Adam Veraszto, Daniel Balint

CraftLikeABosch Software Development Challange

Project Documentation

11/28/2021

Budapest

1 Introduction

Our task was to simulate of an ultrasonic parking system, including object detection, collision avoidance and modelling the direct and cross-echoes from the different ultrasonic sensors. We started our work by laying out the the mathematical concepts for ray tracing of the ultrasonic wavefronts so that we could simulate cross and direct echoes in certain scenarios appropriately. Afterwards, we designed the basic architecture of our software solution. We used Python 3.9 and the PyGame environment to implement our solution, and we created an interactive GUI with control buttons, Drag&Drop features and value indicators.

2 Mathematical Concepts

There can be two type of possible obstacles on the playing field (circle and rectangle), both could be a cause for direct or cross-echo phenomenon, which then results in four possible base scenarios.

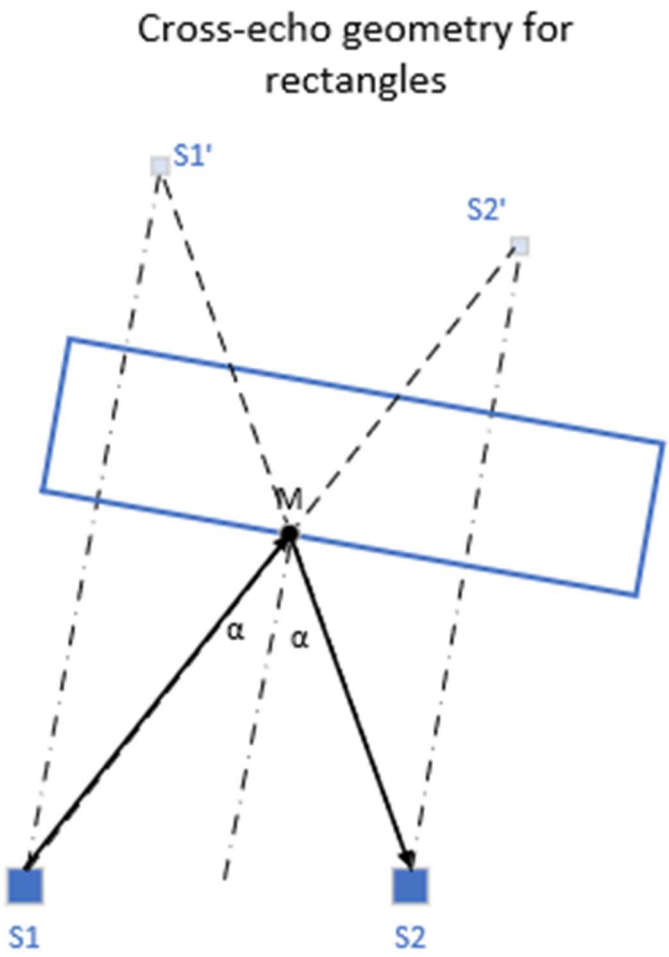
2.1 Direct echo from rectangles

Calculating the direct echo point between a sensor and a rectangular object can be done by taking the normal vector of the sensor facing side, then a line is set through the sensor point in the direction of the normal. Then we calculate the intersection of the two lines, which effectively is solving a linear system containg the two line equations. If the intersection is in range, we have our echo point.

2.2 Direct echo from circles

By connecting the sensor point and the center of the circle, we gather a line that intersects the circle at the echo point and has the same direction as the direct echo ray. With radius of the circle, the sensor and center coordinates, the intersection coordinates can be calculated, and we arrive at another direct echo point if the intersection is in range.

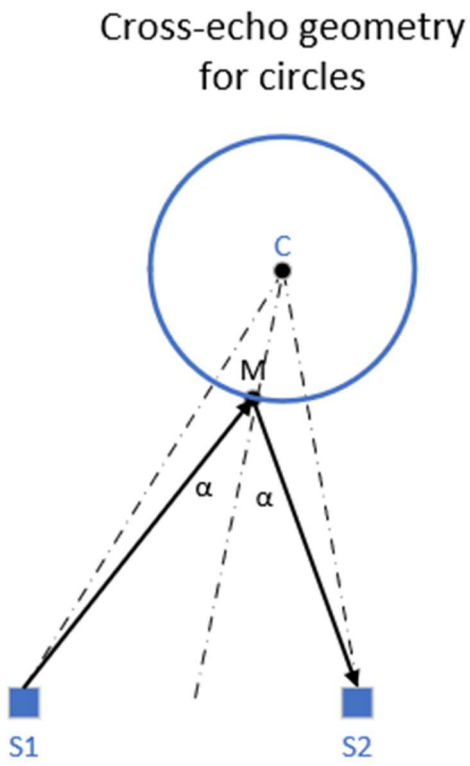
2.3 Cross-echo from rectangle



We could arrive at the cross-echo point for rectangle by mirroring both sensor points to the sensor facing side of the rectangle as seen in the figue above. By connecting each sensor with other sensor's image, we gather the intersection point (**M**) of S1S2' and S2S1' line, and it can be proven by basic geometry that the perpendicular through M is a bisector of the S1-M-S2 angle. Thus, M is the cross-echo point. In practice, we solve the linear system of the two (S1S2' and S2S1') line equations to arrive at M.

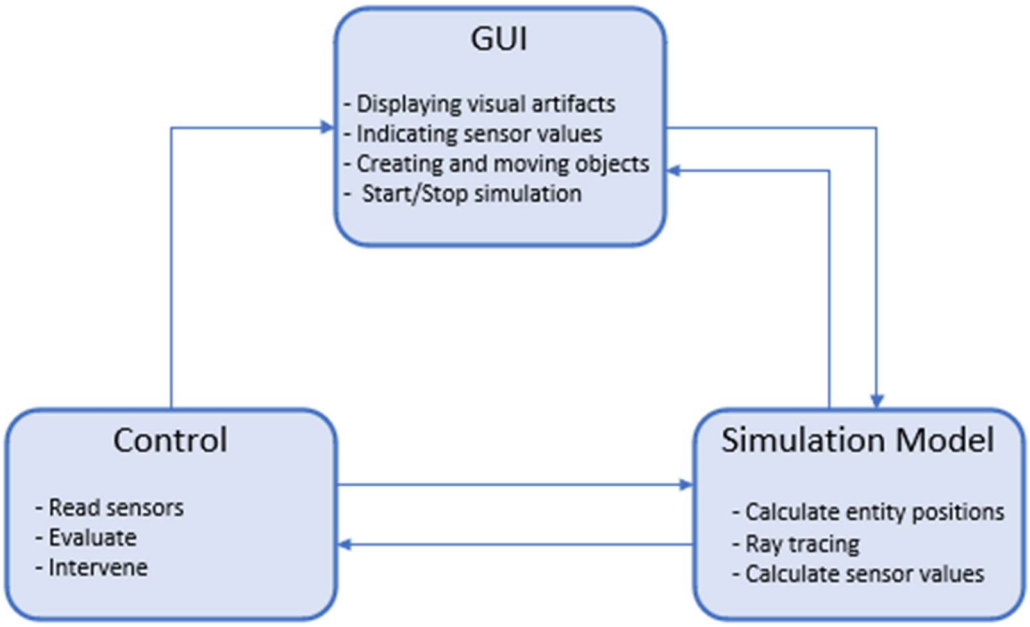
2.4 Cross-echo from circle

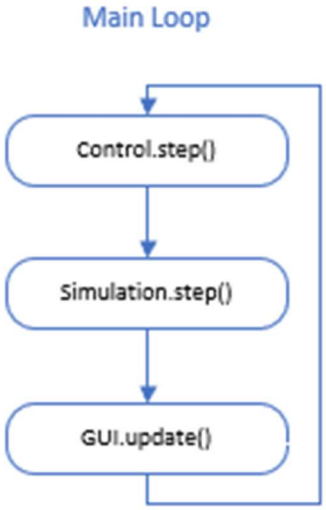
In case of a circle, considering geometrical principle, it can be seen that the corss-echo point will have a M position so that the S1-M-S2 will be minimal. The minima will be on sector cut out by the S1-C and the C-S2 line. We used brute force to find the minima.



A possible way to arrive at an exact solution would be using a parameterized ellipse with focuses in S1 and S2. The ellipse represents a set of locations on the plane such that the sum of distances to the focuses is constant. As we increase the size of the ellipse, it will eventually touch the circle, and that touch point will be the minimal sum of distances between S1, S2 and the circle. However, the equation of the circle and the ellipse leads to 4th-degree system, which is impractical.

3 Software Architecture





4 User manual

In this section a short user manual for the graphical user interface (GUI) of the project will be presented. The GUI is divided into two major parts. The upper half of the window contains the simulation area, while beneath it the dashboard can be found. The simulation area contains the car object with the ultrasonic sensors. The objects of the area can be moved in a „drag and drop ” way. The car can only be moved along the horizontal axis, while the position of the rectangle and circle obstacles can be manipulated freely in the simulation area. The rectangles can be rotated by scrolling the wheel of the mouse while the cursor is over them. The objects can not be placed on each other. New obstacles can be dragged in from the „drag-in” area at right side of the dashboard by dragging in the desired shape. The size of the new element can be changed by selecting a number from 1 to 10 in the „size picker” tool, the selected number is proportional to the size of newly created element.

The middle of the dashboar shows the measured distances of the sensors. If there is no incoming signal, then a cross is placed at the appropriate cell. The range of the sensors are represented with yellow cones in the simulation. The direct echoes of the sensors are shown with green lines, while the purple lines describe the rays of the cross echoes. Depending on the measured distance values the boxes representing the sensors (Sensor 1 -> Sensor 6 from top to bottom) change their colors from green to red as the obstacle is getting closer. (green -> yellow -> orange -> red) .

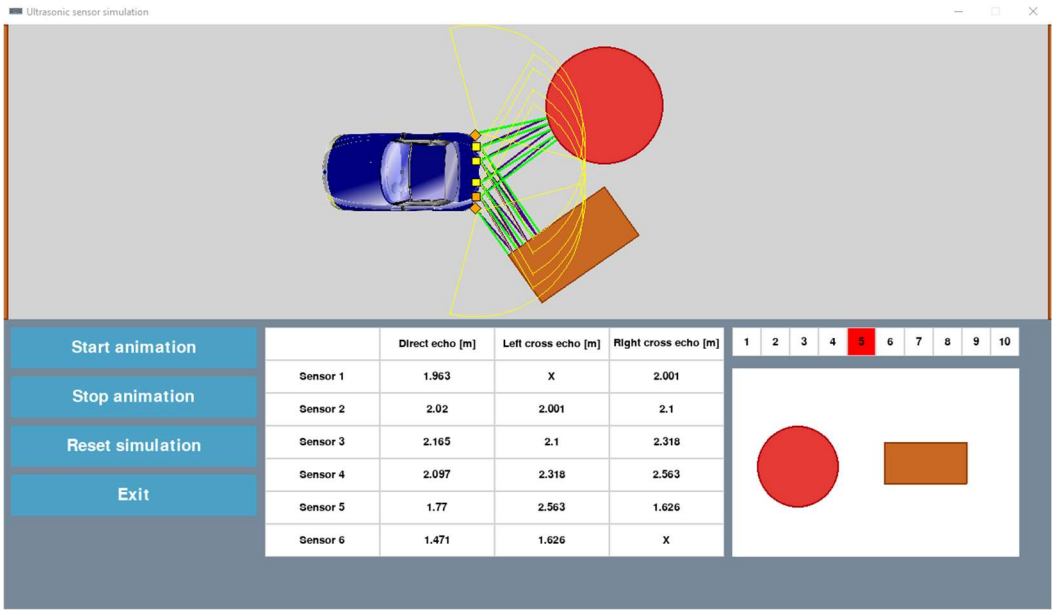


Figure 1 Screenshot of the graphical user interface

On the left side of the dashboard, the control buttons of the application can be found. However, the simulation is always running and the elements can be „dragged and dropped”, there is another function, where the car start reversing. This function can be started with the „Start animation” button and stopped with the „Stop animation” button. During this „animation” the car reverses if it has no obstacles in dangerous proximity. If an obstacle is placed behind the car, then it stops based on the measured values of the sensors. There are border obstacles placed at the sides of the simulation, therefore the working principles of the sensors can be demonstrated without even dragging in any new obstacles. The „Reset simulation” button deletes the obstacles added by the user and reset the car to its initial position. With the „Exit button” the application and the script running under it can be terminated.

5 Install

The application was written in Python language. It was tested with Python 3.9. The dependencies of the project can be found at the „requirements.txt” file of the project and they can installed with the „pip install requirements.txt” command.