Tidy Duty
The tale of a bottle grabber

Jérôme AMIGUET, Riccardo FORMENTI and Yannick POFFET
Professor: Auke J IJSPEER
 Assistants: Alessandro CRESPI and Massimo VESPIGNANI

6th June 2014
Abstract

This report aims to expose the design and creation of a mobile and autonomous robot able to catch bottles amongst obstacles to bring them to a recycling area. The design and the different subsystems will be presented along with the reason of the choices that led to them. The robot built up from this design reached the second place ex-aequo during a robotic competition, collecting eleven bottles.
Contents

1 Introduction 6

2 Robot presentation 7
   2.1 Global 7
      2.1.1 Part 1: The clamps 7
      2.1.2 Part 2: The storage area 7
      2.1.3 Part 3: Rear 8
      2.1.4 Part 4: Front 8
      2.1.5 Part 5: Top 8
      2.1.6 Part 6: Hidden part 8

3 Methodology 9
   3.1 General approach 9
      3.1.1 Project progression 9
      3.1.2 Global design 9
   3.2 Mechanics 11
      3.2.1 Different mechanical subsystems 11
      3.2.2 Solution for the subsystems 11
   3.3 Electronics and Sensors 19
      3.3.1 Motors 19
      3.3.2 Sensors 23
      3.3.3 Boards 28
   3.4 Informatics 28
      3.4.1 General behaviour 28
      3.4.2 Bottles, bricks and walls detection with camera 29
      3.4.3 Odometry 33

4 Results 35

5 Discussion 36
   5.1 Discussion of the performances 36
   5.2 Possible improvements 36

6 Conclusion 37

7 Acknowledgement 38
1 Introduction

To design a robot is the dream of many child, and not for no reason: it is a fascinating job indeed. But it is also a complicate one necessitating a global overview of many subsystems: for the robot to work well, mechanics, electronics and informatics have to fit together in harmony. This supposes the designer to behave in accordance with the systematic approach characteristic of the engineers. In this paper will be exposed the method used to design a mobile and autonomous robot able to catch bottles amongst obstacles to bring them to a recycling area.

The global design will be presented first for the reader to have a general idea of the robot. Then will be presented the different subsystems in three different sections: mechanics, electronics and informatics. The choices that led to the final design of those subsystems will be discussed, along with how those choices were made. Then the average performance of the robot will be evaluated from its behaviour during tests and specifically during a robotic competition with other groups with their proper robot design. Finally, our approach will be commented regarding those performances.
2 Robot presentation

This section is useful to present the global structure of the robot. There are seven interesting zones on the robot. The following figures show these parts.

2.1 Global

Here is a figure showing the global structure of the robot. Basically, the robot has a camera and eight ultrasonic sensors to detect the ambient. To interact with it, the robot has actuated clamps that will grab the bottles once the robot detects one. The clamps will release the bottle into a recipient in order to stock eight to ten bottles before dropping them into the recycling area opening the back door.

Figure 1: Photo of the robot build in this project. The number 1 indicates the clamp, the number 2 shows the storage area, the number 3 points toward the rear part of the robot and the number 4 is the front part. The number 5 stands for the top of the robot and finally the number 6 represents the hidden part of the robot, where the electrical boards are placed.

2.1.1 Part 1: The clamps

The goal of this part is not only to grab the bottles, it is useful to detect if there are obstacles or bottles around the robot. In fact there are five ultrasonic sensors.

2.1.2 Part 2: The storage area

Thanks to this area, the robot can store up to ten bottles. It is important to store lots of bottles in order to spend less time possible doing travels to realise the bottles into the recycling area and the zone where the robots searches the bottles.
2.1.3 Part 3: Rear
The rear part of the robot is composed by three interesting systems. A door, an ultrasonic sensor and a counterweight. The door allows the robot to release the bottles stored into the storage area. The goal of the ultrasonic sensor is to prevent any collision while the robot goes backward and the counterweight allows the clamps to stay at a given height without using the servos too much, in fact, the clamps are not in contact with the ground when the robot is moving. The counterbalance is done thanks to the battery and a socket attached to strings that connect the battery to the clamps.

2.1.4 Part 4: Front
In the front of the robot stand two ultrasonic sensors needed to detect obstacles and a hole where the arm that holds the clamp can go through.

2.1.5 Part 5: Top
In the top part, one can find the camera Xtion, which is a 3D camera that works like the Kinect. This camera is useful for object detection; bottles as well as obstacles and walls.

2.1.6 Part 6: Hidden part
In the hidden zone, there are all the electronic boards and the servo for the clamp rising.
3 METHODOLOGY

3.1 General approach

3.1.1 Project progression

To design a robot is a complicated problem as it supposes to have a global overview of many subsystems. Each subsystem can work perfectly, they will be perfectly inefficient if they are not thought and designed to work with the others. These interfaces are often the springs of the miscellaneous worriment that paves the ways of engineering; from them gush forth delays and complications that are difficult to anticipate.

To avoid these complications as much as possible, a specific effort has been put on communication inside of the team and on the interfaces between different parts, and the robot has been carefully thought with a methodical seek for efficient designs in three distinguished steps: the first one was the planning and the research of solutions, the second one was the prototyping and the testing phase that leads to the third part, which is the final selection of the solution and the improvement of the robot.

For the the design part of the project, the whole team worked together to create the planning, to write down the scope statement and to sketch the firsts ideas of solutions. Then the work has been shared: one person has developed the mechanical part, another one has created a library for the camera and the last one has created the management of the sensors and motors and the obstacle avoidance based on them. Although all team members were specialised, the important decisions and changes were debated as a group.

Once the different subsystems had reached a satisfying level of confidence, all the team members united their efforts again for the final phase: the built up and the debugging. The debugging part was mainly done in the arena with trials, parameters tuning and adaptation of the code, but also of the robot to some extend: position of the sensors, tension in the cables, adaptation of the wheels with tape to avoid friction and so on.

3.1.2 Global design

3.1.2.1 To aim the ultimate robot

If it would exist, the ideal robot would be able to grab every bottles in every four areas (with obstacles, grass, beyond the rocks and on the top of the hill) and to release them into the recycling area. As it was difficult to estimate what could realistically be achieved during the project duration, as well as the time cost of the implementation of many functions, the approach taken was to aim this ultimate robot and to proceed then to the removal of some functions or features as the project would going on and thus as better estimations would become possible. The robot must then be able to move above rocks and on the slope, to store as many bottles as possible, to localise itself and of course to avoid obstacles, in order to not restrict the possibilities too early in the project.

3.1.2.2 To assure the plain robot

To aim the ultimate robot is one thing, but another more important is to now what has to be done necessary in order to assure it will be implemented at the end of the day. So after having been taken at one end, the problem has been taken at the other one: what is necessary to achieve the minimal requirement, and how can it be assured even if anything goes wrong? The minimal requirement have been taken as those necessary to assure that the public will not get bored at the final presentation: the robot has to move and to grab some bottles. The strategy to take as
many bottles as possible has been kept thus: if there is enough time, the localisation and the way back home will be implemented; if not, at least the public will have something to see.

During all the project, this worry to have a safe solution has been kept, and most design that were chosen, a ways reuse it differently to circumvent problems have been thought, and this until the details: in order to simplify the global structure of the robot and to help the eventual repairs that may need to be done on it, the idea was to use the same components as much as it is possible. For example, the same servo motors have been used every time servos were needed, even if they were an overpowered solution in some case, and most of the screws on the robot were either M4 or M2.

### 3.1.2.3 A useful tool for making choices

In order to systematise the way of making choices, criteria grids were used. An example of criteria grid is shown in figure 2. The principle is to give weights to one criteria relatively to another; for instance, is the power consumption more important than the simplicity? Of how much? The importance of each criteria is computed from those weights, and each criteria gets a specific importance from 0 to 1. Then, each solution to be evaluated is attributed a mark from 1 to 5 for each criteria, and a global mark is computed. In the examples, three solutions were evaluated for the bottle collection: the use of a small lift to store them in a compartment, the use of a 4-bars systems to lift them, or the use of a net to pull them on the ground. The 4-bars system receive the best grade and has thus be kept; it was then adapted and reevaluated iteratively to become the system built up: a lever with belt transmission.

This evaluation method has been used for most important subsystems -the locomotion, the collect of bottles, the sensors used, and so on. It is useful because it forces the user to think about what is truly important in a system and why. Other grids have not been displayed here because they present little interest for the reader; the reason of choices have been written and summarised instead.

![Figure 2: The criteria grid method: above, the weights, below, the evaluations](image-url)
3.2 Mechanics

3.2.1 Different mechanical subsystems

For this robot design, there was 3 main mechanical subsystems. The first one is the mobility subsystem, the one that will make the robot move. The second one is the grabber and loader system. It is the one that is charged to transfer the bottle from somewhere outside the robot to inside the robot.

3.2.2 Solution for the subsystems

3.2.2.1 Mobility subsystem

How the robot moves is a pretty important question. It needs to be able to go accros rocks, grass and also on a sloop. The idea was to use wheels or treads to move. Threads are very good to go on every surface and terrain without getting stuck. The problem is that they might be slipping more than wheel when the robot is turning, and slipping means lost of position acquired with the odometry. As we really needed the odometry to move arround, this was a good argument to choose wheels over thread. Moreover a lot of the material needed for wheels mobility subsystem was already in the stock of what could be bought with the virtual budget.

The Wild Thumper structure was already in the stocks and was already meeting a couple of the requirements. It was able to go accros nearly every obstacle, it was large enough to support the base size that was designed, so it looked like the best and simplest solution. However it was not the perfect base, it has a couple particularities that was a trouble so it had to be modified a little bit. The suspensions are very weak, that means that they would not be able to support the robot’s weight without bending entirely and that would mean rub against the
robot structure. So the solution was to block them to prevent them to go to high with a specially
designed 3d printed part. The result can be seen on fig: 4

![3d printed part made to block the suspension of going to high.](image)

Figure 4: 3d printed part made to block the suspension of going to high.

An other particularity of this platform is that the central wheels are lower than the four other
ones as we can see on fig: 3. As the smartest place to put encoders is on the central wheels rightly
because those are the ones that slips the least, one decided to use them for odometry only. So
there is no motors on those one but just a encoder per wheel. As the four external wheels are
supporting all the weight, the central wheels did not had to be blocked as well. The fact that
they are lower than the external one helps actually. It helps to keep a constant contact between
the wheel and the floor since the suspension is constantly trying to push the wheel lower than
ground.

As the encoder cannot support the forces coming from the wheel, a special had to be designed
for this precise purpose. It was 3d printed and forced on a steel rod to have a good rigidity. The
external diameter of this system was chosen to be the same as the motors so it could be mounted
the same way on the chassis as one can see on fig: 5.
3 METHODOLOGY

3.2.2.2 Bottle catching subsystem

To catch the bottle a lot of different solution were possible. It was required that the system lifts the bottles high enough to go above the wild thumper and pile a couple bottles. The catching clamps system was chosen above the lift mostly due to a higher working speed that allows to take a bottle in far quicker. A prototype of the clamps was made to see if it actually catches bottles and to analyse what could go wrong, if the a bottle could get stuck in a strange position (fig: 6). As it was working in nearly every situation, it was decided to go further with this solution.

The closing and opening of the clamps was done using only one servo. Two strings are used to open up the clamps and two other to close them. The servo, when rotating, is pulling two strings and releasing the to other ones. This is how the clamps can be opened by rotating the servo in one direction or an other (see fig: 7). A spring is also used to take care of all the difference in strings length during the motion and save the subsystem in case of colision.
Figure 6: Prototype of the catching system.
For the lifting system, rotating arms are bringing the clamps above the robot container to allow the bottle to fall in. To actuate this system one could have put the motors directly on the rotating shaft and supporting all the weight of the arms as did a other team. But to insure rigidity and unload the motors, a solution using timing belt transmission was chosen. It also allows to have a different gear ratio to obtain more trust reported to the load. A gear ratio of $1 : 2$ was selected. One can see a picture of the final transmission on fig: 8.
But this was not the first design for the transmission. First one started with a smaller belt with much smaller teeth, but it was slipping on the pulley and could not support the weight of the arms. So the belt and pulleys were changed to a bigger size. As the pulleys were 3d printed, the CADs coming from the manufacturer’s website were used and here arose a problem. They were false! The teeth of the belt were not entering the pulleys correctly. To find the right teeth size and pulley diameter, some prototype of each one had to be printed as one can see on fig: 9.

**Figure 8:** Arms lifting system transmission using belt and pulleys.
Even the bigger belt was slipping when the system tried to lift the arms, so the tension in the belt had to be increased a lot using a pusher (see fig: 8).

To have less forces acting against the lifting motors all the time, a counter weight system was designed. The battery holder that is placed on the rear of the robot is used as a counter weight and the force is transmitted to the lifting system through strings attaching on arms as one can see on fig: 10.

Figure 9: This is examples of the tries mad to find a pulley working with the belt.
3 METHODOLOGY

Figure 10: Here we see the counter-weight and the strings transmitting the forces to the lifting system.

3.2.2.3 Bottle releasing system

To release the bottles the simplest system is used. A back door that open to let the bottles slide down the slope and fall in the recycling area. As one can see on fig: 11, the door is actuated using a servo and a transmission four bar system.
3.3 Electronics and Sensors

3.3.1 Motors

Two different kind of motors are used on this robot. The servo motors and brushed DC motors. The following chapters will presents the places where these motors are and how they have been chosen.

3.3.1.1 Servos

There are three places where servo motors are used on this robot. The closing system of the clamp, the rising system of the clamp and the opening of the back door. In this chapter, the computation for the servos are presented.

Back door Here is a schematic view of the forces applied on the backdoor.
Figure 12: Schematic view of the storage system. The orange bar represents the back door, the force $F_1$ is the force due to the weight of the bottles, the force $F_{1x}$ is the projection of the force $F_1$ on the slope of the storage area and $F_2$ is the force that the back door has to apply in order to counter the force $F_1$.

The force $F_2$ is equal to:

$$F_{1x} = F_1 \cos(90 - \alpha)$$

$$F_2 = F_{1x} \cos(\alpha)$$
Once one estimates a maximal load of twelve bottles and a mass of 20 grams per bottle, which gives a total of 240 grams, the following numerical analysis can be computed:

\[ M = 0.240 \cdot 9.81 \cdot 0.075 \cdot \cos(90 - 30) \cdot \cos(30) \approx 0.0765 \text{[Nm]} \]

The servo motor selected is the Dynamixel AX-12A. The value of torque needed computed just before is much smaller than the actual stall torque of the servo motor used. But the fact that the this motor was on the virtual budget was a big advantage, therefore this servo has been chosen.

**Clamps closing**  For the closure of the clamps, it wasn’t needed to have a strong force. Moreover, mechanical elastic systems have been added on this subsystem to ensure it not to break if the clamps get stuck somewhere.

**Clamps rising**  For this part, the major part of the weight of the clamps was compensate with the counterweight. In fact if the servos are not powered, the clamp remains at the height one rises it. The biggest constraint here was the maximal acceleration due to the important inertia of the subsystem. The reason that two servos are used to actuate the rise of the clamp is that it is possible to have a larger acceleration.
3 METHODOLOGY

3.3.1.2 The wheels

The selection of the motors has been directed by four main properties, the total weight of the robot, the friction between the carpet and the wheels, the speed the robot needs to reach and the delivery time.

One can understand easily that a heavier robot needs more powerful motors to move. This robot weights aro. The friction of the wheels to the ground is a pretty good thing for the odometry, as the wheels won’t slip when turning, but the robot has more difficulties to turn. In fact, as it has six wheels, if the robot turns on itself, the center of the rotation will be between the two central wheels and these wheels will be the only one that will not slip. The four other wheels will have to slip on their side because of their position and the center of the rotation. The speed of the robot will directly be influenced by the selection of the motors. The robot has to be slow enough to ensure the sensors to take good measurements and to have enough time to process the data collected but fast enough to maximize the area covered. Having a fast delivery was very important as a motor could break if forced, if this happens they could be changed quickly.

Three different motors have been tried until the good one has been found. These changes where due to the fact that the robot has an elastic joint at the clamps, therefore the clamps starts oscillating too much if the robot moves too fast. In fact the difference that was important between these motors is the minimal speed they reach.

The steps to reach the solution to have four actuated wheels and two passives wasn’t the initial one. At first the robot should have had six actuated wheels in order to turn more easily. The motors used during this idea existed in a version where an encoder is plug directly on the motor. The last motor chose don’t exist with encoder plugged on them, therefor the solution selected has been to use four actuated wheels, two rear and two front wheels. The two central wheels are passive and an encoder is plugged on it to allow odometry. More information about odometry and encoders are presented in the related section.

A problem that has been encountered was that the wheels broke while the robot was turning. This is due to the friction on the carpet that bent the tires. The following figure shows what happens.
3 METHODOLOGY

Figure 14: This figure shows how the wheels are bent when the robot turns. This is due to the high friction between the wheel and the carpet. This can damage the motors and actually breaks the gear box.

A solution to this problem has been to stick tape on the wheels in order to reduce its friction on the carpet.

3.3.2 Sensors

Three major functions should be achievable from the data provided by the sensors:

- To detect items that is to said to detect anything that is not the floor. This include mainly walls, bricks and bottles, but also the rocks, in order to not break the system.
- To classify the items in the four categories above.
- To measure the distance to the item in order to choose what to do: a brick 20cm far should be avoided, but not one 2m far.

3.3.2.1 The Xtion depth camera

The Xtion is a 3D camera that works on the same principles that the more famous Kinect. The reason this camera has been chosen for this project was that it allows the users to explore lots of possibilities for the object detection and localization, which is very interesting and motivating. From the beginning, the idea of obstacles, walls and bottle detection was that it has to be contactless. Moreover, the simply detection of an object wasn’t enough, as the robot is quite big and has difficulties ensure maneuvers without hitting obstacles, the knowledge of the distance between the robot and these obstacles was necessary. A 3D camera can provide such information. But
another question could be why choosing the Xtion instead of the Kinect? The reason is simple, the Kinect is way heavier than the Xtion. Moreover, the Xtion needs only an USB connection, and the Kinect needs an external 12V power supply. Even if the Xtion is more expensive, it is more efficient and convenient.

In the code, the Xtion detects the distance of an object in front of the robot and if this distance is shorter than a given threshold, the robot will start some kind of obstacle avoidance. The general idea for the obstacle avoidance when the camera asks for it is that the robot will go backward, turn on itself and then go straight forward again. Of curse, the Xtion has to ensure obstacle avoidance but letting the robot collects the bottles, therefore only objects higher than a given value will be considered as obstacles.

3.3.2.2 The use of ultrasonic sensors

Why another sensor than the Xtion? The use of sensors has been planned since the beginning. First, the processing of the camera data is slow, and the robot moves pretty fast. Then, the camera only sees in front of the robot, and not closer than approximately forty centimeters. For these two reasons, other sensors had to be used for obstacle avoidance. But there is also another good reason to rely also on another sensor than the Xtion: robustness. The use of a board and a camera add two layers of hardware and two interfaces (Arduino Mega-Cubietruck and Cubietruck-Xtion) which can be considered as complex beside the use of a plain ultrasonic or infrared sensor. This is thus more risky: the algorithm can interpret wrongly the data, the communication can be broken, to double the amount of boards double the chances that the software meet an unexpected error and crash, etc. It is thus careful to have an obstacle avoidance that does not rely on the Xtion, but to use it for localisation and path planning.

The position of sensors  Obviously, to mistake a brick for a bottle can lead to disastrous issues, when to mistake a bottle for a brick is safe for the robot; the system has thus to be able to classify bricks with no false negative, but can have false positive - though not every time as its purpose is to grab bottles: it can’t always miss them. But it is not helpful to distinguish bricks from walls for obstacles avoidance.

There are eight ultrasonic sensors on the robot. Five on the clamps, two in front of the robot and one behind it as shown on the following figures.
Figure 15: Position of the ultrasonic sensors on the clamp. The sensors are circled by light-green circles.
Figure 16: Position of the ultrasonic sensors on the front part of the robot. The sensors are circled by light-green circles.
Figure 17: Position of the ultrasonic sensor on back of the robot. The sensor is circled by light-green circles.

**Ultrasonic sensors versus infrared sensors** Two kind of distance sensors have been considered for obstacle avoidance: infrared sensors and ultrasonic sensors. The infrared sensor is easier to use, but is influenced by the ambient light and the albedo of the reflecting object, which the ultrasonic sensor is not. In addition, the ultrasonic sensor can measure longer distances, until several meters if the object is well-oriented, when the infrared sensor can not detect something beyond 80cm and is way less precise: at 60cm, the infrared sensor is approximately 5cm precise, when the ultrasonic sensor is precise within 1cm. Moreover, the infrared sensor does not always detect bottles, as they are partially invisible to it, and receive often inconsistent data in return as the light is partially reflected. The only serious downside of the ultrasonic sensor is that it meets a problem to detect objects presenting a face oriented by more than 45° with the segment linking it to the sensor; beyond 45°, the sonic wave can be not reflected toward the sensor, but this problem was circumvent by the use of sensors at the tip of the clamps: those can be oriented at approximately 45° while the robot is moving, and the sensors on it can thus detect the obstacles that the sensors on the robot can not see. Of course, the Xtion can also be used for bricks detection, but as discussed before, the obstacle avoidance should be able to act in complete autarky.

3.3.2.3 The use of encoders

**Why use encoders?** The encoders are really useful for the localization. In fact it is possible, reading the number of turns that a wheel does allows a distance measurement. For this robot, two encoders are used and connected to the central passive wheels. Thanks to this setup, not only the position of the robot can be known, but also its orientation. These encoders are useful
for the localization and allow the use of odometry. Even if there is error on the odometry, this measurement can be improved by software and robot’s displacement strategy. This method will be explained in the related section.

3.3.3 Boards

As the robot uses many different systems and sensors, its global structure has been divided into three different parts: The motors of the wheels, the sensors and grabbing system, and finally, the camera. To each of those parts is dedicated a board to manage it; thus three different boards have been used for this project. These separations has been decided according to the mechanical subsystems that they represent but also because the system they have to control are very different.

3.3.3.1 Wild Thumper’s motor control board

Commands are sent to the wheels with the motor control board that is provided with the Wild Thumper.

3.3.3.2 Cubietruck

Why the Cubietruck? A Cubietruck is used to process the data acquired by the Xtion. The Cubietruck has been chosen because of its processing power and memory (2GB DDR3@480MHz), and because Linux could easily be install on it, making way easier the use of the Xtion. The Raspberry Pi was discarded because not powerful enough.

Systems The Cubietruck is running a light version of Ubuntu, Linaro Ubuntu (also called LUbuntu), chosen because it was light and well-supported by the Cubietruck. It has been installed on a microSD card to have more space (15GB instead of 8GB) and to be able to access easily to the files from an external computer. The wa

3.3.3.3 Arduino Mega 2560

The sensors are controlled by an Arduino Mega 2560. In order to simplify the sensors connections, a shield provided and created by Alessandro Crespi at the BioRob/EPFL for this competition. The Arduino Mega 2560 is not only used to read sensors, it is the brain of the robot. It will ask data to the Cubietruck and send orders to the servos and to the motor control board. The communication between Cubietruck and the Arduino is serial and the one between the motor control board and the Arduino is I2C.

3.4 Informatics

3.4.1 General behaviour

The Arduino Mega runs a state machine that check periodically the ultrasonic sensors and ask the Cubietruck about the bricks the Xtion can detect. Depending on these data, the state can be changed and a command is sent to the motors. The state machine update is called by a scheduler in order to be able to effectuate tasks between updates, such as receiving data from the Xtion or compute the position from the odometry. The general behaviour goes like this: if the robot detects something with the sensors at the top of the clamps, it turns around to detect what it is. When it faces it, if it can detect them with the front sensors, it is a brick, as it is high enough; else, it is a bottle, and so it go forth and take it. It happens sometimes that it mistakes a standing bottle for a brick, but as the contrary was very dangerous for the lifting system, it was decided to
keep it so. When the sensor inside the clamps detect something, it lifts it up and store it in the storage area.

Now, if the front sensors detect it is a brick, the robot will turn around to avoid it. Whenever the robot is rotating, the robot will check if the sensors on the sides of the clamps detect something, to avoid the robot to hit something during rotation. If something is detected, and as it is impossible to determine the nature of the object, the robot will go back to be able to see it with the front sensors, and will check if it is a brick or a bottle. A sensor at the back of the robot is ensuring the robot will not hurt something while going back.

This is the general behaviour. It is also useful to indicate that the sensors on the clamps make several measurements to avoid mistakes such as believing there is a bottle because of an echo on the grass, and that the robot avoid rocks because it is the only features on the arena that activates both clamps sensors but not the front sensors. But what makes the system really robust is the detection of the bricks with the camera: as the ultrasonic sensors does not detect something with an higher angle than 45°, the camera is used to detect bricks that are really close to the robot and to launch an emergency routine.

3.4.2 Bottles, bricks and walls detection with camera

The first idea was to detect bottles, bricks and walls with the depth camera and to use the results got to localise via SLAM. The localisation part could not be implemented, but bricks and walls positions have been used instead for obstacle avoidance instead. In order to detect objects, tools were required to process the raw data coming from the camera. The driver was provided by the manufacturer, Prime Sense, and an open natural user interface library, OpenNI, has been used to automatise the data collection as images. From there, two different approaches have been taken.

First, the Internet has been sought in order to find libraries interpreting the depth images as 3D objects. After a few research on the web, a library called Point Cloud Library (PCL) was found. This library was pretty effective for bricks, bottles and walls detection, but compatibility problems between it, the Cubietruck and the Xtion driver. Hence the second approach: as existing libraries could not be used, an homemade library adapted to the needs of the project has been designed. This second method has been implemented beside trials to use the PCL library as the problems rose. After a few decade hours of unsuccessful trials to fix them, the second approach has been used instead and worked well for obstacle avoidance, but the loss of time resulting prevented the SLAM to be implemented during the project duration.

Here will first be exposed the PCL approach as a lot of effort had been put to adapt it when this solution was rejected, and then the homemade library will be presented.

3.4.2.1 Point cloud library

This library is called Point Cloud Library because it is mostly made to work on C++ object called "point cloud". It has a lot of pre-made functions to process the point clouds, including a lot of filters and segmentation very useful in this project. It was very simple to install on a usual computer, but it is an other thing to install it on a unusual processor as will be explained later.

3.4.2.2 PCL and xtion camera

Now that a library suited for our requirements was found, the challenge was to make it work with the camera. An existing tutorial was found showing how to have a code doing multi-threading and acquiring data from the camera. At the start this was not working because it required additional plug-ins, but after that the camera data could be extracted.

The most important difference with how openni extract the data is that PCL is directly putting that into a point cloud object and it contains the xyz coordinates and even rgb of each point.
3.4.2.3 Planar segmentation

The planar segmentation is a technique to find the main plane over a point cloud. Hopefully PCL has a function that is doing that. It needs to be set with different parameters. The most important ones were the threshold at which a point stops being in the plane, the minimum number of points in a plane, and the number of iterations to find the best plane. The planar segmentation method also creates an object containing the plane parameters. An example of a multiple plane detection can be seen on fig 18. This test was done with the camera pointing to the corner of a room. The three different planes are displayed in different colors. To find a new plane with the planar segmentation when one already found, all the points contained in the first plane must be suppressed from the point cloud. Using this technique, all the planes that exist and contain enough points can be found.

![Figure 18: Room angle multi-plane recognition](image)

3.4.2.4 Clustering

An other technique used was the clustering. It is the action of fragment a point cloud in different clusters of points. A cluster is a point cloud where all the points are distant from each other less than a threshold. This fragmentation action called clustering will help us to find the clusters of points showing the presence of bottle or obstacles. Again hopefully there is a method in PCL to do clustering. It takes as interesting parameters the threshold distance between points, minimal cluster size, and maximal iteration to look for them. But as the bottles and bricks are touching the floor, there is a good chance they will not be considered as clusters. This is why the floor points need to be subtracted from the point cloud previously, and to find them, the planar segmentation that was presented just above is used. This procedure was made on the configuration showed in fig: 19. There are two bricks and five bottles. After floor recognition and its points suppression and clustering of the remaining point cloud, that is what was obtained fig: 20.
If the points of the floor detection are added to the result using yellow the result of fig: 21 is obtained. A second plane that is a wall was also found and is added in pink.

Figure 19: Scene on which the example was made

3.4.2.5 Obstacle and bottle differentiation

To differentiate between bottles and bricks simple geometry was used. First the highest point of a cluster is found and then the distance between this point and the floor is calculated. If this distance is smaller than a threshold, the cluster is considered as a bottle.

A problem occurred when the floor planed was not identified first due to the smaller number of point contained that a wall for example. To correct that, the angle of all the planes identified is computed and compared to the supposed floor angle and the closest one will be the floor. If none of them is close enough, the supposed floor plane parameters will be used to difference between clusters and bottles.

3.4.2.6 The use of an homemade library instead

As PCL could not be used, a homemade library has been implemented. It is way simpler than PCL but is adapted to the project needs. With the help of a basis image taken without any objects, and thanks to some basic operators such as the subtraction of two images, threshold, a Gaussian mean or a Sobel filter, all the pixels belonging to the floor are removed from the image. Then, the remaining pixels are interpreted as bricks, walls or bottles depending on their size in pixels and their average distance. This algorithm worked perfectly and could even detect a brick in front of a wall, or distinguish two bricks distant by only a few centimeters. It was less efficient.
Figure 20: Clusters constituting the bottles and bricks

Figure 21: Bottles, bricks and floor
on bottles as they are partially invisible to them, but the most important part is to be able to
detect bricks and walls to not rush against them; the bottles can be detected by the ultrasonic
sensors. An example of brick and walls extraction is shown in figure 22. As this program was
efficient to localise still objects, it could have been used to implement SLAM.

Figure 22: An example of wall and bricks extraction with the homemade library. The software
displayed the position of the three bricks and the wall relatively to the robot.

3.4.3 Odometry

3.4.3.1 Precision, error and relocalization

The problem using only odometry for localization is that there is an error that increases with the
distance travelled by the robot. The odometry was implemented with a simple algorithm using
angle approximation. The position update was called 20 times per second so this approximation
was justified.

\[
\Delta x = \cos(\alpha)(l_1 + l_2)/2
\]

\[
\Delta y = \sin(\alpha)(l_1 + l_2)/2
\]

\[
\Delta \alpha = (l_1 - l_2)/2
\]

After calibrating the angle and the distance, the robot was able to rotate to angle with -2 to
2 degree of mistake and was able to do linear distance with -3 to 3 centimeters mistake over 3
meters. The robot was able to move around, travel 15 meters and then come back at the starting
position with and error smaller than 40 cm. Sadly there was no more time left to use this ability
to bring back the robot to the recycling area.
3 METHODOLOGY

Figure 23: Error propagation representation. This figure shows how the error in position and the error in orientation increases as the robot moves. The robot follows the trajectory represented by the black arrow. The green circles represents the area where the robot thinks it could be and the red triangles represents the orientation the robot thinks it has. The errors showed in this image are scaled, this is only a representation.

To reduce the error related to the encoders, the following strategy has been thought. Once the robot has travelled a given distance, it will aim an angle of the arena and thanks to the ultrasonic sensors, it will know when it actually reaches to the angle. Once it arrives to the desired position, it will change its orientation and the odometry is recalibrated.

The key of this system is to do the recalibration before the robot has an error too big. In fact, if the error is too large, the robot may goes toward the wrong corner of the arena, therefore all the calibration would be wrong.
4 RESULTS

At the end of the semester, during the public presentation of the competition, this robot managed to collect a total of eleven bottles, from the easy zone and the grass. These bottles provided a score of 32.5, a quarter of their points as the robot didn’t bring them back to the recycling area. This afford the robot to raise the second place ex-aquo amongst the five robots.

During the competition, it has not hit an obstacle, the algorithms for obstacles avoidance and the ones to prevent the robot to go on the rocks worked pretty well. The behaviour during some obstacle avoidance was characteristic of an emergency routine sent by the Cubietruck from the camera data, which demonstrates its need and efficiency. The robot have run the whole time without the need of any external intervention except one time, when a bottle has been put in proximity just to show some more grabbing actions for the public. In fact the robot already gathered the majority of the bottles from both the areas it goes through before the end of the ten minutes, and nine bottles out of the eleven it got were collected in less than six minutes.

The power supplied by the battery was enough for the ten minutes as the robot kept moving until the end of the count down, a lack of energy being observed though in lifting device during the last seconds.

The four motors were very efficient and allowed the robot to move quickly even if its storage area was full of bottles.

The clamp didn’t broke even if it received some stress due to the grass. The elastic system was very reliable allowing the clamp to keep grabbing the bottles.

The connections between the different boards was not broken. The camera could send its information to the Arduino during the whole time and the Arduino could sent commands to the motors control board continuously.
5 Discussion

5.1 Discussion of the performances

Even if the way back home could not be achieved during the competition, the minimal requirement, the public entertainment, could be fully achieve, as the robot was able to grab many bottles and to avoid every obstacles. The plan was then followed: the ultimate robot could not be achieve, but the minimal requirements are met and the robot got can be improved in order to approach from the ultimate robot. This has been possible thanks to the systematic methods described above, and to the redundancy of the detection allowing to rely on another sensor when one was not appropriate.

5.2 Possible improvements

For the possible capacity improvements, there could have been the possibility to go back to the recycling area with different methods, the odometry or the camera. The odometry allows the possibility to draw a vector that points toward the recycling zone. The 3D camera could allow some SLAM implementation and this allows the robot to know its actual position and to know the map of the arena. This last method is really efficient if the map doesn’t change and if the robot has the time to pass multiple times in the same places. Odometry was available during the competition and thus the localisation activated, so the robot was aware of its approximated position; but even if the walls and bricks detection was efficient with the camera, time lacked to implement the correction of the belief and the way back home.

In order to reach more complex areas like the one behind the rocks or the other up the slope, some code could have been implemented in order to rise the clamps and continue for example. In that way the robot wouldn’t have hit anything with the clamps, therefore it could have passed and collect bottles that have a bigger value; but it will have to be sure there is no obstacles beyond, as the camera is not available with the clamps up.

The more tedious part for the displacement of this robot is that the clamps are not on the center of rotation of the robot when it turns on itself. This cause some difficult code implementation in order to ensure safety to the robot when it turns. This is the reason why the robot could be a bit different shaped if a new one would have to be built.
6 Conclusion

The solution chosen for the mobility of the robot is to use four actuated wheels and to stick tape on the tires in order to reduce the friction between tires and carpet of the arena. The bottles collecting system is a clamp that grabs the bottles and drops them into a storage area into the robot that can contain up to twelve bottles. To empty the storage area, a backdoor has been designed and is opened and closed by a servo motor. The robot detects obstacles, bottles and walls thanks to a total of eight ultrasonic sensors and a 3D camera called Xtion that ensure distant and contactless detections. The encoders that are connected to the passive wheels in the middle of the chassis allow odometry therefore localization.

The global structure consists in a scheduler that calls several functions to check the data coming from the ultrasonic sensors and the Cubietruck managing the camera and to react on the motors. The behaviour of the robot is modified thanks to a state machine. One main goal of the code for this robot was that it has to be robust and ensure perfect obstacle avoidance. Moreover it has to detect and gather bottles. In order to reach the robustness needed to avoid collisions, thanks to the ultrasonic sensors and the camera, a redundant close objects detection has been implemented so that if the camera or the ultrasonic sensors have some malfunction, the robot will still avoid obstacles and walls and try to collect bottles, even if it won’t be the optimal solution anymore.

The robot of this project arrived second on the competition collecting a total of eleven bottles it took from both the easy area and the grass. The algorithms for the obstacle and rocks avoidance, for bottle gathering and for displacement worked perfectly. The robot never got stuck, never hit any obstacle or walls, collected bottles when it could and therefore this robot has needed no external intervention and can be fully autonomous.
The authors would like to thank:

- Auke Jan Ijspeert for having given us the opportunity to do this wonderful semester project
- Alessandro Crespi for having dedicated a considerable amount of energy to make everything work until the very details
- Massimo Vespignani for his advices and his time all the time during the project
- Jean-Paul Brugger, Jean-Pierre Grobet and the entire workshop team of the AEM for the manufacturing of the robot pieces and their advices about how (not) to design them
- All the other assistants that presented us a workshop or helped us in a way or another
- The other groups that maintained a very friendly atmosphere during the whole project, helping us when it was useful and sharing a beer when it was necessary, and especially:
  - Group 1 for their M4 nuts and for having taught us how to properly avoid vibrations with a bicycle tire inner tube
  - Group 3 for the coffee machine and their battery box design
  - Group 4 for singing right and knowing how to find the nearest multimeter
  - Group 5 for sharing their multimeter and having accepted to combine their burnt motors with usable gears with our usable motors with broken gears
- The Coop Group for providing food on Sunday
- Our parents for having designed us handsome and smart