this experiment are shown in table 8

| Measured parameter | Explanation |
|---|---|
| Age | Age of the participant. |
| Level of education | Level of education the participant is currently applying for, if none the highest level attained, eg: master in Law or bachelor in Engineering. |
| Programming skills | The skills in programming the participant estimates himself to have. Ranging from :<br>1. Novice: No programming skills at all; "Get those things away from me!"<br>2. Amateur: Having done some simple programming; "I can make a program to add numbers."<br>3. Engineering: What can be expected from an average engineering student; "I think I know what object oriented programming is... "<br>4. Computer science guru: Expert programmer; "Hell I could have written those 4200 lines of code in 1 line, noob!" |
| $t_{succes}$ | Time it takes to program the robot to perform a simple task until the first successful attempt of the robot to perform this task. |

**Table 8:** Overview of measured parameters in the validation of usability.

### 5.2.1 Experimental setup

Test persons were asked for their age, their educational level and their self estimated programming skills.

Because this was not a test to measure the intellect and problem solving capabilities of the operators, but merely to show that with a wizard based programming architecture somebody with no experience in programming could program the robot to do a simple task. Therefore the participants were told what the robot had to do conceptually. The ideal flowchart of the configuration of the robot is shown in figure 44.

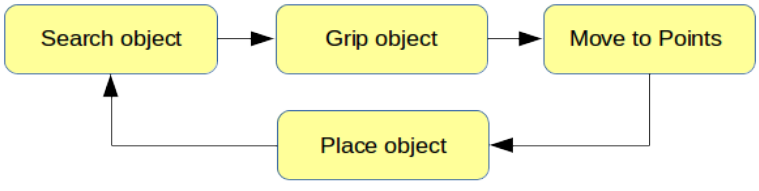



**Figure 44:** Ideal flowchart for the configuration of Baxter for the usability experiment.

The simple task they had to program was shown to them once with an explanation of the configuration of the menus for the subtasks. This task was to find 1 type of object in an area

and to place it in the right box with the right orientation. The test object used was object 1, which is shown in figure 12. In order not to encounter the problems of the compression algorithm the object was adapted to have a distinct maximum value as shown in figure 45b.

In figure 45 the experimental set-up is shown. The blue rectangle in figure 45a indicates the area where the object to be searched for, could be expected. In figure 45b the target location of the object is shown. It consists of an opening in a box, which is a little larger than the object itself, where the object has to be dropped off.

**(a)** Area to search the object in.

**(b)** Location to drop the object.

**Figure 45:** Experimental set-up for the validation of the usability.

Besides the conceptual explanation of the task, the participants were shown once how to operate with Baxter and the wizard,while shortly explaining all the available subtasks. This training took roughly 20 minutes per test person.

Once everything was explained the test person could start the programming. The time it took the person to program Baxter, from the first touch of the experimental set-up. Until the first successful attempt of Baxter to perform the task, was measured.

### 5.2.2 Analysis of the data

In table 9 the results are shown.

| **Data of experiment 2** | | | | |
|---|---|---|---|---|
| # | Age(years) | Educational level | Programming skills | $t_{succes}$ (min) |
| **1** | 24 | Bachelor in Psychology | 1 | 11.1 |
| **2** | 23 | Master in Art-History and Archaeology | 1 | 13.4 |
| **3** | 23 | Bachelor Marketing | 2 | 10.2 |
| **4** | 23 | Bachelor Kinesitherapy | 2 | 8.9 |
| **total** | - | - | - | $10.9 \pm 1.2$ |

**Table 9:** Measured data during the usability experiment.

It was accepted during the experiment that the object didn't pass completely through the opening, because of the known errors on the position.

*Test person 1*
During the experiment with test person 1 a bug was found. When the last action was to grip the object, and this needed to be deleted, the object was still gripped. Therefore the robot had to be restarted. The time measurement was not stopped though.
The test person noticed that on the second try it was easier to go through the configuration. This difference was not measured, but it could be a second step to measure the learning curve, once all the bugs would have been deleted.

*Test person 2*
During the experiment with test person 2 another bug was found. When moving through a search area where point 1 was defined with larger values than point 2 the search algorithm didn't move to the right point. Because of this, the robot was again restarted. The time measurement was not stopped. This bug was fixed for the next test person.
Test person 2 noticed that sometimes the tactile sensor didn't work as well, especially when it was pressed upon for a longer time.
The person also remarked that it was sometimes not very practical that some action had to be done with the buttons on the wrist and others with the navigator buttons on the arm.

*Test person 3*
During the experiment with test person 3 no bugs were detected.
Though test person 3 needed a bit more explanation during the experiment. Test person 3 was impressed at the end of the experiment by the time it had taken him to do it. With 20 minutes training plus 10 minutes programming, the person reached a successful manipulation.

*Test person 4*
During the test with person 4 another bug appeared. When the object was detected and the orientation was found, the last move to the right gripping location didn't happen, even after 3 attempts to grip the object. When pushing the block to another location, in the field of view of the camera, the movement to the right gripping location did happen. Why the first 3 attempts failed are not clear.
Despite the bug, test person 4 needed the least guidelines and even wanted to program a high 5 after the experiment. This was unfortunately not possible because of the design of the analytical inverse kinematic solver.

*General*
Though the result is not statistically relevant, it gives already an indication that people with no programming background, test persons 1 and 2, and people with limited programming background, persons 3 and 4, could program the Baxter robot, which they never used or saw before, to complete a simple task in about 30 minutes time, when taking into account the initiation training of 20 minutes.

During these tests some bugs were found. And it was also clear that the wizard had a limited flexibility when the operator made a mistake, resulting most of the time in starting all over again.

It was also noted that all 4 people enjoyed doing the experiment.

### 5.2.3 Conclusion on the usability experiment

From these tests we can conclude that the idea of high level programming with the use of a wizard to configure subtasks for people with no experience, can work very effectively. However much care has to be taken into the flexibility of the programming. Interpretation of commands differs from person to person and making a program that foresees all possible human interpretations is difficult if not impossible.

## 5.3 Conclusion of the experiments

The result of the first experiment, in section 5.1 showed that the algorithms to detect an object and to find the orientation of the object work very fast with respect to the overall time needed to move the robot to the right positions.

The first experiment also showed that the algorithm to compress the R($\theta$) plots needs to be adapted, since it failed when 2 equal extrema were connected with a "straight" line in the plots. When bypassing this problem, by giving the objects a distinct maximum, the chance of a successful gripping action improved drastically.

In the first experiment it was also noticed that the algorithm for the orientation needs some adaptation after the orientation has taken place. To take into account the movement of the camera, which has an offset with respect to the center of rotation. This caused the camera's field of view to move away from the object.

It was also noted that the algorithm to center above an object could benefit from a better control.

The experiment on the usability, section 5.2, gave already a good indication that people without a lot of training could easily learn a robot to do a simple task using a wizard based configuration architecture.

It was also noted that the wizard should be very flexible, and a lot of thought has to go in all possible mistakes that could happen from the interpretation of the guidelines by the operator.

# 6 Conclusion

Human robot collaboration embraces many subfields, going from AI and CV to inverse kinematics to even psychology.

During this project several algorithms have been developed and implemented to interact with objects. Most important are the detection and discrimination algorithms described in sections 3.1 and 3.2 and the algorithm to find the orientation of a known object (or class) in section 3.2.
These algorithms have been tested during the experiment described in section 5.1 and showed good results on speed. It was shown that the algorithm to compress the R($\theta$) plots had a major drawback. When the problem of compression was bypassed, the algorithms gave a 94 % chance of gripping an object in random orientation.

In section 3.3 the drawbacks of the iterative inverse kinematic solver implemented on Baxter have been discussed and a simple analytical inverse kinematic solver has been developed to overcome these problems. The limitations of Baxter concerning the precision have also been discussed in section 3.3. It has been demonstrated that the robot needs several seconds to reach the precision claimed, and even then the reachable accuracy measured in several millimetres.

In section 4 the architecture of a wizard based configuration tool has been developed to interact with humans. The most important features of this toolbox is to learn the robot how to grab an object (location, orientation and force). This feature is a direct product of the problem solving capabilities of the human and the repeatability (and accuracy) of the robot. This tool has been tested in experiment 5.1 and showed good results, with variations mostly caused by the intrinsic inaccuracy of the robot.

During this project, 6 subtasks were developed in section 4 for the toolbox, but for none of them feedback was provided during the execution procedure.

The usability of the toolbox was tested during experiment 5.2. Though statistical not relevant, it already indicated that the use of a wizard based configuration tool could be a success. The 4 test persons, with no programming background, whom never used Baxter before, got 20 minutes training and afterwards took an average of 10 minutes to implement a simple shape game using 1 shape. During those test the inflexibility of the toolbox in case of mistakes became apparent.

# 7 Further work

Further work can be spilt in several parts:

- **Orientation and CV:** As has been showed in 5.1 there are some changes that have to be made to improve the algorithms. The compression algorithm needs some adaptation as well as the moving algorithm for the orientation. The use of R($\theta$) plots worked well for the simple objects, but will have more difficulty when complexer objects are used. The implementation of S($\phi$) lots to find the orientation could be used. S($\phi$) plots can also cope with occlusion which is good when sorting objects from a bin, see section 3.2.

- **Accuracy:** It would be beneficial for the speed and accuracy of Baxter to overwrite the controls of joints $S_1$ and $E_1$ and make a new controller for these joints. Also a better controller could be built to improve the speed at which the camera is centred over the centre of area of the object.

- **Communication and interaction:** A lot can be done in this section. For now all communications have been mostly one way. From human to robot. The expansion of the execute task procedure to incorporate problem procedures is a major work that needs to be done as a next step.
  Besides this the incorporation of other sensors to observe the human collaborator more closely can be implemented. With it, a gesture based communication could be developed, going from and to the robot, as described in section 2.3.

- **Safety:** In this project only the compliance of the joints have been used as safety system. Since the joints of Baxter are not that compliant, pre-collision strategies should be implemented to guarantee the safety of the human collaborator as well as other objects like laptops, placed in the workspace of Baxter.

- **Wizard:** The wizard can be extended widely and has to be made more flexible for the errors of the human collaborator. To make the wizard error proof will take a lot of tests as described in section 5.2. The wizard should also be extended with the possibility to use logical decisions, for example: If object 1 is at position X, then find object 2 and place it at position Y, else find object 3 and place it at position X.

- **Discrimination and AI :** The tests on the detection and discrimination algorithms have only been tested with 2 objects. More tests should be done with a variety of objects. Also the naming game was not implemented because lack of time. It could be interesting to learn object through reinforced learning by the use of the naming game.

- **Dual arm**: Baxter has 2 arms, but only one has been used during this project. A next interesting step could be to put 2 Lego pieces, or something else, together using both arms as part of an assembly. Although only one arm was used in this project, most code has been made for both arms. The code was only tested on the right arm and it is foreseen that some debugging will be needed to make it work on the left arm.

Besides this a more generic way of building the GUI should be made, such that it is easily adaptable.

# 8 Bibliography

# References


[1] Jane Shi, Glenn Jimmerson, Tom Pearson, and Roland Menassa. Levels of human and robot collaboration for automotive manufacturing. In *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, PerMIS '12, pages 95–100, New York, NY, USA, 2012. ACM.

[2] G.F. Rossano, C. Martinez, M. Hedelind, S. Murphy, and T.A. Fuhlbrigge. Easy robot programming concepts: An industrial perspective. In *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pages 1119–1126, Aug 2013.

[3] Computing Community Consortium et al. A roadmap for us robotics: From internet to robotics. 2009.

[4] Duhamel et al. "rethink robotics - finding a market" stanford casepublisher 204-2013-1. 20 may 2013.

[5] Krishnanand N Kaipa, C Morato, Jiashun Liu, and Satyandra K Gupta. Human-robot collaboration for bin-picking tasks to support low-volume assemblies. In *Human-Robot Collaboration for Industrial Manufacturing Workshop, held at Robotics: Science and Systems Conference (RSS 2014)*, 2014.

[6] http://www.spectrum.ieee.org/robotics/industrial-robots/rethink-robotics-baxter-robot-factory worker. Ieee spectrum site.

[7] Scott A Green, Mark Billinghurst, Xiaoqi Chen, and GJ Chase. Human-robot collaboration: A literature review and augmented reality approach in design. University of Canterbury. Human Interface Technology Laboratory., 2008.

[8] Stefanos Nikolaidis, Przemyslaw Lasota, Gregory Rossano, Carlos Martinez, Thomas Fuhlbrigge, and Julie Shah. Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action. In *Robotics (ISR), 2013 44th International Symposium on*, pages 1–6. IEEE, 2013.

[9] Terrence Fong, Charles Thorpe, and Charles Baur. Collaboration, dialogue, human-robot interaction. In *Robotics Research*, pages 255–266. Springer, 2003.

[10] Carlos Morato, Krishnanand N Kaipa, Boxuan Zhao, and Satyandra K Gupta. Toward safe human robot collaboration by using multiple kinects based real-time human tracking. volume 14, page 011006. American Society of Mechanical Engineers, 2014.

[11] Dana Kulić and Elizabeth Croft. Pre-collision safety strategies for human-robot interaction. volume 22, pages 149–164. Springer, 2007.

[12] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[13] Robotiq collaborative robot ebook, fifth edition.

[14] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. volume 24, pages 15–26. IEEE, 2008.

[15] C Fitzgerald. Developing baxter. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6. IEEE, 2013.

[16] Thomas Lens, Jürgen Kunz, Oskar von Stryk, Christian Trommer, and Andreas Karguth. Biorob-arm: A quickly deployable and intrinsically safe, light-weight robot arm for service robotics applications. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–6. VDE, 2010.

[17] Hoang-Long Cao, Pablo Gómez Esteban, Albert De Beir, Ramona Simut, Greet Van de Perre, Dirk Lefeber, and Bram Vanderborght. Robee: A homeostatic-based social behavior controller for robots in human-robot interaction experiments.

[18] R v Ham, Thomas G Sugar, Bram Vanderborght, Kevin W Hollander, and Dirk Lefeber. Compliant actuator designs. volume 16, pages 81–94. IEEE, 2009.

[19] Matteo Laffranchi, Nikolaos G Tsagarakis, and Darwin G Caldwell. A variable physical damping actuator (vpda) for compliant robotic joints. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1668–1674. IEEE, 2010.

[20] Sami Haddadin, Alin Albu-Schaffer, and Gerd Hirzinger. The role of the robot mass and velocity in physical human-robot interaction-part i: Non-constrained blunt impacts. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1331–1338. IEEE, 2008.

[21] Tobias Ende, Sami Haddadin, Sven Parusel, T Wusthoff, Marc Hassenzahl, and A Albu-Schaffer. A human-centered approach to robot gesture based communication within collaborative working processes. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3367–3374. IEEE, 2011.

[22] Brian Gleeson, Karon MacLean, Amir Haddadi, Elizabeth Croft, and Javier Alcazar. Gestures for industry: intuitive human-robot communication from human observation. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 349–356. IEEE Press, 2013.

[23] http://www.ni.com/pdf/manuals/372668d.pdf.

[24] Rainer Bischoff, Arif Kazi, and Markus Seyfarth. The morpha style guide for icon-based programming. In *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pages 482–487. IEEE, 2002.

[25] Image from site: http://new.abb.com/products/robotics/robotstudio/tutorials.

[26] Baxter research robot specification and datasheet [hardware v1.0, sdk v1.0.0].

[27] Robot operating system, http://www.ros.org/.

[28] Api reference tree for baxter, http://api.rethinkrobotics.com/ baxter_interface/html/index.html.

[29] Sthitapragyan Parida, Juan Pablo Wachs, and Maria Eugenia Cabrera. Dynamic surgical tool tracking and delivery system using baxter robot. 2014.

[30] Matthew P DeDonato, Velin D Dimitrov, and Taskin Padir. Towards an automated checked baggage inspection system augmented with robots. In *SPIE Defense+ Security*, pages 90740N–90740N. International Society for Optics and Photonics, 2014.

[31] Image from site: http://new.abb.com/products/robotics/yumi.

[32] Luc Steels. Course : Artificial intellegence : Programing paradigms. Vrije Universiteit Brussel, 2014.

[33] Luc Devroye and Terry J Wagner. Nearest neighbor methods in discrimination. volume 2, pages 193–197, 1982.

[34] Image from site: http://www.aceultimatedisc.com/red_ultra_star.html.

[35] Bart De Vylder and Karl Tuyls. How to reach linguistic consensus: A proof of convergence for the naming game. volume 242, pages 818–831. Elsevier, 2006.

[36] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

[37] A Ashbrook and NA Thacker. Tutorial: Algorithms for 2-dimensional object recognition. 1998.

[38] Roberts Cross. Canny edge detector. *Algorithm Designs*, page 39.

[39] Opencv tutorials for python, http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_imgproc /py_table_of_contents_imgproc/py_table_of_contents_imgproc.html.

[40] Karen Villaverde and Vladik Kreinovich. A linear-time algorithm that locates local extrema of a function of one variable from interval measurement results. volume 4, pages 176–194, 1993.

[41] Michael Van Damme. Course: Robotics. Vrije Universiteit Brussel, 2014.

[42] Sdk of baxter, http://sdk.rethinkrobotics.com/.

[43] Openrave, http://openrave.org/.

[44] Image from site: http://www.edinburgh-robotics.org/.

[45] Mail with rethink robotics on 9 april 2015. 2015.

[46] Unified robot description format, http://wiki.ros.org/urdf.

[47] Tf, http://wiki.ros.org/tf.

[48] Image from site: http://www.nationalautismresources.com/wooden-shape-sorting-cube.html.

# 9 Annex

## 9.1 Proof of equations 3 and 4

For a population, $X_n$, of n individuals, $x_i$, the mean value, $\eta_n$, and the standard deviation, $\sigma_n$, are defined as :

$$\eta_n = \frac{1}{n}\sum_i^n x_i \tag{29}$$

$$\sigma_n^2 = \frac{1}{n-1}\sum_i^n (x_i - \eta_n)^2 \tag{30}$$

When an individual,$x_{n+1}$ is added to the population the the new mean value, $\eta_{n+1}$ is defined by :

$$\eta_{n+1} = \frac{1}{n+1}\sum_i^{n+1} x_i \tag{31}$$

$$\eta_{n+1} = \frac{1}{n+1}(\sum_i^n x_i + x_{n+1}) \tag{32}$$

By making use of equation 29 one gets:

$$\eta_{n+1} = \frac{1}{n+1}(n.\eta_n + x_{n+1}) \tag{33}$$

This proves the equation of 3.

The standard deviation, $\sigma_{n+1}$, of the new population is defined as :

$$\sigma_{n+1}^2 = \frac{1}{n}\sum_i^{n+1} (x_i - \eta_{n+1})^2 \tag{34}$$

$$\sigma_{n+1}^2 = \frac{1}{n}[(x_{n+1} - \eta_{n+1})^2 + \sum_i^n (x_i - \eta_{n+1})^2] \tag{35}$$

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2}{n} + \frac{1}{n}\sum_i^n (x_i - \eta_n + \eta_n - \eta_{n+1})^2] \tag{36}$$

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2}{n} + \frac{1}{n}\sum_i^n [(x_i - \eta_n)^2 + 2(x_i - \eta_n)(\eta_n - \eta_{n+1}) + (\eta_n - \eta_{n+1})^2] \tag{37}$$

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2}{n} + \frac{1}{n}(\sum_i^n (x_i - \eta_n)^2 + \sum_i^n 2(x_i - \eta_n)(\eta_n - \eta_{n+1}) + \sum_i^n (\eta_n - \eta_{n+1})^2) \tag{38}$$

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2}{n} + \frac{1}{n}(\sum_i^n (x_i - \eta_n)^2 + 2(\eta_n - \eta_{n+1})\sum_i^n (x_i - \eta_n) + n(\eta_n - \eta_{n+1})^2) \tag{39}$$

By using equation 3 one can find that $\sum_i^n (x_i - \eta_n) = 0$, inserting this in equation 39 one gets:

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2}{n} + \frac{1}{n}(\sum_i^n [(x_i - \eta_n)^2] + n(\eta_n - \eta_{n+1})^2) \tag{40}$$

Using equation 30 one gets :

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2}{n} + \frac{1}{n}((n-1)\sigma_n^2 + n(\eta_n - \eta_{n+1})^2) \tag{41}$$

$$\sigma_{n+1}^2 = \frac{(x_{n+1} - \eta_{n+1})^2 + (n-1)\sigma_n^2 + n(\eta_n - \eta_{n+1})^2}{n} \tag{42}$$

This proves equation 4.

## 9.2 Programs

In this section the programs used for the end program are described shortly and their most important classes and methods are explained. Don't mind the spelling mistakes in the names of the classes, methods and variables.

All the code will also be sent by mail with the report.

### 9.2.1 Baxter_Controll.py

Baxter_Controll.py was built as an interface to ROS and the already existing interface to ROS of the SDK.

It consists of several functions, which are mostly needed for the transformation of co-ordinates and one class : the Baxter_robot class.

The Baxter_robot class has several important attributes listed bellow :

- self.cat: The classifier class, which kept track of all the classes of objects learned. This class will be more explained in objects.py .
- self.cam_con["left" \"right"]: Is the camera controller class, this class provided the calculations to transform the image of the camera to movements. It will be explained more in camera_control_movement.py.
- several attributes were used to keep track of the general state of Baxter, for example the angles of the joints or the status of the buttons.

The most important methods of the Baxrer_robot class are:

- self.IK_to_hori_pos: This function calculated the angles of the joints using the analytical inverse kinematic solver described in section 3.3.
- self.move_slowely: Splits a trajectory in a number of steps such that a good control was kept over the position of the end effector while moving. It is mostly used when moving down to grip an object and when moving up once an object had been placed.

- self.gripper_release_slowly : Is a function which makes the grippers open gradually. It was built after the observation that when the grippers moved abruptly when releasing an object that deviated from the original orientation.

### 9.2.2 camera_control_movement.py

The camera_control_movement.py exists of several functions to transform images and 1 class : camera_controller.

The most important attribute the camera_control_has is self.baxter , which is a Baxter_robot class as described in Baxter_Controll.py .

The most important methods of the camera_controller are :

- self.find_contours : This function finds the contours in the image and already selects if a contour is a possible object or not.
- self.find_closset_object: Determines which of the objects found in the contours lies the closest to the projection of the centre of the camera.
- self.close_in_object: Algorithm to close in an object, such that the projection of the centre of the camera is close to the centre of area of the object.
- self.move_to_point: PI controller that calculates the desired position of the end effector based on the analysis of the image taken by the camera.
- self.search_object_in_area: Algorithm that search through a rectangular area defined by 2 points in search of a specific object.
- self.find_orientation_of_object: Algorithm that finds the orientation of the object.
- self.grip_object: Algorithm that grips the object.

### 9.2.3 Learn_new_object3.py

Learn_new_object3.py consists of 2 important functions.

The first is the one that allows the operator to set the gripping force and location for the object.

In the second function the algorithm to learn the object is written. In this function the object is converted into a class.

### 9.2.4 menu.py

In menu.py the class menu keeps track of the status of the different menus.

### 9.2.5 objects.py

Objects.py includes 3 classes: concept, object_val and categorizer, and 1 important function.

The important function is find_local_extremums. In this function the algorithm is written to compress the R($\theta$) .

The class concept represents a class as described in section 3.1. It has 4 important attributes :

- self.angle: The angle the class has, determined by the orientation the object had when Baxter was learning the object.
- self.gripping_centre: The location where the objects of this class have to be gripped.
- self.gripping_force : The force the objects of this class have to be gripped with.
- self.contour: The compressed R($\theta$) plot of the objects of this class.

The concept class has 1 important method, the self.reinforce_object function. In this function the algorithms to reinforce an object to this class are programmed.

The object_val class has one important method, the self.find_best_fit. This function searches for the orientation of the object with respect to the orientation of the associated class.

The categorizer class in fact represents the classifier as described in section 3.1. It has one important attribute, self.list_of_concepts. This is a list where all the known classes are kept.

The categorizer class has 1 important method, the determine_concept_for_object. This function classifies the objects to the known classes.

### 9.2.6 task.py

In task.py 2 classes are written, the Task_element class and the Task class. The task element class has 1 important attribute, self.goal, which describes the goal of the sub task and 2 important methods:

- self.define_goal: This function is to configure the subtask, by filling in the self.goal attribute.
- self.obtain_goal: This function executes the configured subtask by use of the information in the self.goal attribute.

The class Task_element has 6 subclasses, the 6 sub tasks defined in section 4.4.1 :

- wait_time
- wait_for_input

- Move_to_position
- find_object
- grip_object
- place_object

The class Task has one important attribute, self.task, which keeps track of all the configured subtasks. It has 2 important methods :

- self.Make_task: which is the wizard to scroll through all the subtasks and to delete items and so on.
- execute_task : The function that executes the subtask in self.task by running all the Task_elements their self.obtain_goal method.

### 9.2.7 Thesis.py

In Thesis.py everything comes together. It forms the glue between the 3 procedures and the robot.

### 9.2.8 JointCommander.py

JointCommander.py has been written as a ROS interface to move the joints, after observations of failures of the built in interface to ROS.

## 9.3 Mail with Rethink Robotics

**Yuri Durodié <yuri.durodie@gmail.com>**

# Positions of s0 in cartesian coordinates

**Yuri Durodié** <yuri.durodie@gmail.com> 9 April 2015 at 18:01
To: brr-users@rethinkrobotics.com

Hi Ian,

Thanks for the fast reply. I didn't know the tool tf yet. Seems like a pretty strong tool.

Many thanks for the help,

Yuri Durodié,
2nd Master engineering sciences : Mechatronics-Construction at Bruface VUB/ULB
Head organizer engineering job fair at VUB

On 9 April 2015 at 17:38, Ian McMahon [rethink] <imcmahon@rethinkrobotics.com> wrote:

> Hi Yuri,
>
> You can use the standard ROS tools to look up the translation and rotation between any two frames on Baxter. Using the URDF to find the appropriate parent linkage for the right_s0 joint shows that it is the right_arm_mount: https://github.com/RethinkRobotics/baxter_common/blob/master/baxter_description/urdf/baxter.urdf#L587-L594
>
> ```
> $ rosrun tf tf_echo base right_arm_mount
> At time 1428593041.259
> - Translation: [0.025, -0.219, 0.109]
> - Rotation: in Quaternion [-0.003, -0.002, -0.383, 0.924]
>             in RPY [-0.003, -0.006, -0.786]
> ```
>
> Since there are only fixed joints in between the /base frame and the /right_arm_mount frame, this transform will be static. However, this transform will be slightly different on your robot due to the hand-welded arm mounts on ever Baxter torso. These mounts are factory-measured and included in your robot's param_server URDF (and therefore its transforms).
>
> Hope this helps,
> ~ Ian
>
> On Thursday, April 9, 2015 at 10:38:33 AM UTC-4, roboticsvub wrote:
>
> > Dear all,
> >
> > I'm trying to write a simpler IK to move in a horizontal plane. (The IK provided takes to much time to use in a controller) . But for this I should know the position of the joint s0 wrt to the Cartesian coordinates baxter provides in the /robot/limb/___/endpoint_state topic. I found by doing some simple test that the offset in the y direction is about 26 cm, x = 8cm and z = 30 cm. Also the angle at wich s0 is pointing straight forward (With all other angles set to 0) is pi/4 (or 45 deg) . But using that in my calculation doesn't seem to give accurate results.
> >
> > I search trough several specs files, but couldn't find the answer there.
> >
> > Does someone know the precise offsets wrt to the Cartesian coordinates?
> >
> > Thanks and kind regards,

> Yuri Durodié,
> 2nd Master engineering sciences : Mechatronics-Construction at Bruface VUB/ULB
> Head organizer engineering job fair at VUB

--

## 9.4 Wizard screens

All the screens will be sent by mail with the code and the report.

## 9.5 Experiment on algorithms

The spreadsheet with all the results of the first experiment will also be sent by mail.

## 9.6 Experiment on usability

Some movies will be sent by mail.