Dynamic Mobile Robot Obstacle Avoidance and Target Acquisition: A Neuro Fuzzy Evolutionary Approach

Elias Goldsztejn, Prof. Amiram Moshaiov, *Tel Aviv University*

***Abstract*—Autonomous navigation is an increasingly studied subject, targeting from cars used by civilians to robots like search and rescue. In this research we study autonomous navigation for a robot in a known and unknown environment. In contrast to problems where the path of the robot is calculated beforehand, in our study the robot is unaware of the obstacles position; it only knows the target position. This research studies the use of a neuro fuzzy controller trained by a genetic algorithm to optimize multiple objectives. We show the setting of a dynamic environment, the creation of a novel neuro fuzzy controller and its training algorithm. We found that a small set of rules for an aggregated neuro fuzzy controller can make a robot reach its target while avoiding obstacles, and doing so while optimizing time, safety and energy used. Our to bridging the reality gap of standard robot simulations by introducing a simple but complete dynamic model for a differential drive robot and optimizing multiple objectives. We anticipate our assay to be a starting point for the study of more complex arrangements of obstacles in maps. For example, studying an algorithm which trains the controller by exploring more than one single configuration of a map thus generalizing more and making the robot be prepared for a broader set of environments.**

***Index Terms*—autonomus navigation, neuro fuzzy, evolutionary algorithms, journal.**

I. Introduction

TARGET acquisition and obstacle avoidance is the problem where a robot needs to reach a destination while avoiding collision with the obstacles. This field has been much studied; from linear controllers such as PID or Kalman, to nonlinear, such as neural networks or nonlinear backstepping control. Field of forces, sliding model, adaptive controllers have been proposed. In the past two decades, neuro fuzzy systems have gained recognition for letting introduce human expertise to the generation of controllers, while maintaining a vast distance to the amount of involvement in them. These systems are known for their power of generalization, robustness, and simplicity. A fuzzy control system uses fuzzy logic and sets of logical rules to make decisions based on premises and consequences, given the membership functions of the inputs and sometimes of the outputs. The learning capabilities of a neuro fuzzy system are more limited than that of standard artificial networks. For simpler decision-making environments like ours, where the robot navigates in environments where obstacles can be easily identified, and the target position, and robot position and velocities are always known, fuzzy systems can not only suffice but also outperform artificial neural networks with respect to time to reach the target. That is why we decided to make use of neuro fuzzy systems.

Ministry of education, Israel
Manuscript received ..., 2021; revised ...2021

Mobile robot simulations are usually set for kinematic models. Thus, disregarding the great use of power needed to move the robot, and the possibility of slipping. This creates a reality gap; the controllers perform poorly on real world applications. In our study we introduce the equations and code implementation for simulating a dynamic model of a differential drive robot in a friction 2D environment. The robot can sleep frontally as well as laterally, however we did not introduce the X or Y axis moments - we assume the robot is heavy and wide enough not to flip in these axes.

Many navigation models optimize only one objective: the time to reach the target. This simplification of the problem goes well with gradient based training algorithms like ANFIS. But this oversimplification brings downsides, as it is in many cases that we want to optimize more objectives like safety or power consumption. In our study, we optimize not only for time but also safety and power. This closes even more the breach of the reality gap, as the robot is more careful in its use of resources and safety.

When dealing with problems where gradients cannot be calculated, a set of training algorithms exist. Some of them are: Swarm, ABC and Genetic algorithms. These algorithms are more and more being studied as they mimic natural behaviors that are astonishing because of their relative simplicity but highly intricate structures they create. The training algorithm chosen for this study is the Genetic algorithm, more specifically NSGA-II, which is a widely used multi-objective optimization algorithm.

This paper is organized as follows: Section I deals with the representation of the problem and its dynamic model. Section 2 explains the procedure for one single simulation of the robot. Section 3 explains the structure of the controller. Section 4 introduces the training algorithm. Section 5 shows the controller solutions and the best paths. Section 6 introduces generalization for an unknown environment. Section 7 concludes the results of the study and gives possible improvements for the future.

II. Navigation and Robot Representation

In our study, a differential mobile robot is represented. Our model simulation is obtained using a MATLAB simulation toolbox [1]. A diagram of the robot can be seen in 1.

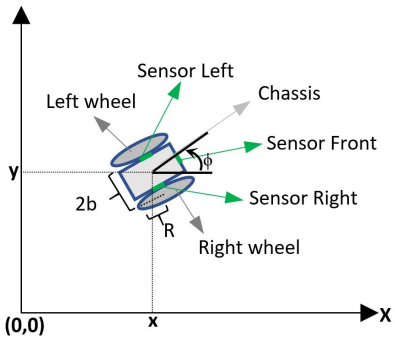


Fig. 1: Robot diagram.

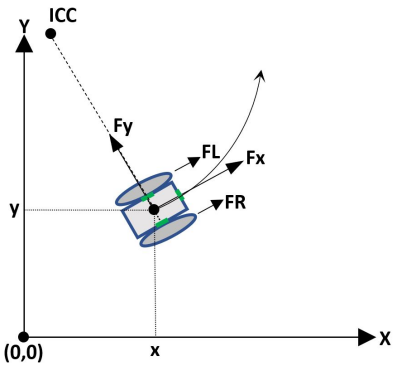


Fig. 2: Forces diagram.

The robot has two wheels with radius R, the distance between them is $2b$. It has three obstacles' sensors (left, front and right). Moreover, always the distance and direction to target is known. The position of the robot is (x, y), and its angle from the reference is ϕ.

A. Dynamic Model

Much research has been done for kinematic holonomic differential drive robots (without slipping of the wheels or car), but little research for dynamic ones. In this research we introduce a new dynamic model for a nonholonomic differential drive robot with friction. The model equations include wheel sleep and lateral slip. The equations are then utilized to calculate new position and velocities of the robot at each time step.

An analytical model can be found in [2]. A Simulink implementation without lateral slip can be found in [3].

B. Dynamic Equations

$$FL = frictionFunc(aL, WL, x, y, frModel) \quad (1)$$

$$FR = frictionFunc(aR, WR, x, y, frModel) \quad (2)$$

$$Fx = FL + FR - sign(FL + FR) * |FL - FR| \quad (3)$$

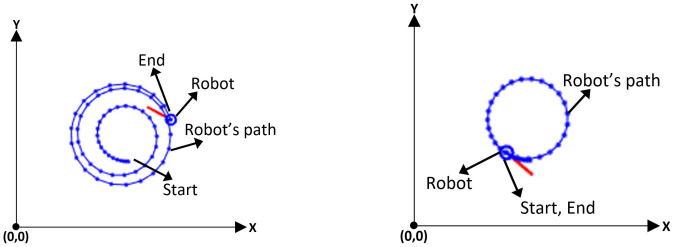
$$ICC = 2b * \frac{Vx}{(WL - WR) * R} - Vy * dt \quad (4)$$

$$Iz = \frac{M * b^2}{8} \quad (5)$$

$$Fy = frictionFunc((WL - WR)^2 * ICC * M, Vy, x, y, frModel) \quad (6)$$

$$WA = \frac{(FL - FR) * b}{Iz} \quad (7)$$

Where: WL, aL and WR, aR are the angular speed and the torques of the left and right wheels. $frictionFunc$ returns the resultant force given the force and velocity applied together with its position and the friction model. It calculates so by applying the chosen friction model to the forces applied. In our study we chose the coulomb friction model - a step function. A diagram of the forces can be seen in figure 2. Given the forces,



(a) Slipping robot. (b) Not slipping robot.

Fig. 3: Friction demonstration.

for the next time step: $t + dt$ we get the updated velocities, and angle of the robot:

$$Vx(t+1) = Vx(t) + (Fx/M) * dt \quad (8)$$

$$Vy(t+1) = Vy(t) + (Fy/M) * dt \quad (9)$$

$$\varphi(t+1) = \varphi(t) + WA * dt \quad (10)$$

$$WL(t+1) = WL(t) + (FL/R) * dt \quad (11)$$

$$WR(t+1) = WR(t) + (FR/R) * dt \quad (12)$$

$$x(t+1) = x(t) + Vx(t) * dt \quad (13)$$

$$y(t+1) = y(t) + Vy(t) * dt \quad (14)$$

$$\varphi(t+1) = \varphi(t) + WA * dt \quad (15)$$

C. Rotation Under Friction Example

Figure 3b and figure 3atwo examples of the robot rotating. We use a step model of friction, with $u = 1$, and $u = 0.1$ respectively at all points of the map. $M, R = 1m.b = 0.5m.Vx = -3m/s.WL–WR = 0.75rad/s.$

In figure 3b the robot describes a circle because friction does not let it slip. In figure 3a the robot starts slipping because its centripetal force is bigger than the friction force. When its angular speed decreases as its turning radius increases, the centripetal force decreases linearly until the robot does not slip anymore.

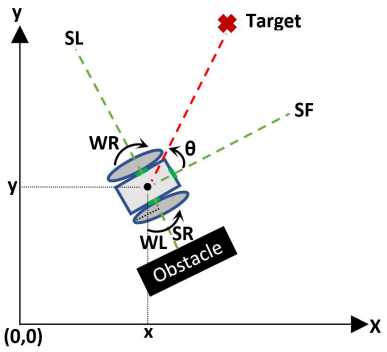


Fig. 4: Kinematic diagram.

```
Initialize controller, map, target, and robot
position.
Time t=0
While (not reached target and
t<tMax)
        SL, SF, SR = getObstacleDist(map, x, y,
        φ)
        TD, θ = getTargetDistAndDir(target, x,
        y, φ)
        aL, aR = controller(robot velocities,
        SL, SF, SR, TD, θ)
        Vx, Vy, WL, WR = dynamicModel(robot
        position and velocities, aL, aR)
        O1, O2, …, On =
        updateObjectiveMeasures(t, SL, SF, SR,
        Vx, Vy)
        Visualize(t) // optional
        t = t + dt
return O1, O2, …, On
```

Fig. 5: Simulation pseudo-code.

III. Simulation

The robot is constantly getting readings from its lidar sensors placed at the left, front and right sides of the robot, which indicate its distance to the obstacles and its direction and distance with respect to the target. It also knows the speed of its wheels and its Vx and Vy velocities. With this information, the controller decides the amount of torque to apply on each wheel. Figure 4 is a diagram of the robot and map scenarios at a certain time t.

At the beginning of the simulation a map is generated, the targets and initial position of the robot are set, and the controller parameters are fixed. Figure 5 is the pseudo code for the robot simulation.

IV. Controller

The controller chosen for this research is a fuzzy one. Fuzzy controllers have increasingly been used in machine control because of its versatility, robustness, and simplicity. The consequence part of the controller is a Tagasaki-Sugeno type, which is known for its low run time complexity. Our fuzzy controller has 4 layers: fuzzification, fuzzy operation (rule), normalization, defuzzification. In our study, the T-S parameters are constant, and the rules are human generated. Figure 6 is the representation of our fuzzy controller.

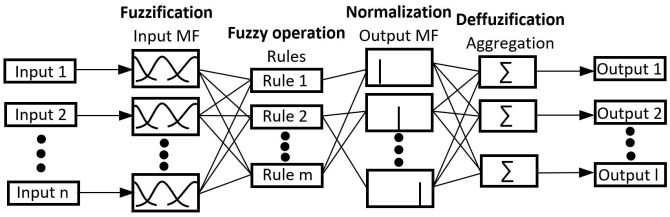


Fig. 6: Fuzzy controller.

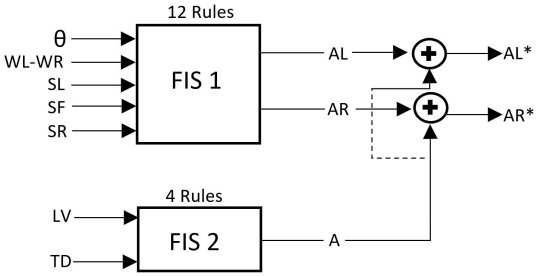


Fig. 7: Controller structure.

A. Structure

Our controller is a sum of two different ones. The first one oversees avoiding obstacles and reaching the target. The second one stabilizes the robot velocity. The inputs of the controllers are the robot velocities - distance and angle to target - and distance to obstacles. The structure used is of a fuzzy aggregation. It is generally used for autonomous decision models where there are sets of independent rules. The aggregation method helps reduce the number of rules while increasing the complexity of the decision making. Figure 7 shows the structure of the controller.

The first FIS has the following inputs and outputs:

Inputs:

- θ – Direction to target.
- WR – Angular velocity of the robot.
- SL – Distance to left obstacle.
- SF – Distance to front obstacle.
- SR – Distance to right obstacle.

Outputs:

- AL – Desired left torque.
- AR – Desired right torque.

The second FIS has the following inputs and outputs:

Inputs:

- LV – Linear velocity of the robot.
- TD – Distance to target.

Outputs:

- A – Stabilization torque.

FIS 1 Rules	
Rule 1	θ == Left & WL-WR == Negative => AL = Zero AR = Zero
Rule 2	θ == Left & WL-WR == Zero => AL = Negative AR = Positive
Rule 3	θ == Left & WL-WR == Positive => AL = Negative AR = Positive
Rule 4	θ == Center & WL-WR == Negative => AL = Positive AR = Negative
Rule 5	θ == Center & WL-WR == Zero => AL = Positive AR = Negative
Rule 6	θ == Center & WL-WR == Positive => AL = Negative AR = Positive
Rule 7	θ == Right & WL-WR == Negative => AL = Positive AR = Negative
Rule 8	θ == Right & WL-WR == Zero => AL = Positive AR = Negative
Rule 9	θ == Right & WL-WR == Positive => AL = Zero AR = Zero
Rule 10	SL == Close => AL = Positive AR = Negative
Rule 11	SF == Close => AL = Positive AR = Negative
Rule 12	SR == Close => AL = Positive AR = Negative

Fig. 8: Fis 1 rules.

FIS 2 Rules	
Rule 1	LV == Negative Fast | LV == Negative Slow => A = Positive Big
Rule 2	LV == Zero=> A = Positive Small
Rule 3	LV == Positive Fast | TD == Close => A = Negative Big
Rule 4	LV == Positive Fast | TD == Far => A = Negative Small

Fig. 9: Fis 2 rules.

B. Rules

There is a set of 12 rules for the first FIS 8. These rules indicate the robot to avoid obstacles and reach the target.

The second FIS 9 has 4 rules. These rules help the robot stabilize its velocity.

C. Membership Functions

The set of membership functions for each variable is fixed. Some variables have 1 to 3 membership functions while others have 5 10.

D. Membership function and rule parameters

The membership functions are of the type: gaussian bell. With a, b, and c parameters:

$$gbellmf(x, a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}}$$

Rules also have weights. Most these parameters are adjusted via the training algorithm.

Variable	MF
θ	Left – Center - Right
WL-WR	Negative – Zero - Positive
SL	Close
SF	Close
SR	Close
LV	Neg. Fast – Neg. Slow – Zero – Pos. Slow – Pos. Fast
TD	Close - Far
AL and AL*	Negative – Zero - Positive
AR and AR*	Negative – Zero - Positive
A	Neg. Big – Neg. Small – Zero – Pos. Small – Pos. Big

Fig. 10: Membership functions.

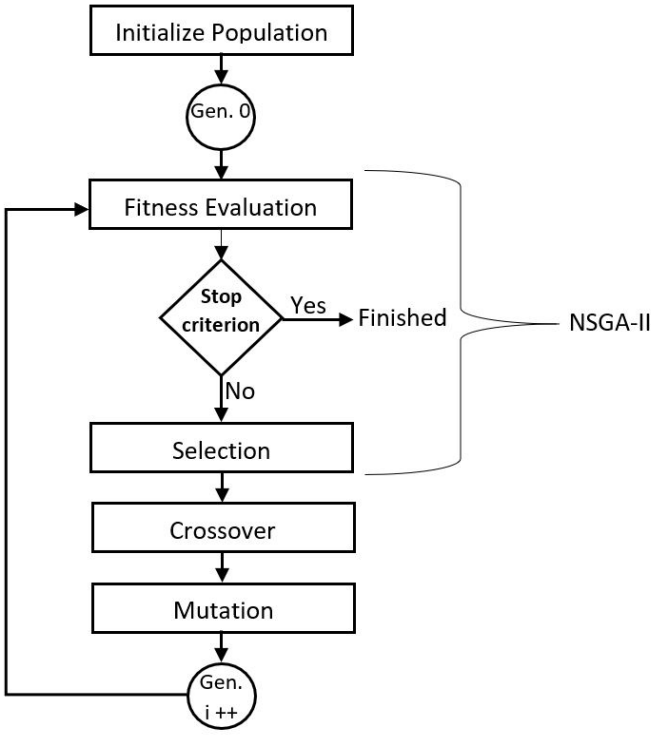


Fig. 11: Evolutionary algorithm.

V. Training Algorithms

There exist several training algorithms for Fuzzy Systems. Much documentation can be found in [4].

For machine control Fuzzy systems with one output, ANFIS training algorithm is widely used however, for multiple output problems – like in our research – genetic, swarm, ABC, etc. algorithms are used.

The advantage of using genetic algorithms, apart from the fact that it is adequate for optimization of multiple objectives, is that it decreases the runtime as evaluation functions can be run in parallel.

Our approach is to train the parameters of the controller using NSGA-II, a genetic algorithm for multiple objectives: NSGA-II calculates the fitness evaluation and performs selection according to an ordering of pareto fronts and density of the solutions, to prioritize dominant solutions with low density.Figure 11 is a schematic of the GA: (NSGA-II).

A. Training Pseudo-code

The NSGA-II GA is applied on the parameter of the controllers. The fitness function is the simulation of the robot, and the fitness evaluation step evaluates the output of each simulation which are the objective measures – Minimum time – Maximum Safety – Minimum energy 12.

The fitness evaluation is done by comparing the objective measures in the pareto front. The dominant solutions pass on their controller parameters to the next generations.

B. Training algorithm advantages and disadvantages

Parallelization of the genetic algorithm is achieved by evaluating each generation of controllers at the same time.

```
Initialize sets of controllers, map, target,
and robot position
Generation num = 0
While (num of generations < maxGen)
        For (i=0 to i= Population Size)
                O_1i,…, O_ni =
                robotSimulation(controller_i)
        If (Stop criterion)
                Break;
        NSGA-II -> Selection
        Crossover
        Mutation
        Generation num += 1
```

Fig. 12: Pseudo-Code.

This can be done because each controller is independent. This makes the training process a fast one in comparison to serial training algorithms like gradient descent. A disadvantage of this specific training algorithm is that exploration and exploitation of solutions depend directly on the input parameter tunings that the author chose. This directly impacts on the overfitting of the algorithm – a serious problem when trying to generalize to other environments. The parameters must be chosen taking in consideration that in one hand, we want a set of "good" solutions, but in the other hand, we do not want overfitted solutions to the specific environment.

C. Objective measures

There are 3 objectives, which the training algorithm optimizes, namely:

1) Time until target is reached.
2) Safety.
3) Energy.

The energy is calculated as: $\int_0^S Fds$.At each time step F is calculated as the difference in velocities between the previous and the current time step, and ds as the difference in positions of the robot. Safety is calculated with the help of two objective measures (SM2 and MinDist [5]) which we will call S1 and S2.

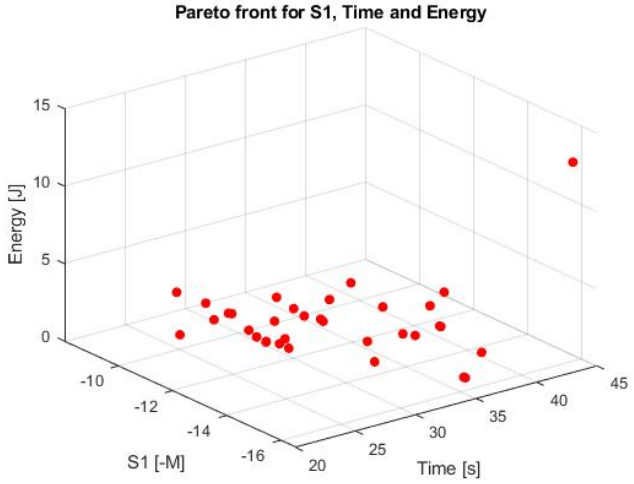
S1 is the mean of the minimum distance registered at each time step from the three obstacle sensors: SL, SF and SR. This safety measure has the disadvantage that it does not penalize the robot being close to obstacles if it is for a short period of time. That is why we include S2.

S2 is the minimum distance registered from the three obstacle sensors through the whole simulation.

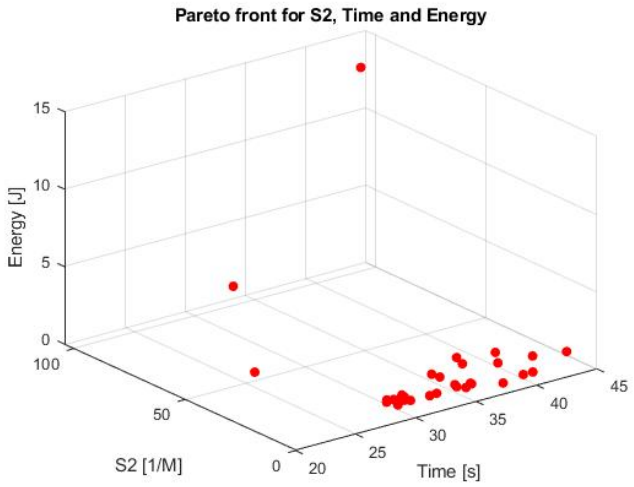
Both S1 and S2 are maximized upon training.

VI. Training

The parameters which undergone training, initially, are the membership function parameters (a,b and c) of SL, SF and SL, together with the 16 rules. In total: 22 parameters. The wheel speed limit is $1.5[rad/sec]$. The obstacle sensors have a reach of $20[M]$.



(a)



(b)

Fig. 13: Pareto fronts.

A. Training tuning parameters

The parameters for the GA are the following:

1) Population size = 100.
2) Maximum generations = 100.
3) Crossover fraction = 0.6.
4) Pareto fraction = 0.9.

The training algorithm was run on one computer with the following specifications:

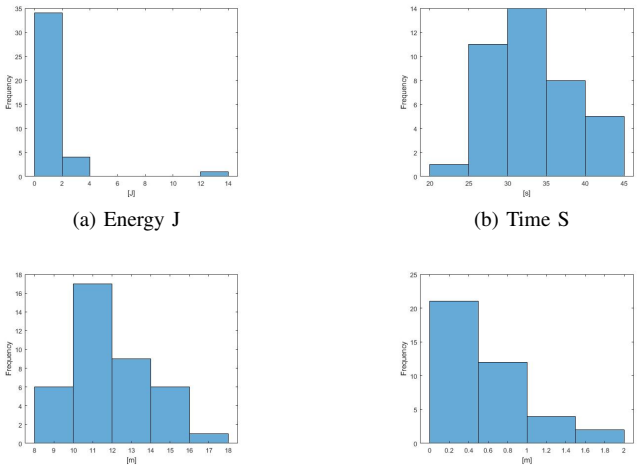
1) 8GB RAM Intel core i7 8th gen – 4 cores
2) GPU: NVIDIA GeForce MX150 2GB RAM

The training took 10 minutes.

B. Pareto Fronts

Because we have 4 objective measures, we display two graphs 13a, 13b. The first one shows the pareto front of dominant solutions of the measures: Time, energy and S1. The second one: Time, energy and S2.

Figure 14 show the histograms for the 4 objective measures.



(a) Energy J (b) Time S

(c) S1 M (d) S2 M

Fig. 14: Histograms of objective measures.

C. Robot paths for best solutions

Figures 15a and 15b show a couple of the best solutions for the robot path. The black boxes are obstacles, the blue line is the path the robot took. The red cross is the target. The color of the floor indicates the friction coefficient of the floor. White being with a coefficient of 0.3, and purple with 0.1.

D. Robot paths analysis

From the paths above we can see that the controller parameters were able to avoid obstacles but keep the robot directions to the target as much as possible. In solution 1, the robot went through the top of the map and in solution 2 through the bottom. Solution 1 is the one which invested the least energy. This can be deduced from the path being the smoothest one. Solution 1 and 2 are similar in respect with time until target acquisition and S2 measure.

VII. Theoretical best vs. results obtained.

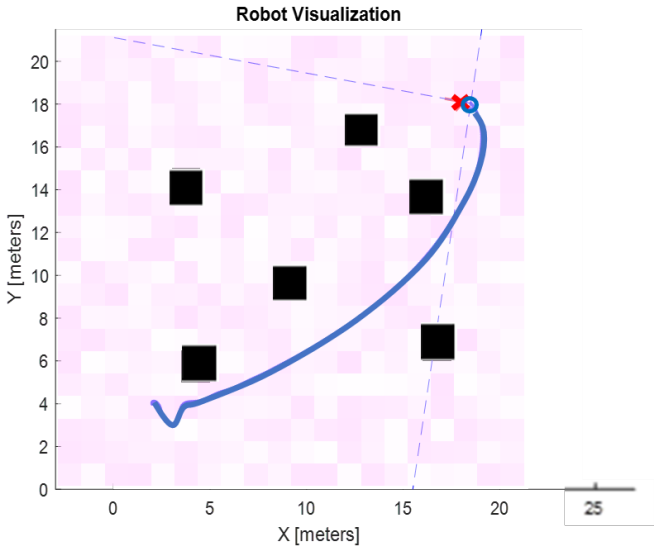
The initial position of the robot is $x = 2$, $y = 4$. And the target is placed at $x = 18$, $y = 18$. If there were no obstacles, and neglecting the time it takes the robot to reach maximal velocity (of 1.5 m/s) and stopping at target, the minimum time to reach the target would be:

$$T = \frac{\sqrt{(18-2)^2 + (18-4)^2}M}{1.5\ M/S} = 14.18\ S \tag{16}$$

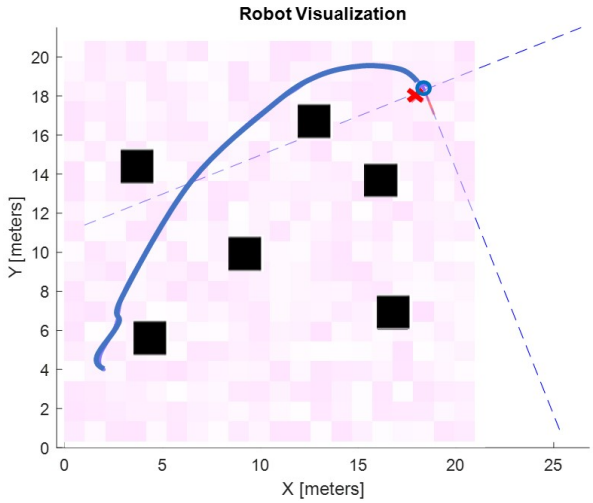
The minimum energy possible given the average velocity of the robot $\hat{v}$ is:

$$E = 2 \times \frac{1}{2} \times M \times \hat{v}^2 \tag{17}$$

The best, median and average objective measures of the set of best solutions found is shown in 16.



(a) Solution 1
$Time = 30.20S$
$S1 = 14.91M$
$S2 = 0.80M$
$Energy = 0.80J$



(b) Solution 2
$Time = 29.30S$
$S1 = 8.01M$
$S2 = 0.70M$
$Energy = 1.19J$

Fig. 15: Pareto fronts.

	Time [s]	Energy [J]	S1 [m]	S2 [m]
Best	23.30	0.57	16.58	1.58
Median	34.00	0.90	11.95	0.44
Mean	33.30	1.51	11.98	0.50

Fig. 16: Results.

	Time ratio	Energy [J]	S1 [m]	S2 [m]
Mean	0.37	0.92	0.67	11.02

Fig. 17: Mean results.

VIII. Generalization to unknown environments

We wanted to test the controller performance in an unknow environment – different robot initial position, target position and friction coefficients of the ground. For this reason, we introduce a concept, which we call conservative set. This set narrows down the number of the final population of non-dominated controllers generated with the evolutionary training algorithm, to those which objective measures are closer to the mean objective measure obtained:

$$C = min\, r,\ s.t. \exists\ n,\ s.t.\ O_{ni} \in |mean(O_i) - r \times std(O_i)|\ \forall\, i \tag{18}$$

This set contains controllers which perform satisfactorily in all objective measures. In our case we got two set of controller parameters in C and we chose randomly one of them. For the same position of obstacles but with different robot and target positions, and friction coefficients of the ground we ran 100 simulations.

We define:

Success rate: Percentage of simulations were the robot reached its target without colliding into objects or getting outside the map limits. Time ratio: Division of the time it would take the robot to reach the target if it followed a straight line by the actual time of the simulation. We got the following results:

Success rate: 95%

A. Generalization results analysis

The success rate obtained shows that the controller chosen could present a good behavior in most of the different scenarios. Moreover, it shows that despite training an algorithm over one fixed friction map; target, and obstacle positions, the controller was general enough to extrapolate to unknown scenarios. The time ratio shows that the actual mean time of the simulations took roughly twice as much than the ideal time. This result is good, since the ideal time does not consider the time it takes avoiding the obstacles. The mean energy is close to the best expected energy of 1 J. S1 and S2 results are satisfactory, considered the configuration of the obstacles in the map.

B. Simulation example

Figure 18 shows an example of one random configuration of the robot initial position and target position. As can be seen clearly, the robot maintains a safe distance from the obstacles while reaching the target.

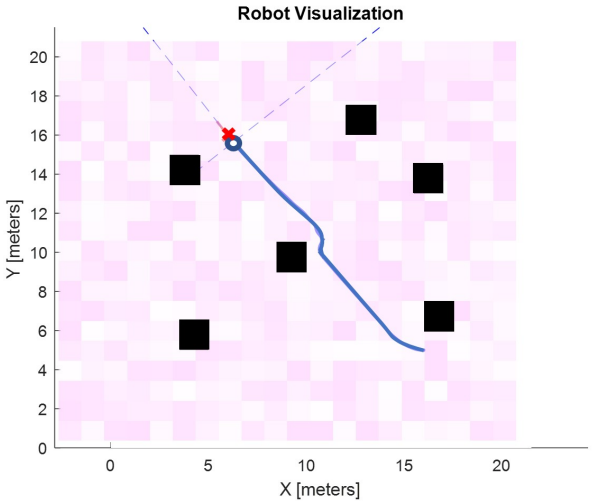


Fig. 18: Random configuration solution.

IX. Conclusion

This paper has proposed a novel neuro-fuzzy controller and training algorithm for an obstacle avoidance and target acquisition problem for a differential drive robot in a dynamic environment. We successfully implemented a dynamic model with friction for a 2D differential drive robot, and, made use of a GA for the training of the neuro-fuzzy controller parameters. A process for creating a controller that closes the reality gap by optimizing time, energy and safety has been created.

It was shown that the controller obtained had a success rate of 95% and overall good performance in the different objective measures for unknown environments. Moreover, the time complexity of the algorithm has been shown to be small.

A. Future improvements

We believe that a more sophisticated training algorithm that evaluates the fitness function over several different environments, rather than only one, could increase the performance of the controller for a set of unknown environments. Thus, increasing the success rate and overall performance of the controller.

Furthermore, we believe that increasing the number of lidar sensors in the robot would make it more precise. Also, adding an aggregated fuzzy system which reduces the angular velocity of the robot could help overcome some issues the robot sometimes faces when it makes fast turns.

Finally, for the robot to be adequate for more obstacle dense maps, where obstacles may sometimes enclose the robot, we propose creating a memory unit that stores a lidar map obtained in real time. With the help of a heuristic path planning algorithm*, the same fuzzy controller could be used, however changing the target position so that it lies along the path planned.

* A pseudo path, which places the heuristic target the closest possible to the actual target inside the real time lidar map.

B. Code

The simulation environments, training codes, and testing options are available in [6]. Follow README instructions.

mds
April 2021

APPENDIX A
ICC DERIVATION

Appendix one text goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] Mobile Robotics Simulation Toolbox
`MathWorks Student Competitions Team (2021).`
https://github.com/mathworks-robotics/mobile-robotics-simulation-toolbox Accesed January 2021.

[2] Torres, E.
`TRACTION MODELING AND CONTROL OF A DIFFERENTIAL DRIVE MOBILE ROBOT TO AVOID WHEEL SLIP`
https://www.academia.edu/29161911/TRACTION_MODELING_AND_CONTROL_OF_A_DIFFERENTIAL_DRIVE_MOBILE_ROBOT_TO_AVOID_WHEEL_SLIP Acessed January, 2021.

[3] CERKALA, J. and JADLOVSKÁ, A. *NONHOLONOMIC MOBILE ROBOT WITH DIFFERENTIAL CHASSIS MATHEMATICAL MODELLING AND IMPLEMENTATION IN SIMULINK WITH FRICTION IN DYNAMICS.* Acta Electrotechnica et Informatica, 15(3), pp.3-8, 2015.

[4] Karaboga, D. and Kaya, E. *Adaptive network based fuzzy inference system (ANFIS) training approaches: a comprehensive survey.* Artificial Intelligence Review, 52(4), pp.2263-2293, 2018.

[5] Nelson David Munoz Ceballos, Jaime Alejandro Valencia and Nelson Londono Ospina. *Quantitative Performance Metrics for Mobile Robots Navigation.* INTECH Open Access Publisher, 2010.

[6] elias github
`elias github`
eliasgithub

APPENDIX B
BIBLIOGRAPHY

[7] Chaochao Chen, Paul Richardson. *Mobile robot obstacle avoidance using short memory: a dynamic recurrent neuro-fuzzy approach* SAGE Journals. https://journals.sagepub.com/doi/abs/10.1177/014233121036664 Acessed January, 2021.

[8] Lee, C. and Chiu, M. *Recurrent neuro fuzzy control design for tracking of mobile robots via hybrid algorithm.*

[9] Subathra, B. and Radhakrishnan, T. *RECURRENT NEURO FUZZY AND FUZZY NEURAL HYBRID NETWORKS: A REVIEW.* Instrumentation Science & Technology, 40(1), pp.29-50, 2012.

[10] Axelrod, B., Kaelbling, L. and Lozano-Pérez, T. *Provably safe robot navigation with obstacle uncertainty. The International Journal of Robotics Research.* The International Journal of Robotics Research, 37(13-14), pp.1760-1774, 2018.

[11] Althoff, D., Kuffner, J., Wollherr, D. and Buss, M. *Safety assessment of robot trajectories for navigation in uncertain and dynamic environments.*

[12] Salleh, M. and Hussain, K., 2016. *A Review of Training Methods of ANFIS for Applications in Business and Economics.* International Journal of u- and e- Service, Science and Technology, 9(7), pp.165-172, 2016.

[13] Salih A., Moshaiov A. *Many-objective Topology and Weight Evolution of Neural Networks and its Application to Robot Navigation Control.* School of Mechanical Engineering, Tel-Aviv University, Tel-Aviv, Israel, Mechanical Engineering Department, ORT Braude College of Engineering, Karmiel, Israel, Sagol School of Neuroscience, Tel-Aviv University, Tel-Aviv, Israel

[14] Xu, L., Huang, C., Li, C., Wang, J., Liu, H. and Wang, X. *Estimation of tool wear and optimization of cutting parameters based on novel ANFIS-PSO method toward intelligent machining.* Journal of Intelligent Manufacturing, 32(1), pp.77-90, 2020.