ELIMIA: exploring a remote location with an autonomous Lego robot

Giulio Zhou
University of Edinburgh
s1775060@ed.ac.uk

***Abstract*— This report proposes a prototype of an autonomous Lego robot capable of exploring a predefined area containing obstacles. The robot is able to navigate through the map avoiding them, detecting special marks on the ground (Points of Interest) and virtually transmitting information to a satellite by aligning the robot's antenna to it. To accomplish the above, the robot uses a number of sensors: two IR sensors for the obstacle avoidance and localisation, two light sensors and a camera for detecting the Points of Interest and two light sensors for measuring the wheels speeds. Due to the high level of inaccuracy, some of the methods proposed in this report were not actually used to complete the task. The robot was tested 10 times and it was able to detect, on average, 2 POIs. However, it was incapable of aligning its antenna on the second and third POIs. The main issue of the robot is that it could not model the drift properly which resulted in the divergence from the computed route and the drop of the accuracy for the localisation performance.**

I. INTRODUCTION

The main objective of this project was to build and program an autonomous robot able to detect specific points of interest (POIs) and virtually transmit data to a satellite. Thus, the robot should be capable of navigating within a defined area avoiding any obstacles, spot the POIs, compute its position and orientation and use this information to align its antenna to a satellite (a fixed position in the sky).
To achieve the above, the robot is equipped with two IR sensors for the obstacle avoidance and localisation, two light sensors for detecting the POIs on the ground, two light sensors for calculating the actual wheels speed and a camera to spot the POIs and, eventually, calibrate the trajectory and position of the robot.
The development process was divided into minor and major milestones. The focus of the first major milestone was to provide basic functions to the robot, allowing it to move while avoiding obstacles, as well as for detecting POIs on the ground and, having all the information regarding the position, align the antenna. The final milestone consisted in the integration of all the different software components and the implementation of a localisation system which allows the robot to complete all the tasks independently.
During the final test and presentation of the project, the robot was able to detect only one of the three POIs in the arena. The original software implementation differs quite heavily from the actually used code. In fact, the latter is a simplified version of the originally planned one. The grid localisation was not used, instead, the robot uses an odometry-based localisation which relies on the accuracy of the movement to retrieve its position. The speed measurements through light sensor readings and the POI detection with the camera were also discarded due to their low level of accuracy.
To navigate the robot a strict number of tracks which cover all the main area of the arena and the most probable zone in which the POIs can be positioned.
The team developed the different software components separately and tried to integrate them once they were completed. However, this caused several issues to the project.

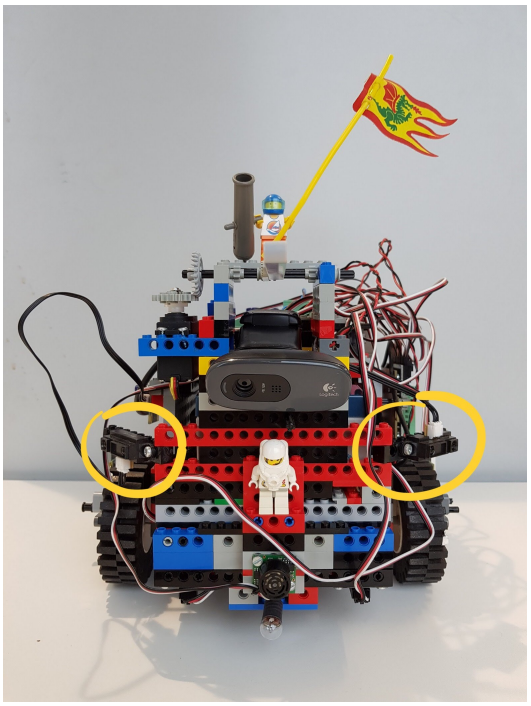

Fig. 1: Front, IR sensors highlighted

II. METHOD

A. Robot architecture

Figure 1 shows the final version of the robot.
The design of the robot is meant to be as compact as possible but at the same time robust. The length of the robot is 30cm (not considering the antenna), the width (wheel to wheel) is 23 cm and the hight (ground to antenna base) is 21 cm.
The robot is composed by two IR sensors, four light sensors, a camera, a servo motor, a hall effect sensor and a light bulb, in addition to the Fit-Pc2, the Phidget Interface kit, the power, servo and motor boards, and a battery.
The main body of the robot is 13 cm width. The IR sensors are positioned on the front of the robot, Figure 1 at 11 cm from the ground rotated respectively by 45°on the left and the

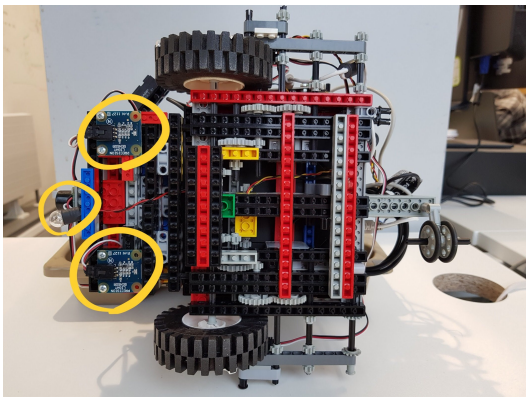

Fig. 2: Bottom, light sensors and bulb highlighted

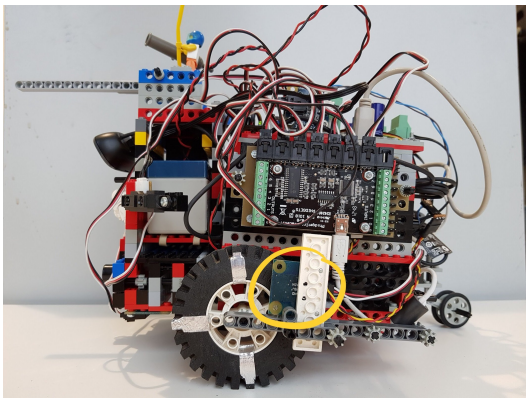

Fig. 3: Left side, light sensor highlighted

right. On the front, in the centre, it is positioned the camera at 16 cm from the ground, rotated by 45°down. Two of the light sensors and the light bulb are positioned at the bottom of the the robot, Figure 2, at 4 cm from the ground. The other two light sensors are put in front of the main wheels at a distance of 0.8 cm from it, Figure 3.
The wheels are attached to the motors with a ratio of 1:13.5, Figure 4, and the hall effect sensors is attached to the right wheel gears with a ratio with the wheel of 4.5:1.
The antenna is attached to the servo motor with a 1:1 ratio.

B. *Control flow*

To complete the task the robot uses a hybrid approach to the problem, which means that the planning part takes place before the actual movement and sensor readings. Some information is available to the robot: the dimension of the arena, the obstacles' and antenna's positions, the starting position of the robot and roughly the area where the POIs are. Given this information, the robot chooses a destination and select the route to follow. During the actual navigation, the robot uses its sensors to detect obstacles and eventual POIs (which exact position is unknown) while updating its position. Figure 5 shows the processes and decisions made by the robot during the execution of the task.

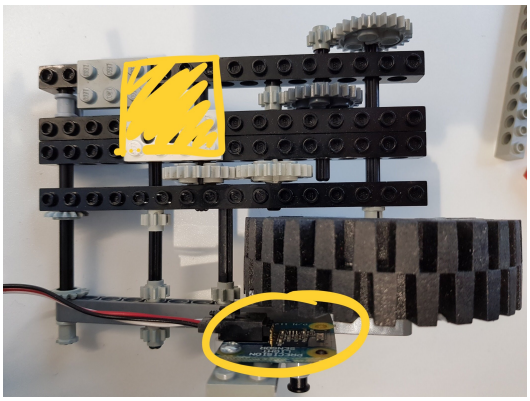

Fig. 4: Right wheel with light sensor, gears and motor position

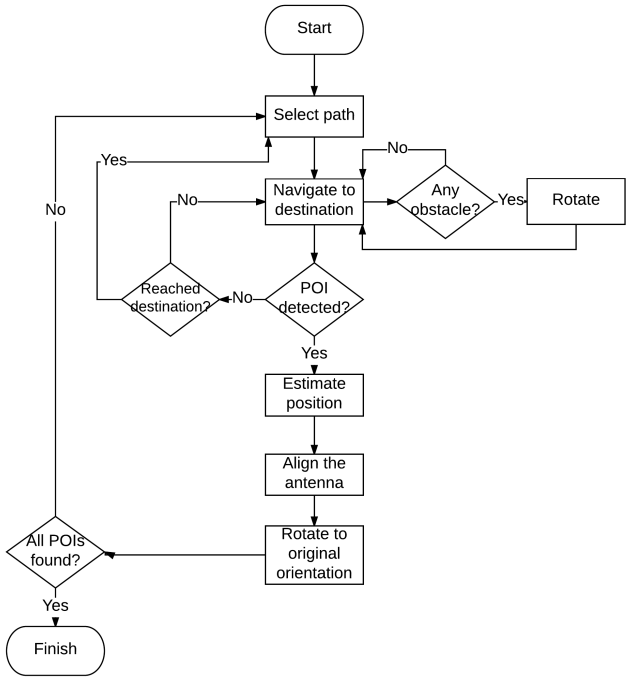

Fig. 5: Control loop

C. *Routing*

Having different types of information at our disposal, it is possible to calculate the directions from one point to another. For this reason, the routing problem was reduced to a "shortest path problem". The map of the arena can be represented by a graph where each free space piece is connected with the ones adjacent to it. Independently by the algorithm used, the computational cost is high due to the high-dimensionality of the graph. It is possible to reduce the number of nodes in a proportionally way, but still, the computation is not fast enough and it reduces the accuracy of the model.
To overcome this problem, the model created allows the robot to navigate through a very limited number of paths. The graph built consists of 10 nodes and 10 arcs as shown in

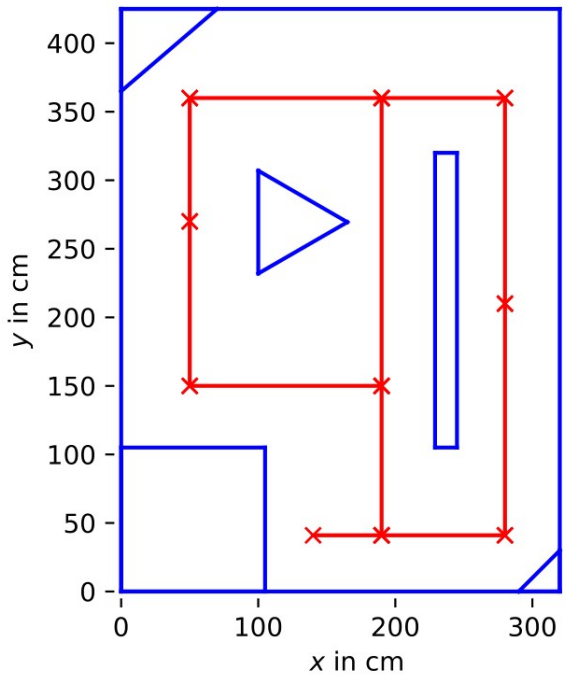

Fig. 6: Map of the arena with waypoints and predefined tracks

Figure 6. Although not all the spots in the arena are covered, it provides a reasonable approximation of the paths the robot can take.
The robot's routing system is hence based on a number of waypoints through which it has to cover in order to reach its destination. If its position is not on a track, it reaches the closest waypoint, it calculates the shortest path from one waypoint to another using the Dijkstra's algorithm, and finally uses the computed directions to the destination.

D. Movements accuracy and odometry

The accuracy of the movements plays a fundamental role in the navigation process. Having decided the track to follow is not enough to determine whether the robot will be able to reach the predefined destination. In fact, it is necessary to address two major problems: distance travelled and drift.
The movements of the robot are based on time making it hard to determine the absolute distance travelled by the robot within a certain time interval.
The solution proposed is to calculate the real speed of the wheel. To achieve this, light sensors were used as rotation sensors by detecting the reflective tapes attached to the wheels, Figure 3. The distance travelled is given by $s = 4.1(90/dt)t$ where dt is the time necessary to the wheel to rotate by 90°, 4.1 is the radius of the wheel and t is the time travelled. In addition to the light sensors, the hall effect sensor estimates the distance travelled by counting revolutions, each of which corresponds to 5.72 cm travelled. On the other hand, despite the fact that the power provided to the motors is the same, the actual velocity of the two wheels do not rotate at the same velocity resulting in the robot not able to drive perfectly straight. Probably causes are the robot construction (e.g. it's not balanced) or little differences in the motors themselves. To straighten the trajectory of the robot it is necessary to calibrate one of the wheel velocity, Figure 7. The adjustment amount is calculated by a proportional controller $p = K_p \log(|error| + 1)$, where K_p is a fixed constant. The error is not directly multiplied in order to model better the non-linearity of the relation between motor power and wheel speed.
Unless having perfect light conditions, the light sensors can fail to detect the reflecting tape on the wheel, for this reason, the robot discharge unreasonable error values through a threshold.

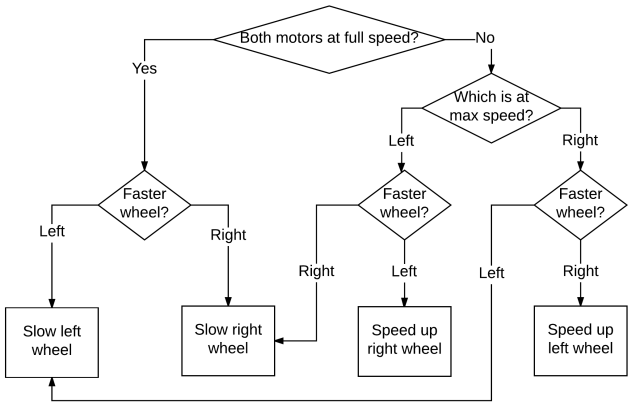

Fig. 7: Decision tree to choose which wheel to adjust

E. Localisation

The first method to determine the position of the robot was a grid-based approach. For this purpose, the arena was divided into 5x5cm^2 grids each of which associated with a probability, resulting in a map similar to Figure 6. The grid is initialised with 0 probability except for the starting grid which is set to 1. At each movement step, all the probabilities are shifted towards the direction of the robot, it then retrieves the IR measurements to detect the surrounding objects and updates the 3x3 around the robot's estimated position. The probability of each grid corresponds to the following joint probability $P(x|z_l, z_r)$ where z_l, z_r are the left and right sensor readings, which is solved applying Bayes Rule. The result is then a combination of the $P(z_{l,r|x})$, which is a Gaussian and where the variable is the difference between the actual sensor reading and the raycast for a given grid, and $P(x)$, the prior probability of the grid.
The second approach is a localisation based on odometry. The position estimate is then corrected with measurements provided by the IR sensors.

F. POI detection

The main system used to detect the POIs is the combination of light sensor and light bulb positioned. The robot

calculates the average value detected by the two sensors and keeps updating it throughout the entire task execution. When a sensor detects a drastic increase in the sensor reading, a POI is found.
As a support to the light sensors, the camera attempts to spot a POI. The image captured by the camera is preprocessed by converting the RGB image into HSV, followed by the application of the Otsu thresholding [1] in order to segment the image into different regions: walls, ground and eventual POI. Lastly, the region on top of the image (which represent the wall) is filled leaving only the POI highlighted if present. If a POI is detected by the camera, the robot computes also the distance necessary to reach the POI, this information is retrieved by converting the image coordinates of the POI into physical ones using CV2.GETPERSPECTIVETRANSFORM.

G. Obstacle avoidance

The IR sensors are able to detect objects in a range between 10cm and 80 cm. When one of the sensors detects an object within 20cm distance, the robot turns by 20°in the opposite direction, and repeat until the path is free. If both sensors detect an obstacle, the robot goes backwards by 15cm and turn left by 40°.

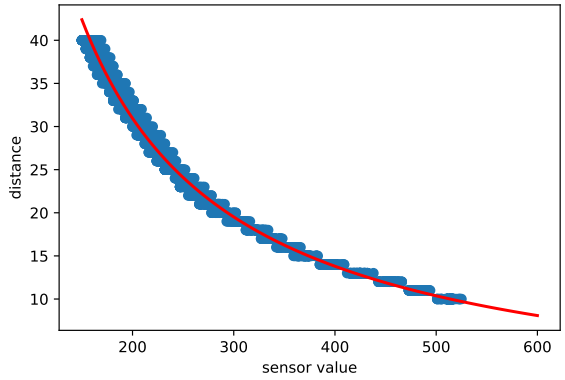

Fig. 8: Function to compute the distance given the IR readings

H. Antenna alignment

The alignment of the antenna consists of two movements, turn the robot and use the servo motor to point the antenna towards the satellite. Both rotation degrees are computed through simple trigonometry equations. Given the position and orientation of the robot and the coordinates of the satellite, the robot calculates the distance between itself and the satellite and uses the formula $\theta = arctan2(\|\mathbf{v_1}\|\|\mathbf{v_2}\|, \mathbf{v_1} \cdot \mathbf{v_2})$ to retrieve the angle.

III. RESULTS

A. Drift

Without any obstacles, with a full battery and both motors at full speed, the robot drifts averagely 30cm each meter. To improve the performance of the movement, it is possible to reduce manually the power of the fastest wheel, which is, in our final design, the left one. However, the tests have shown that it is impossible to define the difference in power necessary to the wheels to move at the same speed a priori. More than 95% of the times, the robot kept drifting. We speculate that the there are variables that depend on the motors themselves.

B. Speed estimation

The light sensors for the speed estimation has been tested under different conditions:
-One wheel in dark environment: on average, 80% of the time the speed is not updated due to the error threshold. The rest of the time the system breaks, what happens is that the wheel under good light keep slowing down until it stops;
-On normal condition with initial manual straightening: the robot goes straight for around 2 m but then one of the wheel's speed begins to oscillate. The oscillation introduces some drifts and although it tries to keep the wheels at the same speed, it cannot recover from the error accumulated. In addition, on an average of 1 test out of 5, the system breaks with the same behaviour as above. Overall, the robot can drive 4m with drifting 30-40 cm.

C. Localisation

The grid localisation has been tested with different map resolutions, however, none of these provided satisfying results. The main problem found is the tradeoff between accuracy and computational cost. The lower the resolution of the map the less is the model accurate. As a consequence, it is impossible to determine neither the position nor the orientation. On the other hand, reducing the grid size provides a reasonable localisation of the robot thanks to the high accuracy of the raycast. The tests have shown that a map with 5x5cm^2 grid is sufficient to determine the pose of the robot but on average it requires 5 sec to update, while an even smaller 1x1cm^2 grid requires more than 1 min.

The odometry-based localisation provides instead an adequate estimation of the pose of the robot and does not require any additional computational cost. On average, after one minute of testing in the arena, the estimated position was off by 35 cm from the real one.

D. POI detection

The light sensors for detecting the POIs succeded in all tests.
The use of the camera for detecting the POI did not perform as hoped. Although it can detect effectively the POI in the image if present, the number of false positive is so high that the overall accuracy of dropped under 50% makes it impossible to use it in the task. The false positives are given by the similarity of some pattern on the ground to the POI (e.g. shadows).

E. Complete test

Based on the previously showed results, the grid localisation, the camera, the speed measurement system and the control loop feedback for the motors (which depends entirely by the measured speed) were not used.
Starting from the home position and an orientation of 90°, the main task was tested 10 times. The robot was able to detect the first POI and align the antenna to the satellite 9 times. Nonetheless, it was able to reach the second and third POIs only 6 and 4 times respectively and to align the antenna just once at the second POI and never at the last one.

IV. DISCUSSION

Despite the fact that the robot was able to complete the task once and, on average, was able to detect two POIs at each test, we can state that project failed. Some of the originally implemented major software components were discarded due to their high inaccuracy. For this reason, although the robot could perform perfectly the POI detection on the ground, the alignment of the antenna given the pose, the obstacles avoidance and the computing of the route, the rest of code was not sufficient to the robot to complete the task.
To sum up, the core problem of the robot, which we were not able to solve, is the drift, in fact, if the robot could move precisely, the localisation would've improved significantly permitting the robot to detect the POIs which were on the predefined tracks and align correctly the antenna.

A. Improvements

1) Robot design: The design did not work well as expected at the beginning of the project. We believe that the 3-wheel design contributed significantly to the drifting problem. Furthermore, the design did not allow an easy maintenance of the hardware components and the robot was not well balanced. A 4-wheel design would have allowed more accurate movements.

2) Speed measurements: Measuring the actual speed of the wheels was a good idea. The actual code for this task is working however it could not provide reliable information due to the inaccuracy of the sensors readings. In fact, the major con was that the light sensors needed an optimal lighting to generate the correct values. The design with light sensor was chosen to avoid the rebuild of the entire robot since it was a feature introduced in the latest stage of the development.
More reliable designs to retrieve the rotation speed of the wheel could be one that allows the sensors to have only light/no light readings or that uses buttons instead of the light sensors.

3) Speed sync: For this project, a simplified version of a PID control was implemented for the synchronisation of the wheel speed. The error in the implemented version could not reach 0, for this reason adding the Integral component would improve the performance of the controller. In addition, the proportional constant K_p could have been tuned more.

4) Localisation: To enhance the accuracy of the grid localisation, more sensor readings should have been used. The main problem was that many grids did not "die" as fast as expected, this resulted in the spreading of grids with high probability making hard to determine the most likely position. Combining more sensor data would have cut off the most unlikely grids.
The odometry-based localisation worked decently, it did not work perfectly because of the drift.

5) POI camera detection: For the objective of this project, the POI detection through vision wasn't necessary. Nevertheless, it could have improved the efficiency of the robot by adjusting the trajectory when it detects a POI. The amount of false positive has to be reduced drastically, a possible improvement could be a dimension threshold to cut off all the "POIs" too small, or introducing a more accurate shape analyser.

REFERENCES

[1] Vala, Miss Hetal J., and Astha Baxi. "A review on Otsu image segmentation algorithm." International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 2.2 (2013): pp-387.