Activity Bot's Catch Game
Ultrasonic Sensor Fusion for Object Tracking

Bruno Silva
bruno.silva1@student.um.si
Faculty of Electrical
Engineering and Computer Science,
University of Maribor
Koroška cesta 46
SI-2000 Maribor, Slovenia

ABSTRACT

This paper describes the development of two small robots from *Parallax*, the *Activity Bots* - a seeker (Figure 1) and a runner (Figure 2) - that use ultrasonic sensors to play the *Catch game*. The seeker robot was programmed to detect the runner robot and chase it, while the runner robot was programmed to evade the seeker robot. The paper discusses the design and construction of the robots and the algorithms used to control their movements. The results of experiments conducted to test the robots' performance in the game are presented and analyzed. The experiments show that the seeker robot was able to detect and pursue the runner robot successfully. This paper also goes briefly on how to incorporate the workflow of parallax's robots programming in the popular text editor VScode.

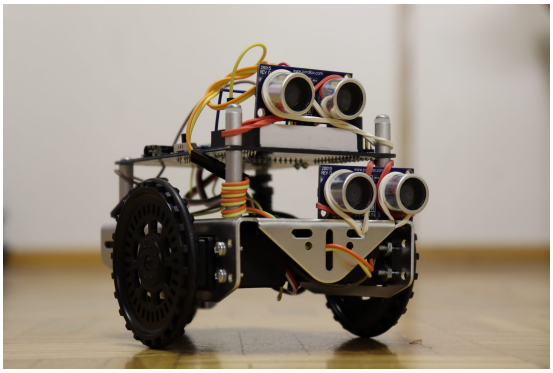

Figure 1: Seeker Robot.

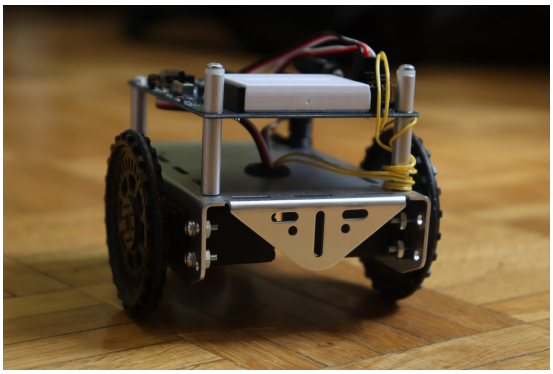

Figure 2: Runner Robot. PLACEHOLDER IMG

1 INTRODUCTION

Tracking and detection algorithms are the base for many applications in robotics field. There are many examples of other studies backing this up, such as the use of Yolo's deep learning algorithms for object detection [1], or the use of tracking algorithms for robot control [2]. While RGBD cameras or 2D/3D LiDARs are often the preferred choice for most applications[3], ultrasonic sensors, even though simple and quite less powerful, are sometimes a good alternative due to their much more affordable price[6]. The approach used in this research has some reassemblance with the working principles of stereo cameras[4]. Using 1 ultrasonic sensor proved insufficient to predict the target's direction. Switching to 2 sensors and measuring deltas instead of absolute values allowed the seeker to perceive the direction the target was coming from. Some studies show the credibility of fusing ultrasonic sensors [5]. This working principle will be further expanded on in the seeker's robot chapter of this paper.

2 DEVELOPING ENVIRONMENT

This chapter is meant to explain the working environment of the programming of Parallax's robots and the work flow that was used throughout this project

2.1 SimpleIDE

SimpleIDE is Parallax's official developing platform. As the name suggests, it's a rather feature lacking IDE as shown in 3. Basic features like autocompletion, keyword highlight, or even error squiggles are not present. Lack of customizability is what sets it behind, not being simple. Other editors like Vim are also pretty simple by default and are very popular due to their unmatched customizability ceiling.

2.2 VScode Integration

Fortunately, it is possible, with some workarounds, to migrate the grand majority of the workflow into a more popular environment like VScode.

(1) PlatformIO
- PlatformIO's IDE is a VScode extension which transforms VScode into a IDE capable of fully programming a wide range of microcontrollers, from code writting to

Figure 3: SimpleIDE working environment.

EEPROM flashing. There's a platform there for working the Parallax's microcontrollers (Github). However, upon exhaustive testing this platform seems to be a half finished attempt which doesn't work.

(2) VScode
- One way is to add SimpleIDE's libraries to the default C path for the default C libraries. This approach is better for long term development, but on this project wasn't used.
- Another way is to import the libraries straight into the project's folder, allowing VScode to treat them as user built libraries.
  This was the approach used. To replicate the following folders have to be copied to the project folder:

```
$ /home/${USER}/Documents/SimpleIDE/
Learn
$ /opt/parallax
```

  It's important to note that this applies only for Ubuntu based systems and that both these folders should be added to the repository's *.gitignore* as they are quite big and not important to the project itself.

3 SEEKER ROBOT

The core of the seeker's algorithm is these 2 steps:

(1) Assessing targets in range of ultrasonic sensors;
This involves stopping, getting 2 measurements from each sensor spaced by a predefined $\Delta t[s]$, calculating the $\Delta d[cm]$ from the right and left sensor and figure out which direction the target is coming from and outputting accordingly.
(2) Giving moving order to the servos to chase the target. In this step the purpose is to, given the direction the target crossed from, tune how much the robot should rotate and go forward. This ideally would be dynamically controlled, but since the robot cannot know how fast the target crossed, it has to be manually tuned for a given target velocity.

3.1 Assessing Targets

The first thing this function does is assure the robot is still before reading the differential distance measures. The code could account for the velocity of the seeker and correct the measure, but that would be mathematically a lot more challenging.

Then, after reading the distance measurements, the robot could be left with 5 possible case-scenarios given the 2 differential measures :

(1) $\Delta D_L < 0 \wedge \Delta D_R \approx 0$
This would mean the target came from out of range and is started being detected the by left sensor, since the measure distance reduced. The robot should thus turn **RIGHT**;
(2) $\Delta D_L \approx 0 \wedge \Delta D_R < 0$
This would mean the same as case 1, but instead the target is approaching from the right, so the robot should turn **LEFT**;
(3) $\Delta D_L > 0 \wedge \Delta D_R \approx 0$
This would mean the target was already in sight of the left sensor and left suddenly to the left, allowing the sensor to see into the further away background. This would mean the robot should turn **LEFT**;
(4) $\Delta D_L \approx 0 \wedge \Delta D_R > 0$
This would mean the target was already in sight of the right sensor and left suddenly to the right, allowing the sensor to see into the further away background. This would mean the robot should turn **RIGHT**;
(5) $|\Delta D_L| > 0 \wedge |\Delta D_R| > 0$
This would mean that the target is currently in front of the robot and is either getting further away or getting closer. Either way, for the seeker robot the order should be to go **forward**.

In practice, a *threshold* was used instead of 0, to account for sensor measurement error.

3.2 Measuring distances

Initially, in order to assure the robot could always read the latest reading from the sensor when needing to evaluate distances, and to take advantage of the parallax's multicore chip, the sensor data was being stored on a global variable that was constantly being updates parallel to the code. The code for acquiring data was running on a separate core. However, due to timing reasons, this proved to be more harmful to the overall performance of the robot. In the end, whenever the robot isn't actively pursuing the target, he is in a semi-idle state doing nothing other than to measure distances to try to identify a potential target.

3.3 Issues with the current implementation

- As of right now, the seeker isn't identifying how close the target is. As of right now, the robot upon identifying the target's direction, its servos get an input for a set amount of time at a set power. A runner coming into frame further away should result in a smaller turn time while another closer away should result in a sharper turn. Finding a function that correlates the distance to the correct turn amount is not an easy task tho, a trial and error approach could be used, but such approach is reckoned to be too volatile and sensitive to environment changes.
- Using ultrasonic sensors instead of cameras basically limits the robot perception field into a line instead of spacial. This is means if for some reason the runner gets completely out of the runner field of view, the seeker becomes clueless

again until the runner comes back into frame again. One approach to this problem could be to have the seeker pan around itself whenever he got more than a set amount of repeated null differential distances readings.
- The ultrasonic sensors used are not very accurate and from time to time yield outrageous readings, the robot if left facing a plain wall still for a big enough period of time, its almost certain that he will spontaneously turn because of an inaccurate reading. The sensors were replaced and switched and the problem remained.

4 RUNNER ROBOT

This robot does not possess any intelligence, it's just working as dummy robot that goes around in circles to provide the seeker something to follow. The only possible mounting position that would somewhat suit this robot is facing backwards, as there is never a scenario where the runner wants to be facing the seeker, but even then the runner ideally shouldn't be directly in front of the seeker, since it would be in its perception field.

5 EXPERIMENTS AND RESULTS

The following section goes over how the algorithm performed when faced with a few tests. It was tested the reaction time and the accuracy of the assessing, meaning if the seeker determined the right runner's direction.[1]

5.1 Reaction time

It's important to note the following : the reaction time is mainly controlled by the delay between differential measures, ΔT. In theory with a really low ΔT, the response time would be really great. In practice, this proved to have a lot more drawbacks:

- The runner robot is not 100% opaque, meaning its hollowness makes the robot's behavior with a low ΔT unpredictable.
- A low ΔT also makes the robot's behavior very jittery and nervous.

Nevertheless, with a delay of *50ms*, here are some measurements of the response time with a moving object.

Table 1: Results of the reaction time experiment[2][3]

Trial	Runner Speed[4]	Reaction time[s]
1	10	0.3
2	20	0.4
3	50	0.6
4	70	0.7
5	100	-
6	150	-
7	200	-

[1]There are demo videos in the Github Repository

Each time was measured from the instant that the runner entered the direct line of sight of the seeker until the start of its movement.

5.2 Accuracy of target assessing

To test the seeker's accuracy, a series of objects approached the seeker from different angles and speeds and here are some results:

Table 2: Results of the accuracy experiment

Trial	Runner's Speed	Approach Direction	% of accurate response
1	10	Right	80
2	10	Left	80
3	20	Right	60
4	20	Left	50
5	70	Right	30
6	70	Left	20

5.3 Discussion of results

From the reaction time experiment, it could be seen that the robot does not react to objects that move at speeds faster than 70[4] and does not answer accurately neither for objects moving at those speeds. How the other hand, for low speeds, the robot's algorithm is working within margin of error.

This would suggest that the solution implemented is not enough for more demanding environments, which might very likely be due to the points discussed in subsection 3.3. If those were to be tackled in a future implementation of this algorithm, its plausible that the results would be much better.

Regardless, the results were enough to proof that the concept of ultrasonic sensor fusion for object tracking has some potential. At the current implementation it works for low speeds decently[1].

6 CONCLUSIONS

In conclusion, the purpose of this paper was to assess whether ultrasonic sensor fusion for tracking objects would be viable. The method adopted showed some potential and interesting results, however it proved insufficient in its current state for fast moving targets. The reaction speed could be increased by increasing the delay between measurements but at the cost the accuracy, as explained in subsection 5.1.

Overall the research was successful at demonstrating a proof of concept and if the issues referred in subsection 3.3 are tackled, the algorithm will work much better and at a much lower price than a more sophisticated sensor such as a RGBD camera or LiDar.

The final project code is accesible on *Github*.

[2]Each trial consisted of 3 measures and the written reaction time is the mean average of the measurements for each speed

[3]The time was measured with by recording at the 60fps on a DLSR camera and then analyzed in Adobe Premiere. Still,this results are merely approximate and meant to be taken as qualitative

[4]This speed is the integer given to the servos of the robot, it's a function prebuilt in the library as the author had no way of accurately measuring the runners speed

REFERENCES

[1] Z. Chang, H. Wu, and C. Li. 2023. YOLOv4-tiny-based robust RGB-D SLAM approach with point and surface feature fusion in complex indoor environments. *Journal of Field Robotics* 40, 3 (2023), 521–534. www.scopus.com

[2] A.I. Comport, E. Marchand, and F. Chaumette. 2004. Robust model-based tracking for robot vision. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 1. 692–697 vol.1.

[3] P. Gu and Z. Meng. 2023. S-VIO: Exploiting Structural Constraints for RGB-D Visual Inertial Odometry. *IEEE Robotics and Automation Letters* 8, 6 (2023), 3542–3549. www.scopus.com

[4] Surachai Panich. 2010. Comparison of distance measurement between stereo vision and ultrasonic sensor. *Journal of Computer Science* 6, 10 (2010), 1108.

[5] Zou Yi, Ho Yeong Khing, Chua Chin Seng, and Zhou Xiao Wei. 2000. Multi-ultrasonic sensor fusion for mobile robots. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511)*. 387–391.

[6] V A Zhmud, N O Kondratiev, K A Kuznetsov, V G Trubin, and L V Dimitrov. 2018. Application of ultrasonic sensor for measuring distances in robotics. *Journal of Physics: Conference Series* 1015 (may 2018).