Atlas Robotics
Internship Report

Can Ersoz

21.08.2019

Preface and Acknowledgements

I worked as an intern at Atlas Robotics between June 24th and July 26th. Under the supervision of Dr. Çetin and Tekin Meriçli, I worked alongside Burak Eruçar on Atlas's Artificial Intelligence Guided vehicle system. My major task was to integrate a new sensor to the production robot and then optimise it.

The internship enhanced my interest in computer science and robotics and helped me expand my Python and ROS skills. I am sure that I will work on and further develop these skills throughout my high school career and beyond. Most importantly, this internship gave me the chance to experience what it was like to work in a CS/robotics startup environment, a path that I would love to pursue in the future. The time Burak spent with me to explain the hardware/software systems of the robot, their future business plan and concepts such as localisation and route planning was one of the highlights of my experience here.

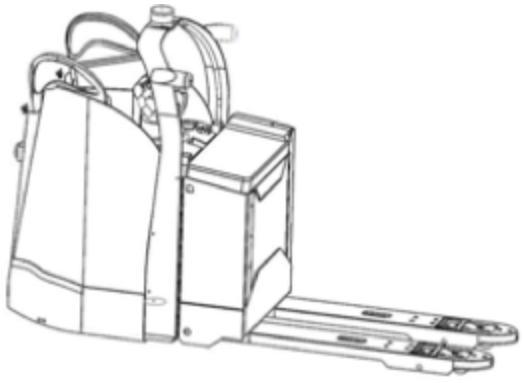
I would like to thank Dr Çetin Meriçli for giving me the opportunity to work at Atlas Robotics and making this wonderful four weeks possible. I am truly grateful to Burak Eruçar for being more than willing to help me at any moment, supporting me at any step. I would also like to thank all of the other employees at Atlas for offering me a very friendly environment.

Executive Summary

Atlas is a robotics company working on Artificial Intelligence Guided Vehiclers (AIGV) for autonomous use in factories. In this internship, I worked on Atlas's robotic transpalette system to integrate a new set of sensors to the blindspots of the robot for avoiding possible collisions with objects or people close to the robot.

I tested three sensors and selected the one which works best for the given task. The final system I created consisted of an Arduino connected to multiple sensors which reads the sensor data and publishes it to individual ROS topics using rosserial; and another node which reads this data and applies various optimisations for more accurate results.

The Atlas Transpalette AIGV



Atlas is a robotics startup aiming to produce and market special products utilizing the latest autonomous vehicle technologies. Their project, Atlas AIGV™, is an autonomous transpalette aiming to replace human operators in the chaotic factory environment to make logistic operations faster and safer. Atlas produces Artificial Intelligence Guided Vehicles (AIGV™) to provide

the flexibility and safety that dynamic environments where humans and robots work together require. Using its state-of-the-art perception technology, Atlas AIGV™ localizes very precisely in the environment, recognizes pallets, humans, and other vehicles without using any fiducials.

The Sensor Integration Project

Atlas Robotics's transpallete mainly uses a LIDAR and depth cameras for autonomous navigation and mapping. However; these leave blind spots close to the robot and the ground. Therefore, proximity sensors have to be added to certain spots on the robot to remove these blindspots and avoid possible collisions with lower objects or people.

The objective of my project was to select a new set of sensors to be used in the robot and to create a system that would make the sensor reading available to Atlas's ros system. My project was expected to work with multiple sensors and to read accurately in very close proximity (below 15cm) since the aim of the project was to detect objects closer to the robot. The sensors also had to have a relatively high maximum distance of 4-6m since they would be used in larger factories. The project had four main components:

- Sensor selection
- Circuit design
- ROS node development
- Sensor optimisation

Sensor Selection

The first challenge was to select the optimal sensor for the task. The sensor had to have a relatively high maximum distance (4-6m) for use in spacious factory environments and a low minimum range (10 - 15cm) to avoid hitting objects close to the robot. The sensor also had to be reliable and have a fast refresh rate because of its crucial use. Apart from online research, I tested three sensors: LV-MaxSonar-EZ1, Sharp GP2Y0D815Z0F, JSN-sr04t.

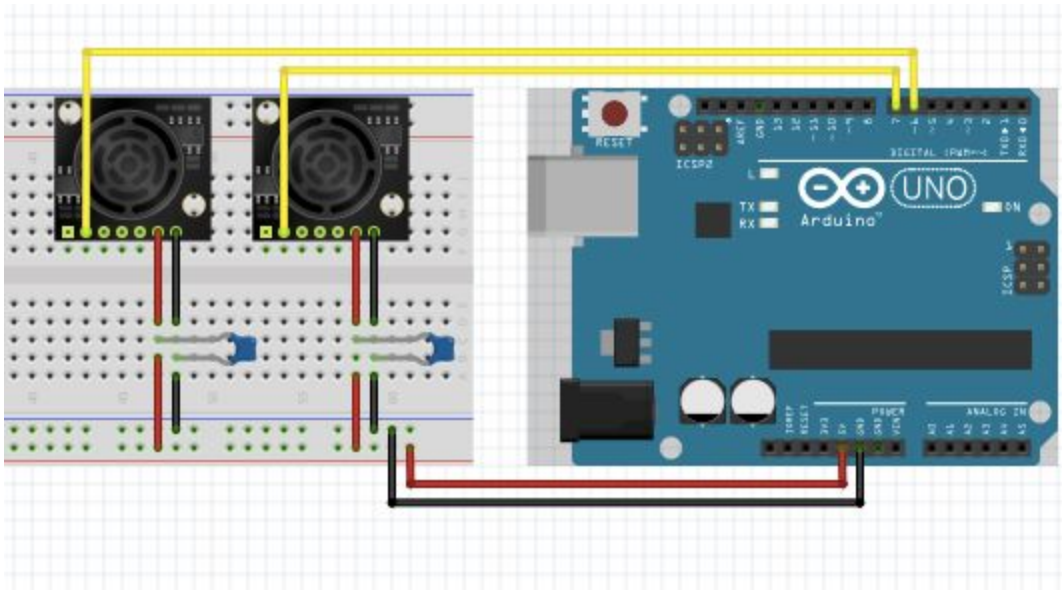
	JSN-sr04t	LV-MaxSonar-EZ1	Sharp GP2Y0D815Z0F
Type	Ultrasonic	Ultrasonic	Infrared
Range	25-450cm	15-645	0-15
Pros	-Low price -Waterproof	-Accurate readings between range	-Sensor returns true when an object is below 15cm and false otherwise. Therefore very accurate.

Cons	-Very unreliable readings: the sensor often paused for a couple of moments and then returned 0	-Higher price -Fluctuating readings below 15cm	-Doesn't read specific distance -Infrared sensors are unreliable for shiny, reflective objects.

After testing each sensor I decided that LV-MaxSonar-EZ1 was the best choice due to its range of 15 to 645cm, its higher accuracy and reliability compared to other sensor, 20Hz refresh rate and many other optimisation features.

Circuit Design

You can find the schematics for the circuit using two sensors below.



ROS Node Development

The software of the project consists of 2 ROS nodes. The first one is the the Arduino which acts as a ros node using the rosserial protocol and rosserial_ardunio library. The Arduino reads the data from the sensors and publishes it to individual sensorData topics. The second node, sensorOptimise subscribes to this topic and publishes new data with various optimisations to the optimisedData topic. Both of these nodes can be launched at the same time using a launch file.

You can access my code from the link below.
https://github.com/Canersoz02/Atlas-Sensor

Sensor Optimisation

I tried many techniques to optimise the sensor data and added each of them as a method in the sensorOptimise.py file. The user can select any one they prefer. All of the techniques gather an array of 5 sensor readings and optimizes them. This optimisation is especially crucial considering that the sensor readings fluctuate a lot when an object is closer than 15cm.

avg()
The average method simply returns the average of the given array. Although it is good at estimating the distance when the data points are close to each other, it isn't very reliable when the object is too close to the sensor since the sensor gives very inconsistent readings below 15cm. ([12, 640, 30, 500, 23] for example)

stdDev()
This method tries to determine when the object is closer than 15cm by looking at the standard deviation of the given array. Since the values fluctuate a lot when the object is too close, the method returns 0 when the standard deviation is more than 4. This method also optimises data sets that do not fluctuate by using the lowest() method. Although stDev() works well in guessing when the object is below threshold value, it also identifies the robot's nonuniform movements as being below 15 cm since the data also fluctuates. Therefore, this method is not very suitable for real life applications.

lowest()
Despite its simplicity lowest() in my opinion is the most suitable sensor optimisation method for real life applications. It simply takes an array and returns the lowest value in it. I preferred calculating the lowest instead of the median since the sensor never returns a value that is closer than the object really is. Therefore, the smallest value is nearly always the correct one. The only shortcoming of this method is when all of the measurements in the set is wrong.(such as [30, 48, 34, 500] when the real distance was 12.) stdDev can be used in this case however it has other problems as discussed above. Because there will be many sensors on the robot, we can assume that with high probability, one of them will guess the correct value below 15cm.

inRange()
This method simply returns true if one of the measurements is below 20cm.

Progress and Timeline

Week 1

I worked on installing and working rosserial. By the end of the week, I was able to read data from a sensor and publish it to a ros topic. Then, I worked on the turtlebot simulation to gain more knowledge on ROS. I wrote a node which controlled the turtlebot's movements based on the sensor readings.

Week 2

I worked on the optimisation of the sensor data and tried using the jSN-sr04 park sensor. I wrote most of the sensorOptimise.py file this week. I also tried to visualize the sensor data through rviz but I wasn't successful.

Week 3

I couldn't work much this week due to problems with electricity and internet however, I improved the optimisation code, added more methods and decided on using the MaxSonar sensor.

Week 4

I worked on adding multiple sensors to the system, wrote the launch file, wrote a library for easier sensor initialization and data gathering. Finally, I documented my work and added everything to a GitHub repository.

Areas for Improvement

As stated above, the lowest() method cannot return the right value when all of the values in the set are highly inaccurate. The sensor could be switched with a lower minimum range or two sensors could be used together (the Maxsonar and sharp for example) one for distance readings and one for warning below 15cm. Perhaps, another optimisation technique could be utilized that I haven't thought of.

I tried minor upgrades to the program such as a library for initializing a sensor set with a given sensor number or taking the optimisation method as a parameter of the ros node however I couldn't finish in time. Minor adjustments like this could improve the user experience. The maxbotix sensors also have a wiring system between sensors to avoid any interference that can happen between the ultrasonic sensors. For more accurate results, one could apply that wiring from the link:. https://www.maxbotix.com/documents/LV-MaxSonar-EZ_Datasheet.pdf