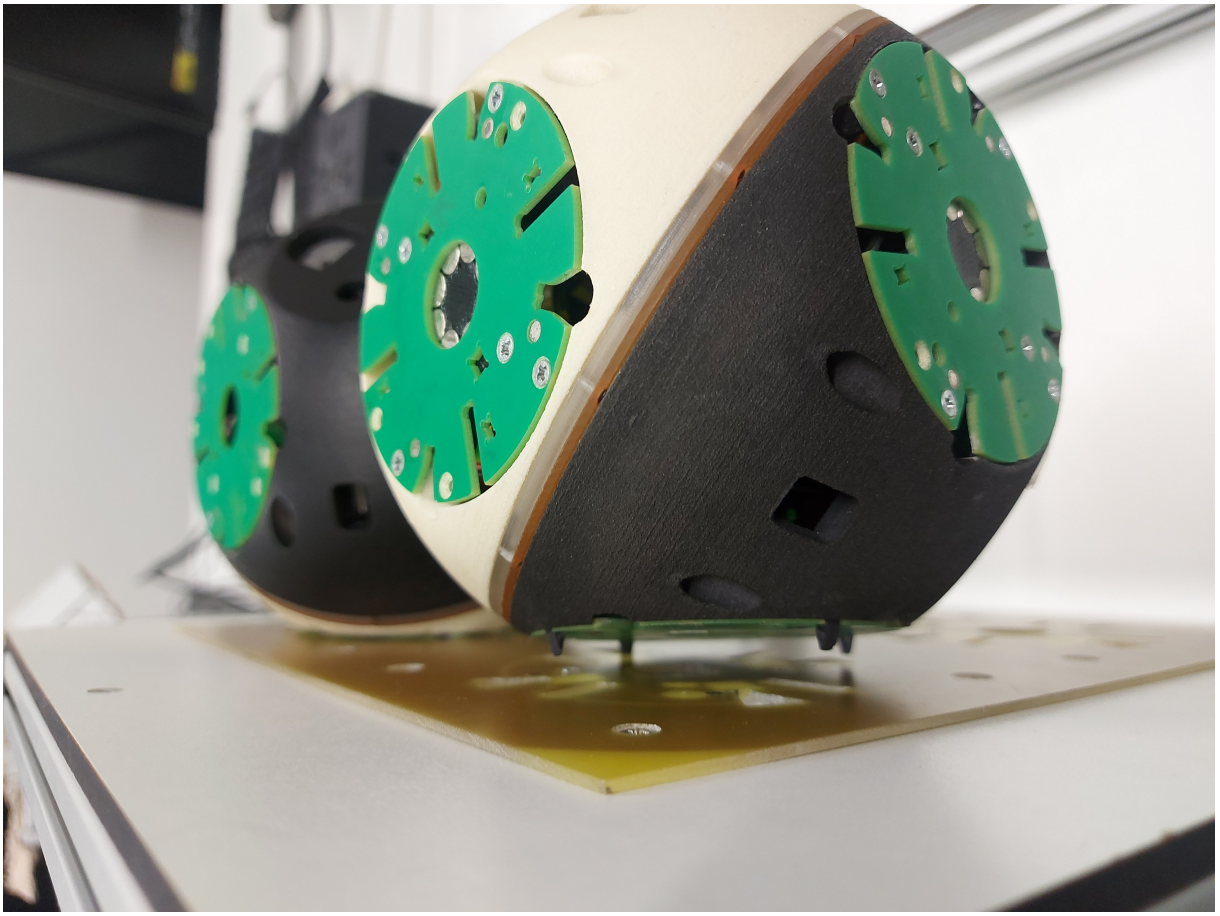

Improving Roombot's docking procedure

Fall semester project - Biorobotics Laboratory (BioRob)

EPFL

Alexandre de Terrasson de Montleau - 288111

Contents

1 Introduction

1.1 Problem exposure

Roombots are self-reconfigurable modular robots developed in the BioRob laboratory (Pr. Ijspeert). Their basic functionality is to connect and disconnect to other modules or to passive elements. This is achieved with a mechanism called Active Connection Mechanism (ACM). For a more extensive description of the Roombots' functionalities, we refer the reader to the following paper: [2].
ACM contains sensors (Hall-Effect, IR, IMU) and rotating latches that grip a passive connection plate: in consequence, a perfect alignment between the two connecting faces is required. For a new attachment between modules, robustness and speed of connection is crucial, but challenging due to the difficulty to ensure the alignment mainly due to gravitational effects and motor backlash (Figure 1). An example of a Roombot in a docking procedure can be seen on the front page image.
To solve the alignment issue, a first solution (described in the next part) was developed in the form of a plugin by H.Khodr during her Master thesis [1]. the baseline solution guarantees the robustness of alignment but not a significant alignment speed. The goal of the project is to find a new version of the plugin that improves the baseline solution in terms of speed of alignment.

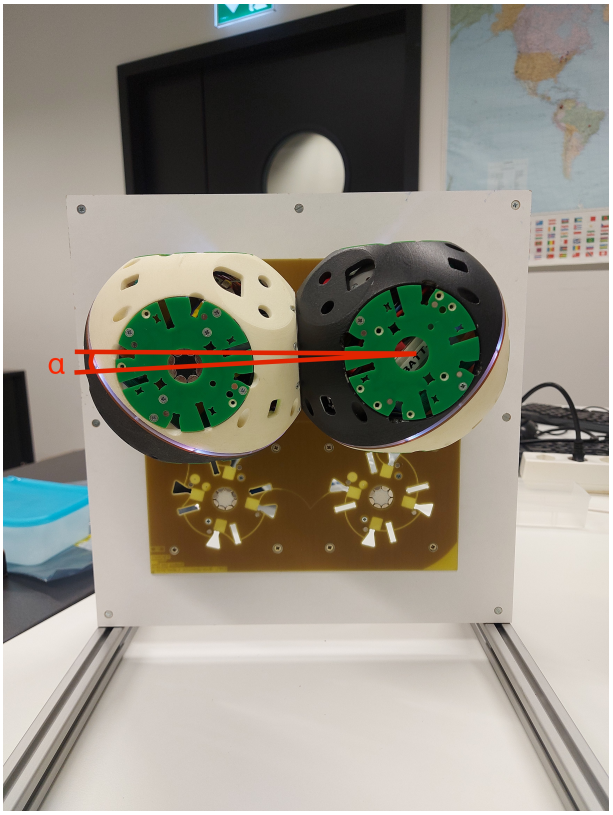

Figure 1: Example of the effects of gravitation and motor backlash on the Roombot creating and angle α between the first and second part of the module, resulting in a misalignment of one of the Roombot's ACM with the passive connection plate

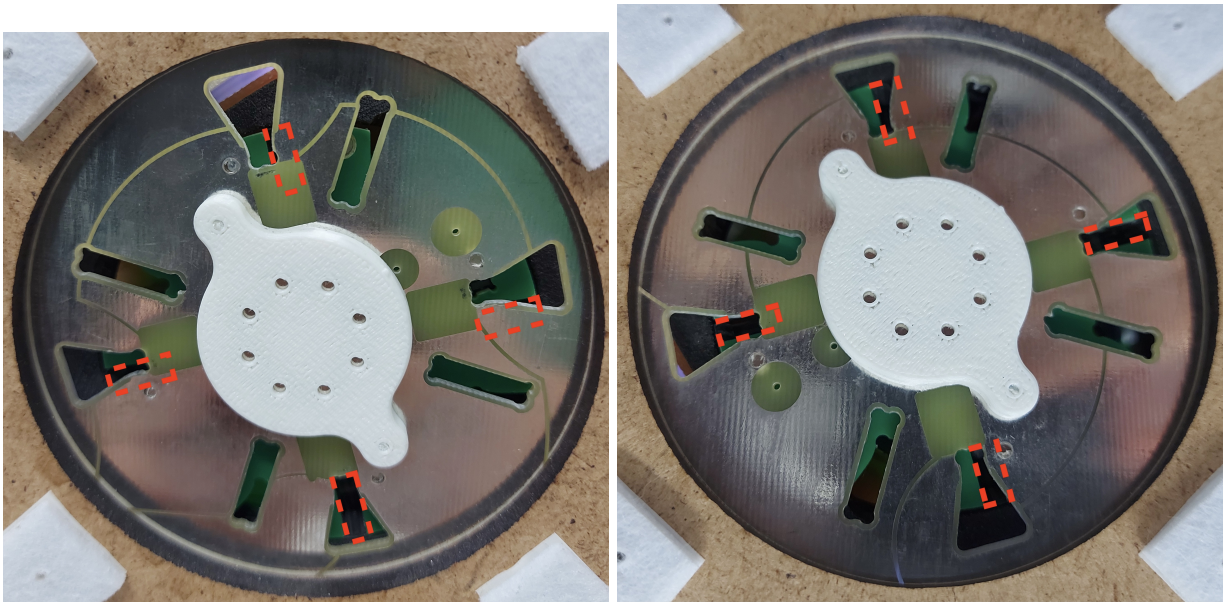

Figure 2: Two examples on the same case of misalignment of the ACM with the passive plate. Position of the hooks are highlighted in red.

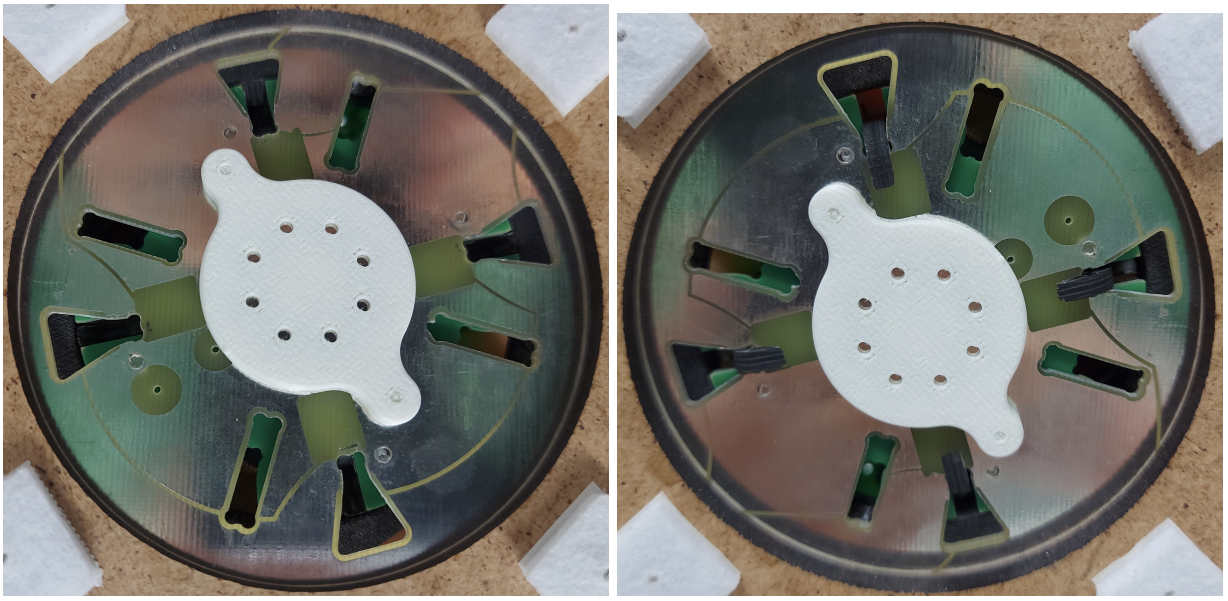

Figure 3: Example of alignment of the ACM with the passive plate. Left Figure: The hooks of the ACM are open and facing the holes. Right Figure: The hooks are closed and attached to the passive plate.

1.2 The baseline solution

After sending the motor command to the Roombot in order to reach the alignment position, due to gravitational effect and motor backlash, the ACM of the Roombot could not be aligned with the passive plate. If a misalignment is detected, the baseline plugin is triggered

The baseline plugin correspond to the first developed solution that ensures the alignment of the ACM with the passive connection plate (which allows a successful connection). The algorithm consists of a basic local random search as shown on figure 4. In this algorithm, at each step N (chosen to be 4), random angles neighbors of the current position of the 3 motors of the Roombot are chosen (circles on figure 4). Based of their fitness function (normalized sum of all four Hall effect sensors), the new current angles are those resulting in a position of the roombot with the lowest fitness function value. These steps are repeated until each hall effect sensors are lower than a certain threshold which indicates alignment of the ACM with the passive connection plate (red rectangular on figure 4). According to experimental results of the baseline solution [1], the success rate of a successful connection with the baseline plugin is 100% with a time limit of 5 minutes.

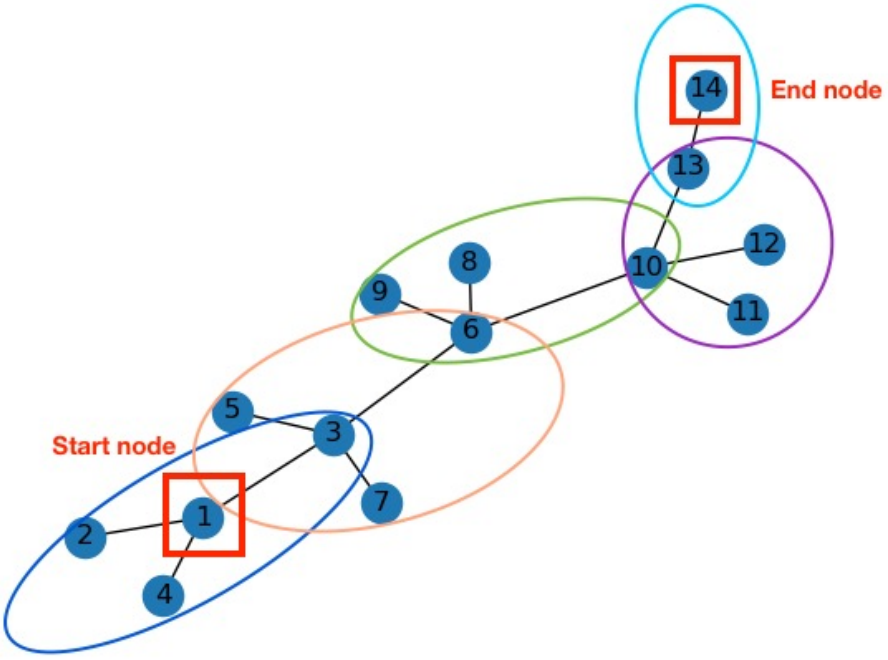

Figure 4: Graph representation of the random search algorithm of the baseline solution

1.3 Small reviews on existing docking procedures in the literature

According to the scientific literature, IR sensors are widely used for localisation of robots in general (ex: [5]). Closer to the Roombot, some self-reconfigurable robots use sensors fusion for their docking procedure (ex: [3], [5]). Infrared sensors are also used in multiple solutions [3], (ex: [4]). This results give us great directions to explore for the Roombot.

1.4 Definitions

Here are some short denominations we will use for the rest of this document:

For each four cases we will consider in this study, following names were assigned to them (Figures 5, 6, 7 and 8). These cases were chosen because they represent most of the alignment-wise problematic configurations the Roombot can meet.

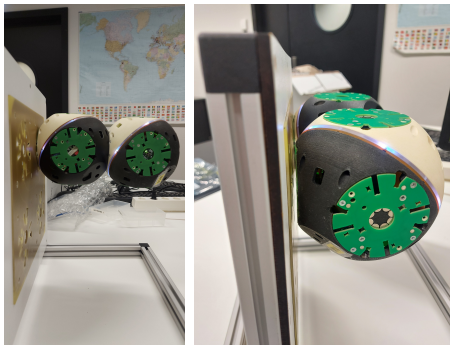

Figure 5: Case side to side (SS)

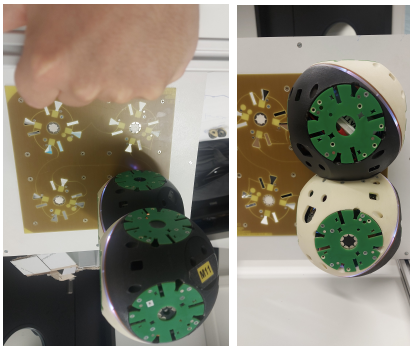

Figure 6: Case side to up (SU)

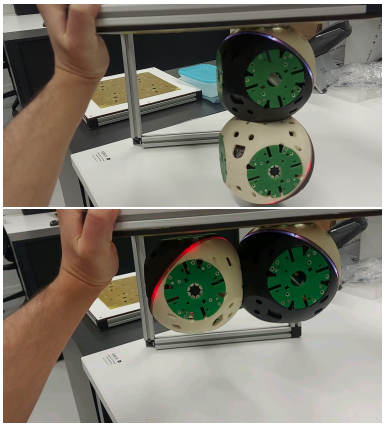

Figure 7: Case down to side (DS)

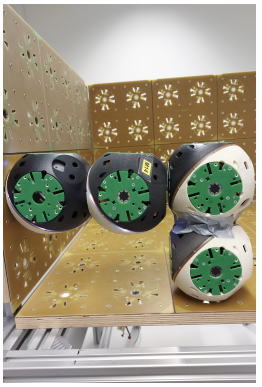

Figure 8: Case face to face (FF)

The Roombot is composed of three motors. Their values determines the position of the ACM in space. We assign to them a unique number: 0, 1 and 2 (figure 9).

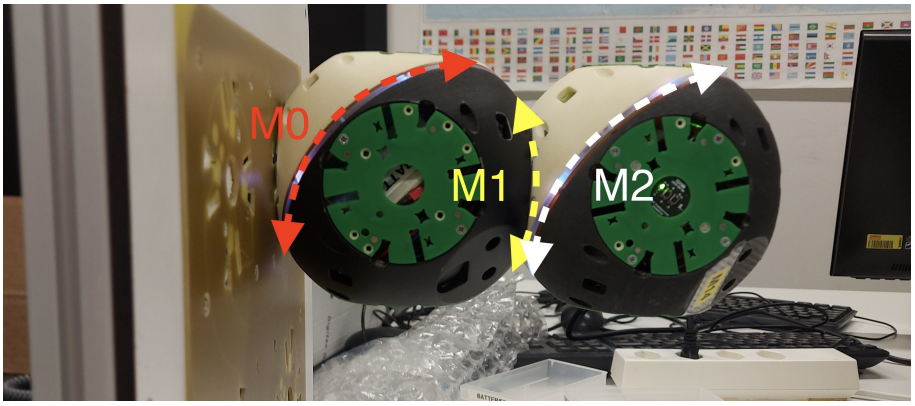

Figure 9: Side view of the Roombot and the direction of movement of each of its motors

2 First approach

In this section, we present the first approach which is focused on computing the right motor command to reach alignment.

2.1 Computation

After sending the motor command to reach the alignment position (the target position), due to effect discussed above, the Roombot is not at the target position, so it is not aligned. We explore the possibility to directly compute and send the correct motor command to the Roombot and reach alignment of the ACM with the passive plate. This would outperform the random search because it would take only one trial to reach alignment from the initial non aligned position.

To calculate the correct motor command, we follow three steps:

1. In simulation, using the unified Robot Description Format (URDF) of the Roombot and forward kinematics, we let the ACM end effector reach random positions from random choices of motor values. We calculate the angle β between the ACM frame of the reached position and the ACM frame of the target position (position where the Roombot should be aligned without impact of effects, so its corresponding ACM frame is parallel to the passive connection plate) using their respective homogeneous transformation matrix.

2. We only keep motor commands that result in a position with an angle β of maximum 0.5 degree. We choose 0.5 degree because we want the ACM plane of the Roombot to be almost parallel to the passive plate. These key motors commands result in key positions. Key positions in the neighborhood of the initial (non aligned) position are displayed on Figure 10.

3. The correct motor commands are those resulting in a key position that is the closest to the target position. We keep the corresponding motor commands, send it to the real Roombot and reach alignment.

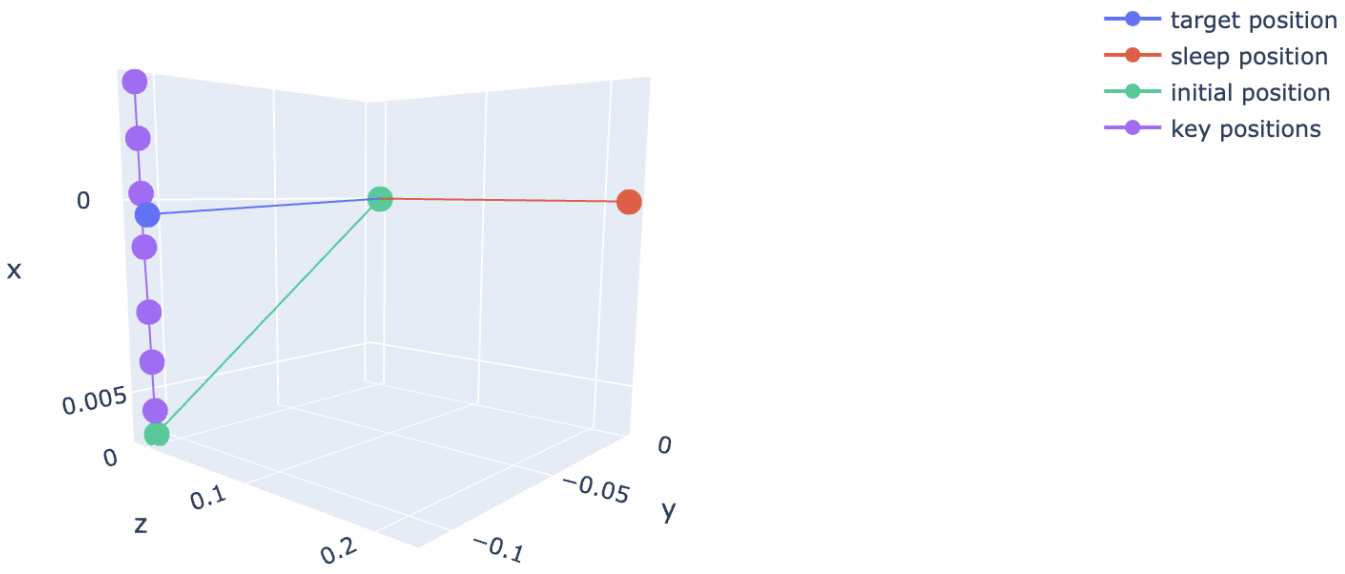

Figure 10: Plot of the measured initial position of the Roombot, its target position and the calculated key positions

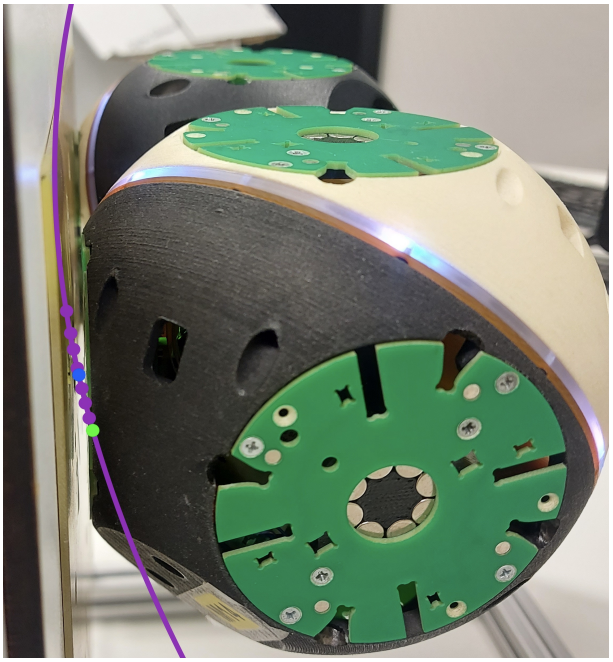

Figure 11: Corresponding real view of Figure 10 on a real Roombot placed on the initial position

2.2 Challenges regarding this approach

The main challenge in this approach is that we need to know the exact position of the Roombot and where it needs to be attached. We could know the position of the Roombot based on its motor command and the accelerometers values. Also we could get the direction of alignment using sensors by testing each directions, measuring their IR sensors values and choose the direction with the smallest IR sensors values. The direction with the smallest IR sensors values implies that the Roombot gets closer to the plate.
This approach is promising to solve the alignment problem for a list of identified connection configurations, as it would result in a simple look-up operation. However, it is not practically implementable with the current hardware as accelerometers are not reliable enough to identify such configurations. For that reason, and also because the number of configurations is not bounded when increasing the number of modules in the system, we decided to focus our efforts on the second approach presented below.

3 Second approach

In this section, we present the second approach which is focused on enhancing the baseline solution.

3.1 New fitness function

A first observation we can make from the baseline algorithm is that it doesn't use all available sensors. Indeed only Hall effect sensors are measured for the fitness function. Hall effect sensors deliver good alignment information but they quickly become useless when the distance between the magnets is important. They don't deliver distance information. However IR sensors are well suited for distance information but not alignment information. Our approach is to use sensor fusion. We decide to combine the alignment information (Hall effect sensors) and the distance information (IR sensors) in a new fitness function which is the normalized sum of Hall effect sensors values and IR sensors values:

$$f = \sum_{i=1}^{3} Hall(i) + \sum_{i=1}^{3} IR(i).$$

Our changes led to a significant increase of the alignment speed in all cases. Figures 12 and 13 display the result of the fitness function change on the two cases with the slowest alignment speed in the baseline solution. The improvement of alignment speed is significant for both examples because we can notice that in both cases, the upper quartile of the plugin with the new fitness function is lower than the lower quartile of the original plugin.
In all cases, the plugin with the new fitness function has a significant increase of speed alignment compared to the original plugin.

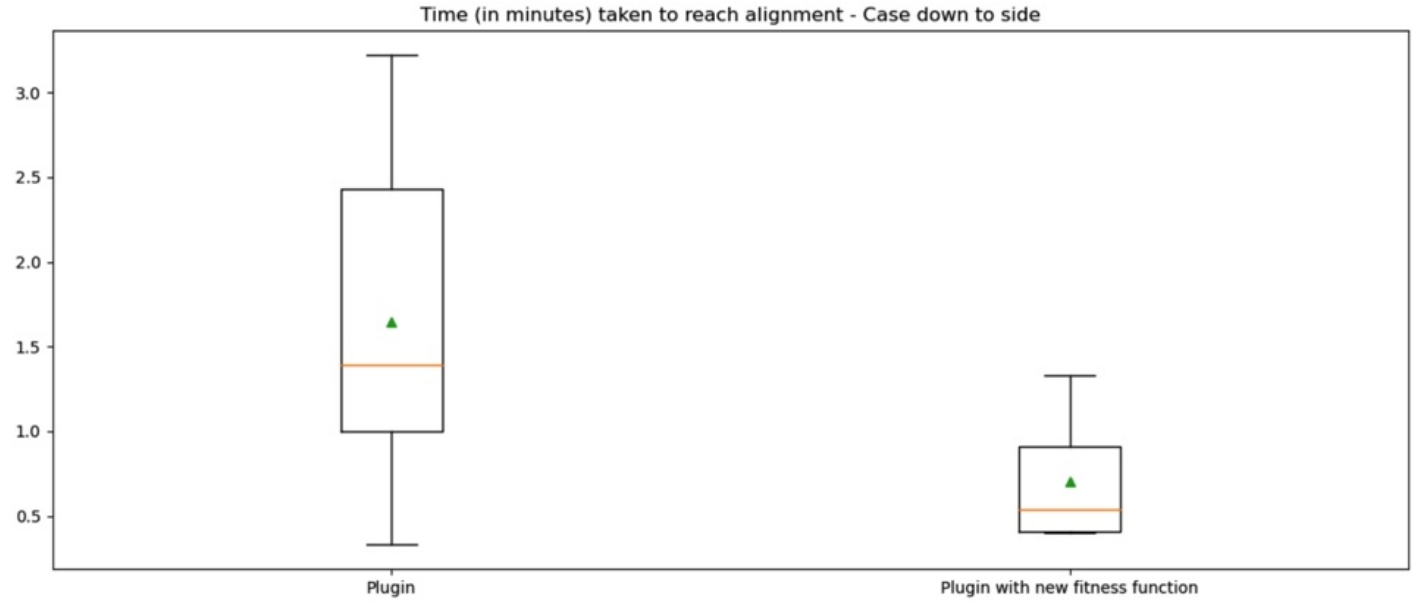

Figure 12: Box plots of the time (in minutes) taken to reach alignment for the baseline solution (left) and the baseline solution with the new fitness function (right) - Case down to side - 10 tries

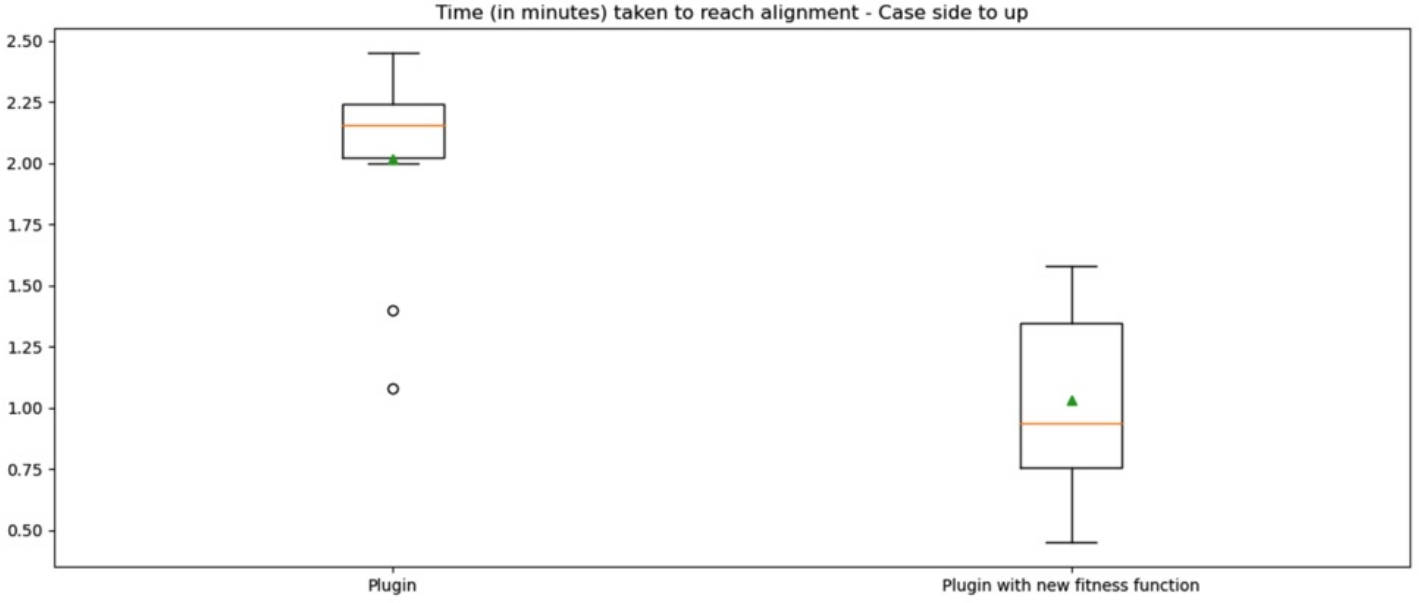

Figure 13: Box plots of the time (in minutes) taken to reach alignment for the baseline solution (left) and the baseline solution with the new fitness function (right) - Case side to up - 10 tries

3.2 Set a limit of the motor commands distance

Coming back to the baseline solution, the input of the plugin are the initial motor commands (motor commands supposed to put the Roombot on the alignment position without gravitational and backlash effects) and which ACM we want to attach on the passive plate. Because of the misalignment between the ACM and the passive connection plate, a random search is started around this initial position in the aim to find the position that guarantees alignment as we have discussed above. At each step during the random search, a random combination of the 3 Roombot's motors is chosen, which allows the Roombot to reach a new random position and measure the fitness function when the position is reached. Focusing on cases "down to side" and "side to up" (the two cases with the slowest alignment speed in the baseline algorithm), we calculate a distance between the initial motor command and the alignment motor command. As an example, in the case down to side, alignment is reached thanks to motor commands = [121, 180, -121] which has an absolute distance of 2 with the initial motor commands = [120, 180, -120]. The distance between the initial motor commands and the alignment motor commands follows this equation:

$$motor\_commands\_distance = \sum_{i=1}^{3} abs(aligned\_motor\_commands[i] - initial\_motor\_commands[i])$$

After more than 30 experiments on two cases, we noticed that at every tries during the local random search, alignment is always reached with motor commands that have a distance from the initial motor commands of maximum 3. This distance changes at every tries but is always under or equal to 3. Also we decided to run 30 times the baseline algorithm, and measure and plot for the two slowest cases, the fitness function value of the current motor commands (randomly chosen by the local search algorithm) and the distance between the current motor commands and the initial motor commands (figures 14 and 16)). Then we decided to apply on each plots a linear regression to confirm our intuition that in general, the smaller the distance is, the smaller the fitness function value becomes which gets the Roombot closer to the alignment position. Thanks to the linear regression ($\_f = B0 + B1 * dist$), we measure the coefficient B1. B1 represents the mean increase of the fitness function for every additional motor command distance. For both cases, this mean increase is significant because the corresponding p-value of B1 is lower than 0.05 (Figures 15 and 17).
According to this results, we decide to add the following constraint to the baseline solution: the random search algorithm should only reach random positions that have a motor commands distance with the initial ones lower or equal to three using.

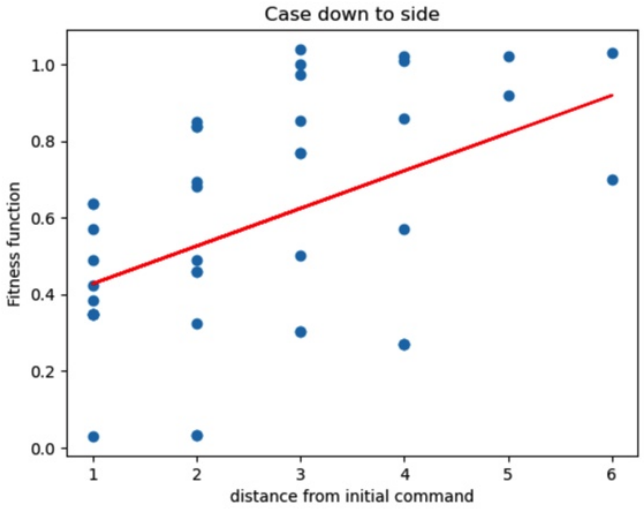

Figure 14: Fitness function value of the current motor commands in function of the distance between the current motor commands and the initial motor commands, and its linear regression - Case down to side

OLS Regression Results

Dep. Variable:	y	R-squared:	0.221
Model:	OLS	Adj. R-squared:	0.202
Method:	Least Squares	F-statistic:	11.62
Date:	Thu, 29 Dec 2022	Prob (F-statistic):	0.00148
Time:	16:34:22	Log-Likelihood:	-2.8430
No. Observations:	43	AIC:	9.686
Df Residuals:	41	BIC:	13.21
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.3295	0.085	3.894	0.000	0.159	0.500
x	0.0982	0.029	3.409	0.001	0.040	0.156

Omnibus:	3.598	Durbin-Watson:	2.196
Prob(Omnibus):	0.165	Jarque-Bera (JB):	2.446
Skew:	-0.403	Prob(JB):	0.294
Kurtosis:	2.154	Cond. No.	6.72

Figure 15: Results of the regression with coefficient B1 highlighted in the red rectangular - Case down to side

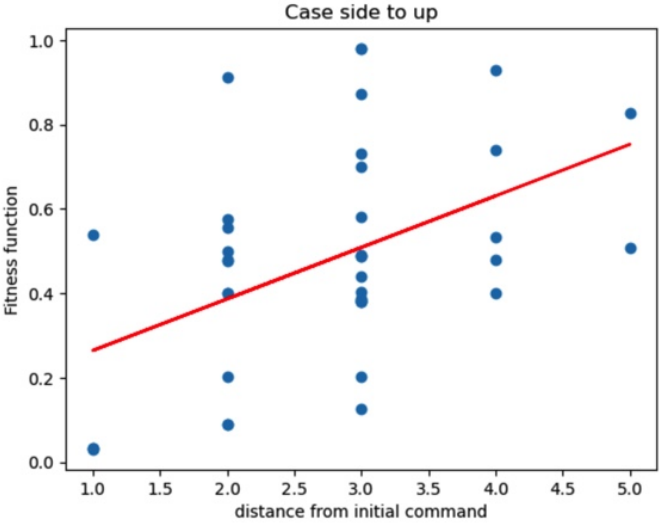

Figure 16: Fitness function value of the current motor commands in function of the distance between the current motor commands and the initial motor commands, and its linear regression - Case side to up.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.212
Model:	OLS	Adj. R-squared:	0.192
Method:	Least Squares	F-statistic:	10.25
Date:	Thu, 29 Dec 2022	Prob (F-statistic):	0.00276
Time:	17:07:52	Log-Likelihood:	2.4183
No. Observations:	40	AIC:	-0.8365
Df Residuals:	38	BIC:	2.541
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.1421	0.111	1.278	0.209	-0.083	0.367
x	0.1222	0.038	3.202	0.003	0.045	0.199

Omnibus:	2.491	Durbin-Watson:	2.243
Prob(Omnibus):	0.288	Jarque-Bera (JB):	2.325
Skew:	0.550	Prob(JB):	0.313
Kurtosis:	2.569	Cond. No.	9.71

Figure 17: Results of the regression with coefficient B1 highlighted in the red rectangular - Case side to up.

3.3 Combine the new fitness function with the limit of the motor commands distance

In the section, we combine the new fitness function calculated in part 3.1, with the limit of the motor commands distance set to three. From now we have been focusing on the two cases with the slowest alignment

speed in the baseline algorithm. Now we apply our changes to all cases and plot the results (Figure 18).

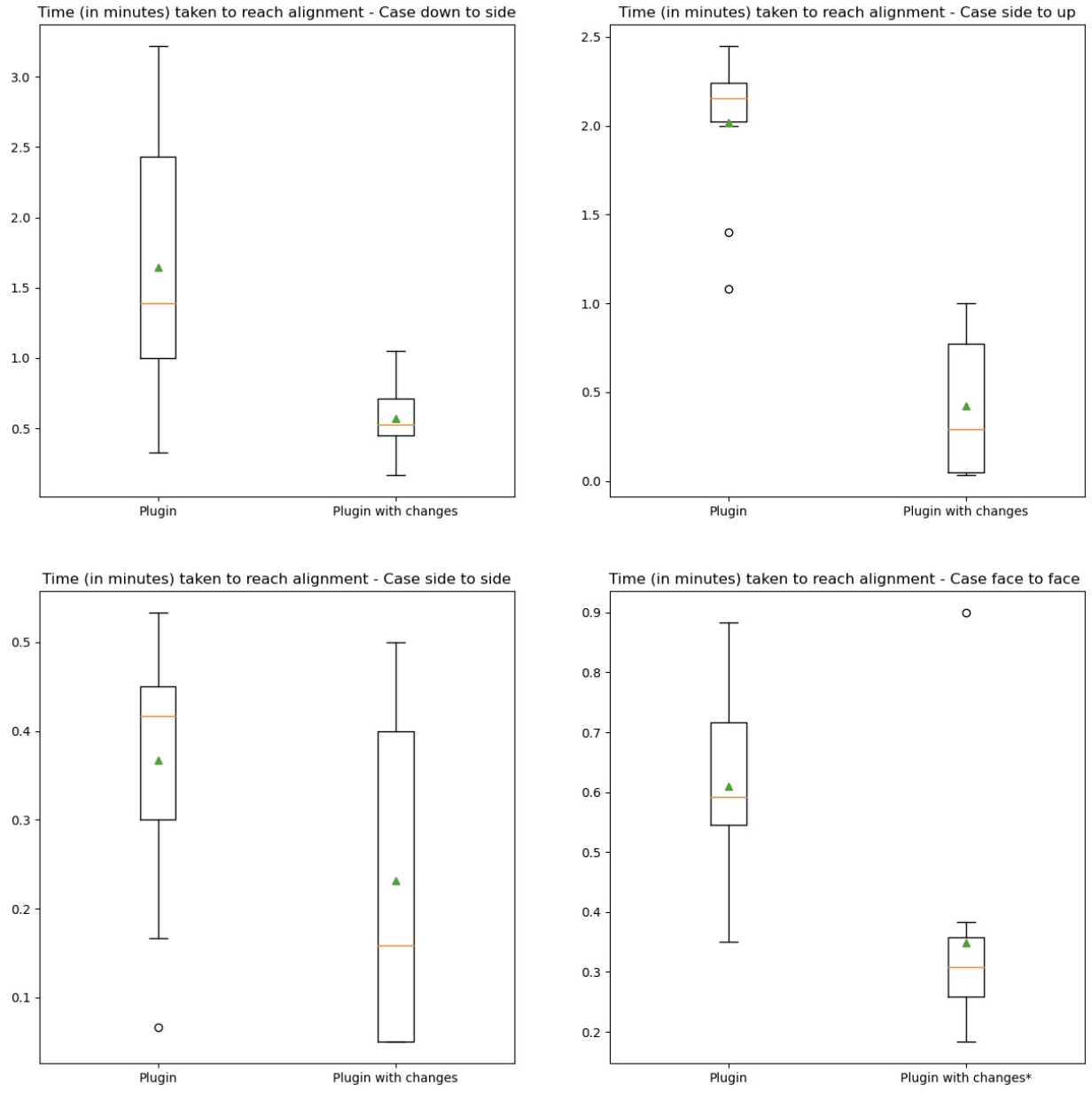

Figure 18: Box plots of the time (in minutes) taken to reach alignment for the baseline solution (left) and the baseline solution with the new fitness function (right) - 10 tries each

For case down to side and side to up, the improvement of alignment speed is again significant because we can notice that in both cases, the upper quartile of the plugin with changes is lower than the lower quartile of the original plugin. For case side to side, the improvement of alignment speed is still significant because the upper quartile of the plugin with changes is lower than the upper quartile of the original plugin. Also the mean alignment speed of the plugin with changes is much lower than the mean alignment speed of the original plugin.
For the case face to face, setting a limit of the motor commands distance equals to 3, led to impossible alignment. Indeed with such a condition, the fitness function value in the case face to face is always better or equal to one (which are the maximum possible values of the function). After 30 tests on the case face to face, we realized that this distance needs to be greater than 4 in the aim to have alignment of the ACM with the passive plate. We decided to modify the distance condition for this case specifically by changing the following: always reach a random neighbor that has motor commands with a distance greater or equal than 4. This additional condition led to great results for the case face to face (Figure 18 bottom right) because the Roombot is able to reach positions with a big enough motor command distance in the aim to get alignment.

Since case face to face has a different motor commands distance condition than the other cases, we need to find a way to differentiate these two groups when the plugin starts. Here is a possible solution: When the plugin starts, the random search algorithm has the new fitness function and the limit of maximum 3 in the motor commands distance. If the first 3 reached positions have a fitness function equal of better than 1, we detect we are in the case face to face and we change the condition of motor command distance by imposing it to be better than 4.

A video showing the docking procedure improvement for all cases thanks to the second approach versus the baseline solution is available at this link: https://youtu.be/Lus3NZTKzrQ .

4 Conclusion

Our second approach which contains a new fitness function and constraints in the motor commands distance led to a significant increase of speed alignment compared to the original plugin. Indeed, with the second approach, the docking procedure of the Roombot is well improved. Results in the second approach were found thanks data collection coming from a big amount of docking experiments. Experiments were sometimes quite challenging due Roombot's reliability problems. Nevertheless, the ultimate goal for docking would be to dock in a single attempt as we have tried in the first approach. For future work on improving Roombot's docking procedure, we could explore a machine learning approach. Indeed, for each cases, we can manually dock the Roombot multiple times and train an algorithm on finding the correct alignment motor commands based on its initial position from sensor reading.

References

[1] Hala Khodr - *Collaborative locomotion in self-reconfigurable modular robots*

[2] S.Hauser, M.Mutlu, P-A.Léziart, H.Khodr, A.Bernardino A.J.Ijspeert - Roombots extended: Challenges in the next generation of self-reconfigurable modular robots and their application in adaptive and assistive furniture - https://www.sciencedirect.com/science/article/abs/pii/S0921889019303379

[3] Guifang Qiao, Guangming Song, Weiguo Wang, Ying Zhang, Yali Wang - *Design and Implementation of a Modular Self-Reconfigurable Robot* - https://journals.sagepub.com/doi/10.5772/58379

[4] Payne, Kenneth, Everist, Jacob, Hou, Feili, Shen, Wei-min - *Single-Sensor Probabilistic Localization on the SeReS Self-Reconfigurable Robot* - https://www.researchgate.net/publication/221032714_Single-Sensor_Probabilistic_Localization_on_the_SeReS_Self-Reconfigurable_Robot

[5] Alexey Romanov - *An Automatic Docking System for Wheeled Mobile Robots* - https://www.researchgate.net/publication/350932640_An_Automatic_Docking_System_for_Wheeled_Mobile_