Dumpsty

P5 Project
Group SW510E16
Software
Aalborg University
21st Dec 2016

AALBORG UNIVERSITY
STUDENT REPORT

Fourth semester at
Department of Computer Science
Software
Selma Lagerlöfs Vej 300
9220 Aalborg East, DK
http://www.cs.aau.dk/

Title:

Dumpsty

Project:

P5-project

Project period:

September 2016 - December 2016

Project group:

SW510E16

Participants:

Christian Dannesboe
Frederik Børsting Lund
Karrar Al-Sami
Mark Kloch Haurum
Lasse Lyngø Nielsen
Søren Lyng

Supervisor:

Bent Thomsen

Pagecount: 29
Appendix range: X
Published 21-12-2016

Synopsis:

Preface

This project has been developed as part of the fourth semester project by project group SW408F16 from Aalborg University, Software Engineering, from the period 1st February to 26th May 2016 .
The project is based on the *Aalborg-model*, study method, where problem and project based learning is the focus. The theme of this semester was to create a compiler for a new language. To do so, some subjects were introduced and the subject the group chosen, was *Domain Specific Language for Robocoders* .

The group would like to thank supervisor, Giovanni Bacci for his very much appreciated advice and guidance during the whole project.

Signatures

Christian Dannesboe	Frederik Børsting Lund	Karrar Al-Sami
Mark Kloch Haurum	Lasse Lyngø Nielsen	Søren Lyng

Reading guide

This project has followed the courses Syntax and Semantics & Languages and Compiler. The context of this project has been written according to the order the course materials was taught and learned.

The sources in the report are being referred to by the Harvard citation method. This includes a last name and a publication year in the report, and in the ***Bibliography*** chapter all the used sources are listed in alphabetical order.
An example of a source in the text could be: ***[?]****.*
If the source is on the left side of a dot, then that source refers only to that sentence and if the source is on the right side of a dot, then it refers to the whole section.

Figures and tables are referred to as a number. The number is determined by the chapter and the number of figure it appears as.
For example: The first figure in a chapter will have the number ***x.1****, where x is the number of the chapter. The next figure, will have the number* ***x.2****, etc.*
The listings of source code are also referred to as the tables and figures.

Source code in the report are listed as code snippets, and they're not necessarily the same as the source code, meaning that code snippets may be shorter than the actual source code or missing comments from the source code. In order to show that, the use of the following three dots are used: "...", which means that some of the source code isn't listed in the code snippet, as it may be long and irrelevant.

Contents

1 Introduction **1**

2 Analysis **3**
2.1 User story . . . 3
2.2 User requirements . . . 3
2.3 System capabilities . . . 4
2.3.1 Throwing . . . 4
2.3.2 Detecting and tracking . . . 4
2.3.3 Catching . . . 4
2.4 Problem statement . . . 5

3 Theory **7**
3.1 Field of view . . . 7
3.2 Throwing . . . 7
3.3 Detecting and tracking . . . 7
3.4 Catching . . . 8
3.5 Trajectory prediction . . . 8

4 Hardware **11**
4.1 Sensors . . . 11
4.1.1 Microsoft Kinect . . . 11
4.2 LEGO NXT Gyroscope . . . 11
4.3 LEGO NXT Servo motor . . . 11
4.4 Arduino . . . 11
4.4.1 Arduino WifiShield . . . 11
4.4.2 Arduino MotorShield . . . 11

5 System requirements **13**

6 Design **15**
6.1 The robot . . . 15
6.2 Throwing . . . 15
6.3 Detecting and tracking . . . 15
6.4 Catching . . . 15

7 Implementation **17**

8 Tests **19**

9 Discussion **21**

10 Conclusion **23**

11 Future work **25**

Bibliography **27**

A Appendix **29**

1 Introduction

An embedded system is a computer system which only have a few functions. It is embedded as a part of a whole device, which then also includes hardware and/or mechanical components. Embedded systems are everywhere in our everyday lives. The range for embedded systems could be all from saving lives with pacemakers to fun gadgets.[Techopedia.com]

An embedded system as a gadget, is a technological and small object or apparatus, which has a certain functionality. This functionality is limited to a certain niche.
In this project, the embedded system in consideration would be designed as a gadget, with a sole purpose of catching trash thrown at it. A trash bin that can catch trash would make the act of cleaning more fun and interactive, and the availability of each individual trash bin would be tremendously increased.

For this gadget to be usable, it must be designed as a real-time system, as this system will have certain deadlines for each task to be executed in time. The trash bin must be able to identify an object coming towards it, make some computation to specify a point to catch it from, and then compute a path to catch the object. All this must be done before the object lands on the ground, which makes the use of real-time system design a part of this project. A real-time system is a software system which is subject to a real-time constraint, and the system must control and affect an environment, by receiving data and process these, within a certain time limit.

Analysis 2

From the analysis chapter it should be possible to derive the requirements for the smart waste bin, Dumpsty. The user stories will be initial and will help make the user requirements for the project. After the user requirements the information from the user stories will be analysed in greater detail, depicting three different phases of the process: Throwing, detecting/tracking and catching. These three phases will be the inspiration for the system requirements. The analysis chapter will end up with a problem statement for the project.

2.1 User story

As mentioned before the user stories should help make the user requirements for the project, one user story have been made that showcase the use of Dumpsty.

Benjamin is a software engineer who is tired of wasting his precious time on the job with walking back and forth from the waste bin in his office, and therefore wants to be able to throw his trash in the general direction of the bin instead. His aim when throwing the trash isn't that of a trained basketball player, so he often has to pick up the litter after throwing it at the bin.
Benjamin wishes that the waste bin could move and collect the trash for him, so that he can throw his trash in the general direction of the bin, and the bin could then place itself in a way, that allows it to catch the trash before it lands on the floor. This would optimize the time Benjamin uses each day on collecting the trash he did not land in the waste bin.
If Benjamin throws at the bin from a designated side of a sensory camera, the robotic waste bin should identify the trash, move the bin to a place where it would be able to catch the trash, before it hits the ground.
If Benjamin throws outside a designated perimeter of the robotic waste bin, it should not try to catch the litter, as it would compute that it is not able to get to the point of catching before the trash hits the ground.
If Benjamin and another person from his office throws trash to the waste bin at the same time, it should prioritize the first identified object.

2.2 User requirements

The user requirements of the project have been conducted from the section 2.1. User requirements are simple requirements written in a natural language, which should help everyone get an understanding of the requirements for the project.

- The waste bin must be able to move itself to the colliding position of the trash within a certain time limit, with a certain precision.
- The trash is always thrown from a designated side, according to the camera sensor.
- The waste bin will always start from a designated position, according to the camera sensor.
- Multiple trash thrown at the waste bin will not be considered.
- The camera sensor must be able to identify a specific object, which would be considered as the trash in this report.
- The waste bin should only consider trash thrown within a certain perimeter of itself, according to the attained maximum speed of the robot.

2.3 System capabilities

This section will define the required capabilities of the system, which will include the functionality and hardware needed to fulfill the tasks of the embedded system. This is divided into three different categories, which are the subsections of the section. These categories, along with the user requirements, will define the system requirements for the project.

2.3.1 Throwing

To make it easier for the sensor to detect and track the thrown object. The object should be round of shape and in a clear colour such as red. This will help the sensor because it the object will stand out from other objects.
The ball should be thrown as slow as possible, to give the sensor and waste bin more time to detect, track and calculated the collision point for the object and the waste bin. This could be done by throwing an underhand throw or bouncing the ball at the area of the waste bin.

2.3.2 Detecting and tracking

Detecting and tracking are a very essential part and is considered as one phase. For detecting the sensor should be able to recognize the specified object within it's field of view. The waste bins area and the area in front of the waste bin should be monitored by the sensor.
When the object have been detected, it should then be tracked by the sensor. The sensor should track the speed of the object and the direction it is heading in. This should be used to calculate the objects collision with the waste bins area.

2.3.3 Catching

The catching phase of the project is for the waste bin to calculate the objects collision within the waste bins area, and the movement to the designated collision point. The waste bin gets information about the flying object from the sensor and uses the given information to calculate the collision point. What data the waste bin gets from the sensor and how the calculation is done, will be explained in later chapters.

2.4 Problem statement

Based on the above analysis and its limitations a problem statement for the project has been constructed:

How can an embedded system control a waste bin to detect, track and catch a thrown object within a designated perimeter?

Theory 3

3.1 Field of view

Skal vi have noget om det synsfelt vi har til at opfange bolden i??

3.2 Throwing

When throwing the ball towards the trash bin, the group have two main methods in order to ensure that the sensor gets the most reliable data.

The first one being, that when throwing the ball, one must be standing with the camera/sensor to his side. From testing the Microsoft Kinect this has provided the best test results, when measuring for the distance of throw and comparing to the distance the sensor has given us.

The second method, is when throwing the ball, the trajectory curve of the ball, should have a high height so that it will cover more distance. In the future, this should not be a requirement either, but as of now, it should in order to give us more time to track, calculate and catch the object.

3.3 Detecting and tracking

The way the trash bin detects and track an object, happens when the object is of a defined form, such as a tennis ball, which we have chosen for this project. By predefining the object the sensors should look for, the error margin becomes smaller, and therefore this is how we plan on doing it, for this project. For the future, this point should not be predefined, as trash can be in various forms.

In order to ensure that the sensor detect where the object is, the object has to come from a specific angel, that provides the sensor with the best actual distance. Once again, this is just for the project as of now, and in the future, this should not affect the way detecting works, as trash can be thrown from multiple angels, depending on the location of the person and the trash bin.

3.4 Catching

3.5 Trajectory prediction

When the trash, referred to in this section as the projectile, is detected and the tracking of that projectile has started, the trajectory can be predicted. This prediction is limited to the amount of data sent by the sensory camera, meaning that for every camera reading, one detection of the object is gained. The precision of the prediction will increase according to the time a projectile has been tracked.
In this project, since the prediction is done indoor, the outdoor weather conditions that might affect the projectile trajectory is not considered. As well, the effects of air resistance, also called drag, will not be considered.

The trajectory of a projectile is the path of a thrown projectile without propulsion, affected by gravity. For calculating a trajectory of a projectile, the initial height, the angle which the projectile is launced from, the speed of the projectile at launch and the gravitational acceleration must be taken into account.
The initial height in this project is the height at which the projectile is detected, and the angle and speed at which the projectile is launched will be calculated from the first few trackings after detecting the projectile. The gravitational acceleration is considered as $9.81m/s^2$, which is the standard near the earths surface.
In this project we are interested in catching the projectile, and to do that, we need to calculate the distance the projectile travels before hitting the ground, and the amount of time before the projectile hits the ground. This is done with two mathematical formulas:

g : *the gravitational acceleration* $(9.81m/s^2)$
θ : *the angle at launch*v : *the speed at launch*
y_0 : *the initial height*
d : *the total horizontal distance traveled*
t : *the time of flight*
v_{vert} : *the vertical velocity*
v_{hori} : *the horizontal velocity*
t_h : *the time since first detection*
d_t : *the distance at time t*

From these variables we can express the formulas needed to provide the total distance traveled by the projectile, and the amount of time this would take.
The distance traveled is:

$$d = \frac{v\ cos\ \theta}{g}(v\ sin\ \theta + \sqrt{(v\ sin\ \theta)^2\ +\ 2gy_0})$$

The time of flight is:

$$t = \frac{d}{v\ cos\ \theta} = \frac{v\ sin\ \theta\ +\ \sqrt{(v\ sin\ \theta)^2\ +\ 2gy_0}}{g}$$

To calculate a trajectory of the projectile, the altitude and distance of the projectile at any time during the flight must be calculated, according to the initial height (y_0):

$$y = v_{vert}t - \frac{1}{2}gt^2$$

$$d_t = v_{hori}t$$

After this section, the projectile can be tracked, and the path for a given projectile can be calculated to a certain degree of correctness.

Hardware 4

4.1 Sensors

4.1.1 Microsoft Kinect

4.2 LEGO NXT Gyroscope

4.3 LEGO NXT Servo motor

For the robot to be able to move, it would need motors. In this project the LEGO NXT 9v Servo motors has been used, which at full power with no load can reach 170 RPM. These motors has a gear range of 1:48, split on the gear train in the motor. This motor includes an optical fork to provide the rotation sensor functionality, which can provide data of motor rotations down to a 1° precision.

4.4 Arduino

4.4.1 Arduino WifiShield

4.4.2 Arduino MotorShield

System requirements 5

Design 6

6.1 The robot

6.2 Throwing

6.3 Detecting and tracking

6.4 Catching

7 Implementation

Tests 8

9 Discussion

10 Conclusion

Future work 11

Bibliography

Techopedia.com. Techopedia.com. *Embedded Systems*. `https://www.techopedia.com/definition/3636/embedded-system`. Accessed: 25-09-2016.

Appendix A