Efficient autonomous navigation for mobile robots using machine learning

Abderrahim Waga[1], Ayoub Ba-ichou[1], Said Benhlima[1], Ali Bekri[1], Jawad Abdouni[2]
[1]Department of Computer Science, Faculty of science, Moulay Ismail University, Meknes, Morocco
[2]Department of Computer Science, National School of Applied Sciences, Ibn Tofail University, Kenitra, Morocco

Article Info

Article history:

Received Jul 16, 2023
Revised Dec 11, 2023
Accepted Jan 15, 2023

Keywords:

Autonomous navigation
Ensemble methods
Knowledge learning
Learning from demonstration
Machine learning

ABSTRACT

The ability to navigate autonomously from the start to its final goal is the crucial key to mobile robots. To ensure complete navigation, it is mandatory to do heavy programming since this task is composed of several subtasks such as path planning, localization, and obstacle avoidance. This paper simplifies this heavy process by making the robot more intelligent. The robot will acquire the navigation policy from an expert in navigation using machine learning. We used the expert A*, which is characterized by generating an optimal trajectory. In the context of robotics, learning from demonstration (LFD) will allow robots, in general, to acquire new skills by imitating the behavior of an expert. The expert will navigate in different environments, and our robot will try to learn its navigation strategy by linking states and suitable actions taken. We find that our robot acquires the navigation policy given by A* very well. Several tests were simulated with environments of different complexity and obstacle distributions to evaluate the flexibility and efficiency of the proposed strategies. The experimental results demonstrate the reliability and effectiveness of the proposed method.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Abderrahim Waga
Department of Computer Science, Faculty of science, Moulay Ismail University
Avenue Zitoune, Meknes 11201, Morocco
Email: a.waga@edu.umi.ac.ma

1. INTRODUCTION

Nowadays, mobile robots have become an obligatory part of human daily life. since the latter needs help in certain tasks, for example, the navigation of autonomous cars [1]-[3] or autonomous underwater vehicles [4]. In general, autonomous navigation has to be realized by combining two main tasks, global path planning [5]-[9] and local motion control [10]. The generated trajectories are often optimal by optimizing some criteria such as path length or collision risk with obstacles [11]. Trying to adapt this task to unknown environments or unexpected actions is still a challenge since it requires an expert who will re-program the sequence of actions or movements that the robot must perform to generate an optimal trajectory. In robotics, in order to overcome this challenge, some researchers try to exploit the strength of artificial intelligence with autonomous navigation. In this sense, learning from demonstration has attracted more interest in the last ten years since it tries to imitate the behavior of an expert [12] and also involves interactions between the mobile robot and the unknown environment and many other constraints, which poses some difficult requirements for artificial intelligence. This aspect is inspired by how humans learn by being guided by experts from infancy to adulthood, so the principle is to teach new tasks to mobile robots without doing heavy programming [13].

The fundamental idea of learning from demonstration is that the robot learns through several demonstrations by the expert. The demonstrations collected by the mobile robot are sequences of state-action pairs that are recorded during expert navigation. Unfortunately, this aspect finds challenges in the field of autonomous navigation, e.g., a proposed approach based on inverse reinforcement learning [14] but requires a large training cost or the proposed approach requires an increase in data [15]. The majority of existing work in this area of demonstration-based learning focuses on manipulative arm robots [16] or humanoid robots [17], but the use of this technique is rarely applied to wheeled mobile robots in general. In this paper, we propose a new intelligent and efficient technique to overcome these problems that relies on the proper education of the mobile robot, by generating demonstrations from the A* expert [18]. The generated intelligent models are validated and tested on new environments seen for the first time. The rest of the paper is organized as follows: in the second section, we will see some innovative research about our topic, followed by the background and the tools used in this paper. The fourth section will begin with the experiments performed to validate our approach, followed by a discussion of the obtained results, and finally with a conclusion and possible future works.

2. RELATED WORK

In this section, we will discuss the work on autonomous navigation, followed by the work on learning from demonstrations and the relationship with autonomous navigation. Autonomous navigation is the crucial key of mobile robots, but to guarantee perfect and collision-free navigation, our robot must be equipped with several sensors, which makes this task a little bit complex. Zainuddin *et al.* [19] proposed an autonomous navigation system using a single camera Microsoft Kinect XBOX 360 as input to exploit the depth of the scene captured by the camera. He chose to use this sensor for three reasons low cost, efficiency, and real-time processing. The result showed the effectiveness of the proposed method in indoor environments, but the author points out that there is still a need for improvements in the depth map obtained. Other approaches have been proposed [20] based on deep reinforcement learning to educate the robot on a very good autonomous navigation policy. This method is based not only on the depth of the image but also on a feature extractor improved predictive recurrent neural network (PreedRNN++) to exploit the spatio-temporal features well. A mobile robot must necessarily reach its final goal in a reasonable time and avoid all obstacles.

According to Adwan [21], an intelligent approach based on path planning is proposed. This approach guarantees an optimal path and also does not require much computation compared to traditional methods such as A* and potential field. The author divided the navigation space into subspaces and also limited the degree of robot rotation to $45°$ to minimize the localization error and computation time. In the last ten years, several contributions have been present in learning by demonstration [12], [22], it is a powerful technique since it gets its strength from the way humans learn new features. Several strategies in this area have been studied [23], [24]. In the first research, Xia presented a smart method based on reverse reinforcement learning [25]. The idea behind this technique is to represent a nonlinear policy by exploiting neural inverse reinforcement learning (NIRL). In this proposed method the mobile robot not only imitates the behavior of an expert but also understands why the expert has chosen such an action based on the reward function. The results showed the feasibility and robustness of the method, even if in dynamic and unknown environments. In the second research, they tried to educate the robot by trying to generalize the rapidly exploring random trees RRT^* cost function by merging RRT^* and inverse reinforcement learning. The results showed that the approach can with a large percentage approximate the RRT^* cost function. Local and global planning, both, requires a map, in [26], overcame this challenge by exploiting deep imitation learning. The proposed map-less navigation method was tested on four environments twenty times and the result shows that the success rate reaches up to 75%. The strength of this method is the increase of samples since they generated 250,000 samples and also the precision and coverage of the Lidar sensor.

In our work, we facilitate the control of the mobile robot and make it intelligent and allow the robot to adapt to new situations without doing heavy programming by exploiting only the sensory inputs. Our research focuses on learning through several demonstrations by experts. This leads us to our childhood, when we try to study or imitate well the behavior of our parents, mobile robots also need to acquire this skill. Especially in new situations where robots have to adapt to unknown events without human supervision. We tried to simplify this process based only on scalar inputs and our robot managed to adapt to most environments without any prior information.

3. BACKGROUND

The transfer of the navigation policy from the expert to our mobile robot can be considered a Markov decision process problem since this policy is only correspondence between the state and suitable action. Therefore, the expert will navigate through all the training environments, and at the same time, it will generate in each crossed state an appropriate action that will be transmitted to the different machine learning techniques.

After the generation of the dataset, another critical step that applies called preprocessing is in particular the normalization for the scaling of all the attributes. About the machine learning techniques used in this paper, we focus on four techniques namely the random forest, Xgboost, Adaboost, and Bagging classifier [27]. These four algorithms are known in the classification literature. Still, our contribution will be to group them to obtain efficient models using the voting technique [28] which will help to improve the generated models. We have chosen to use these techniques since we have structured data, i.e., tabular data and we have generated a medium-sized dataset.

3.1. A* algorithm

Path-finding algorithms are essential for determining efficient routes between two points by optimizing a cost function. These algorithms search for coordinates within a data structure under specific conditions. A* is a widely used example, which computes costs for neighboring nodes and selects the path with the lowest cost iteratively until all nodes are explored, ultimately identifying the optimal route based on a cost function F(n).

$$F(n) = g(n) + h(n), where : \tag{1}$$

- g(n) is the cost of moving from one node to another. This will vary from node to node.
- h(n) is the heuristic approximation of the value of the node. It is not a real value but an approximate cost.

3.2. Dataset generation

From the previous section, we described that the policy is just correspondence between states and suitable actions, so it can be schematized by the Markov decision process, from this assumption, the data collector will have to be composed of a pair of useful state actions for the mobile robot. In our autonomous navigation system, all the data is drawn from expert demonstrations. Given an M-length demonstration T_i for $i = 1, \ldots, M$ the generated trajectory can be written as (2):

$$D^{(i)} = \left\{ \left(x_t^{(i)}, y_t^{(i)} \right) \right\} \tag{2}$$

The expert will generate a dataset after navigating the training environments. We have the model of a mobile robot, a robot which has eight ultrasonic sensors to measure the distance, and also, we distributed the ultrasounds in a way illustrated in the figure to minimize the noise and draw the maximum amount of information about the external environment. The information collected from the robot at time t is stored in a vector:

$$D_t = [d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8]\, t \tag{3}$$

- di the measurement of sensor i with i belongs $[1, \ldots, 8]$.

You will find our mobile robot model equipped with multiple ultrasonic sensors distributed as illustrated in the Figure 1. With this information, the robot will not have the ability to navigate toward the goal since it simply does not know the goal's location or estimate the remaining distance between it and the goal. Based on this information. as shown in Figure 2, we introduced three additional pieces of information, the goal region GR = $[0, \ldots, 7]$, which sensor detects the goal DG= $[1, \ldots, 8]$ and the remaining distance between our robot and the target DR.

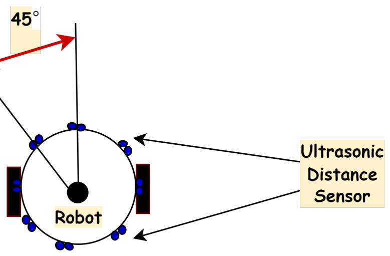


Figure 1. Model of the mobile robot used

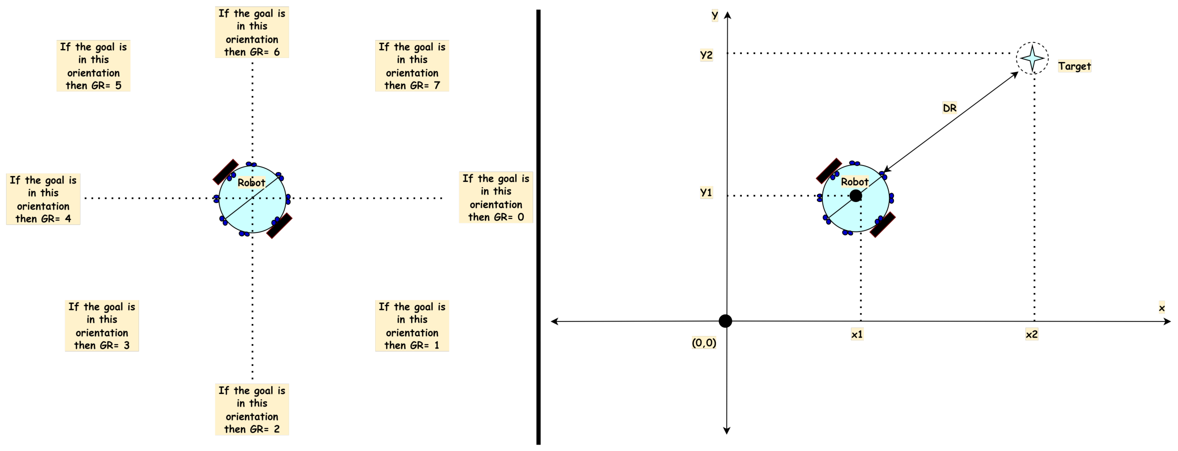

If the goal is in this orientation then GR= 6
If the goal is in this orientation then GR= 5
If the goal is in this orientation then GR= 7
If the goal is in this orientation then GR= 4
Robot
If the goal is in this orientation then GR= 0
If the goal is in this orientation then GR= 3
If the goal is in this orientation then GR= 1
If the goal is in this orientation then GR= 2
Y
Y2
Target
DR
Y1
Robot
(0,0)
x1
x2
x

Figure 2. Mobile robot and target

The Euclidean distance is calculated by (4):

$$DR = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4}$$

The representation of each state at time t is described as (5):

$$\mathrm{X}_t = \begin{bmatrix} D_t \\ GR_t \\ DG_t \\ DR_t \end{bmatrix} \in \mathbb{R}^{11*1} \tag{5}$$

The first eight components are the values returned by the ultrasonic sensors, the ninth component is the target area, and the next component indicates which sensor has been able to detect the target, if no sensor has managed to detect it, it returns the default value of 0, and the last component is the remaining distance between the robot and the target. The robot's movement or possible actions are discretized into eight actions: move forward, turn right with a -45° angle, turn right with a -90° angle, turn left with a 90° angle, turn left with a 45° angle, turn left with a 135 ∘ angle, turn right with a 135° angle, and move back. To schematize these actions, we constructed a vector Y_t :

$$Y_t = [y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8]^T \text{ Avec } y_k = \begin{cases} 1, & \text{if } k = u_t \\ 0, & \text{elsewhere} \end{cases} \quad \text{for } k = 1, \ldots, 8 \tag{6}$$

3.3. Soft and hard voting technique

A voting ensemble is an ensemble machine learning model that combines predictions from several other models. This is a technique that can be used to improve model performance, ideally achieving better performance than any single model used in the ensemble. The Figure 3 shows the difference between the two techniques, namely the soft and hard voting techniques, now let's assume that our generated models can predict only two actions right and left, we have three techniques that suggest the right action versus the left action, hard voting technique chooses the most redundant action, but the soft voting technique sums the probability of each action.

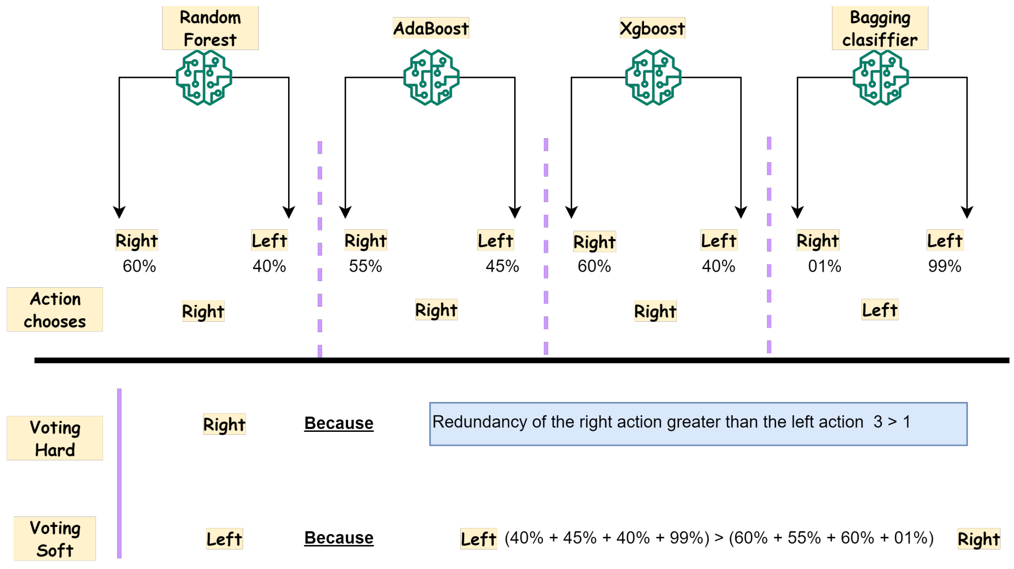


Figure 3. Difference between voting hard and soft

4. CONTRIBUTION

With these four techniques, we can calculate the possible combinations without taking into account the order. As illustrated in the Table 1, we note NTC the number of possible combinations using these four techniques.

$$\text{NTC} = C_4^2 + C_4^3 + C_4^4 = 6 + 4 + 1 = 11$$

Table 1 highlights the generation of 11 combinations. We will apply these combinations for the two techniques we have just mentioned, that is to say, 11 possible combinations for the hard voting technique and the same number for the soft voting technique. The Figure 4 illustrates the organizational chart of our solution, starting with dataset collection, preprocessing, model training, hybrid model generation, and testing in the test environments.

Table 1. Possible combinations

Combinations	Names of the algorithms used	Abbreviation
RF-XGB	Random forest + XgBoost	C_1
RF-ADA	Random forest + AdaBoost	C_2
RF-BAC	Random forest + BaggingClassifier	C_3
XGB-ADA	XgBoost + AdaBoost	C_4
XGB-BAC	XgBoost + BaggingClassifier	C_5
ADA-BAC	AdaBoost + BaggingClassifier	C_6
RF-XGB-ADA	Random forest + XgBoost + AdaBoost	C_7
RF-XGB-BAC	Random forest + XgBoost + BaggingClassifier	C_8
RF-ADA-BAC	Random forest + AdaBoost + BaggingClassifier	C_9
XGB-ADA-BAC	XgBoost + AdaBoost + BaggingClassifier	C_10
RF-ADA-BAC-XGB	Random forest + AdaBoost + BaggingClassifier + XgBoost	C_11

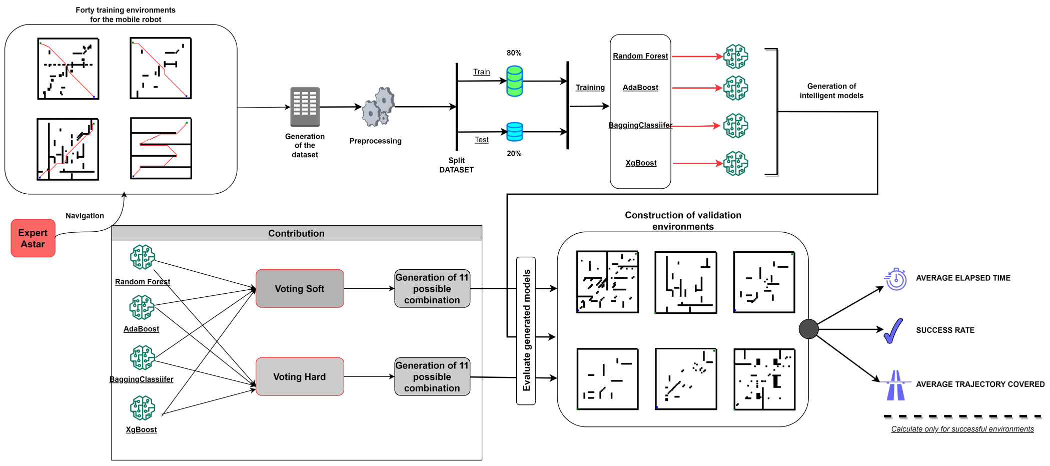

Forty training environments for the mobile robot
Generation of the dataset
Preprocessing
Train
80%
Test
20%
Split DATASET
Training
Random Forest
AdaBoost
BaggingClassiifer
XgBoost
Generation of intelligent models
Expert Astar
Navigation
Contribution
Random Forest
AdaBoost
BaggingClassiifer
XgBoost
Voting Soft
Voting Hard
Generation of 11 possible combination
Generation of 11 possible combination
Evaluate generated models
Construction of validation environments
AVERAGE ELAPSED TIME
SUCCESS RATE
AVERAGE TRAJECTORY COVERED
Calculate only for successful environments

Figure 4. Flow chart of the proposed method

5. EXPERIENCES

To validate the performance of the method proposed in this article, several experiments were done in environments of different complexity. We chose three levels of complexity: easy, medium, and difficult. Each category contains 13 environments, plus one environment that contains no obstacles. So in total, we have 40 environments to validate the intelligent models, knowing that all environments have a size of 53x53 pixels. To cover the maximum possible cases of validation of our models, we choose two long paths between the departure and arrival and in two different places in each environment, the starting point and the target point are respectively (2, 2), (49, 2) and (48,48), (3,50). The simulations are run on a PC with an Intel(R) Core(TM) i3-2348M CPU @2.30 GHz and 8 GB of internal RAM, the code was written in python. The Figure 5 shows some examples of validation environments. These environments are divided into three categories: The image belongs to the first category where the obstacles are not numerous and of small shape, the second category contains obstacles of a little different shape, but the last category contains a lot of obstacles and also dead ends in several places.

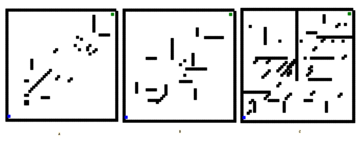


Figure 5. Example of validation environments

5.1. Performance criteria for the training phase

In conducting this evaluation we followed the established practices, in the field of classification. Our focus was on performance criteria. We measured the effectiveness of our methods using four widely recognized metrics; accuracy, precision, recall, and F1 score. We chose these metrics based on recommendations from studies in the field to ensure an rigorous assessment of our approach. By considering these metrics our goal was to provide an unbiased analysis of how our techniques perform in classifying and categorizing data. This analysis offers insights, into their applicability.

5.2. Performance criteria for the validation phase

We evaluated the models generated by three essential criteria that is the success rate in the environments noted.

– TR is the average of successful environments.
– TE is the average elapsed time for successful environments.
– TP is the average of path traveled for the successful environments in pixel.

$$TR = ((80 - B)/80) * 100 \quad (7)$$

With B is the number of environments that failed to reach the goal.

$$\mathrm{TE} = (\mathrm{BL}/\mathrm{NE}) \tag{8}$$

With BL the amount of time needed to navigate in the successful environments. With NE the number of the environment that succeeded the robot in reaching the goal.

$$\mathrm{TP} = (\mathrm{PL}/\mathrm{NE}) \tag{9}$$

With PL sum of pixel to browse the environments that managed the robot to reach the goal.

6. RESULTS AND DISCUSSION

In this section, we will address the results at different stages, namely training and validation. In the first subsection, we will examine the metrics previously discussed in the models and hybrid models section, along with the time required for model training. In the second subsection, we will perform the evaluation of the models obtained in validation environments seen for the first time.

6.1. Training result

After training the models and generating combinations, we calculated the metrics mentioned in the previous section, and we obtained results. The Figure 6 shows the four metrics calculated during the training phase. The separate line shows the average accuracy for the four models, and as you can see the two techniques, random forest and Xgboost outperformed the average. The random forest technique reached up to 95.4% in accuracy, and Xgboost reached 95.2%. These two techniques managed to classify the majority of the actions they should do, unlike the techniques AdaBoost and bagging classifier, which failed to classify most of the actions. These results can be explained by the fact that the random forest technique exploits the strength of the decision tree algorithm and xgboost processes the data sequentially.

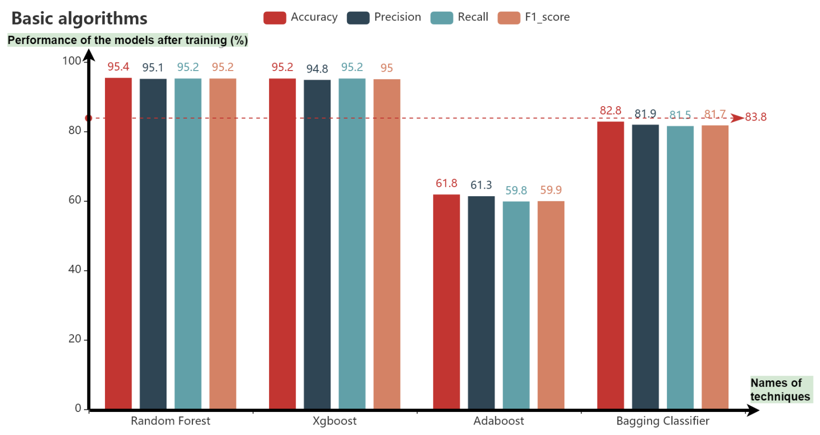


Figure 6. Metrics for the training phase

For both soft and hard voting techniques, we focus only on two metrics, namely precision and the F1 score. We are limited by these two metrics because they give us an overall view of the model's performance, and the F1 score includes all of the other metrics. The Figure 7 shows the separate line indicating the average precision for the 11 combinations to generate. The voting soft technique: this technique with the most generating combinations succeeded in obtaining very good results, as you can see that the average of the precisions exceeded (94%). The first score was for the combination of random forest with xgboost (96%), and the last score was for the combination of AdaBoost and bagging classifier (82%). These results show that the combinations containing either random forest or xgboost always succeeded in classifying the majority of actions. Combinations that failed to exceed the precision average contained at least one technique from either AdaBoost or the bagging classifier. The voting hard technique: this method failed to classify the majority of stocks, as you can see that the average precision did not exceed 89%. The first score was for the combination of random forest, Xgboost, and Adaboost (95.6%) and the last score was for the combination of AdaBoost and bagging classifier (73.5%). These results show that the combinations that contain the two techniques, AdaBoost and bagging classifier, obtain unsatisfactory results.

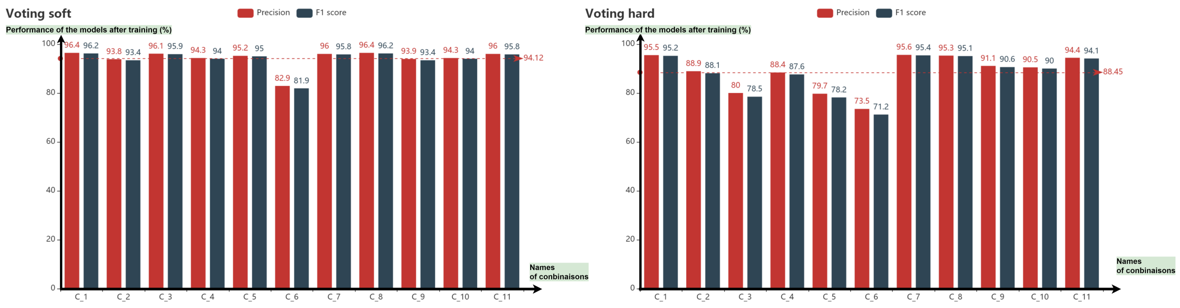


Figure 7. Metrics for the training phase

The Figure 8 shows the time the different techniques took during the training phase. Starting with the four basic techniques, the random forest did not take much time almost 2.36 s. On the other hand, the other techniques took a considerable time. The combinations generated, especially those that contain the technique bagging classifier always take a considerable time for training because the bagging technique uses several samples in parallel to form models of average performance.

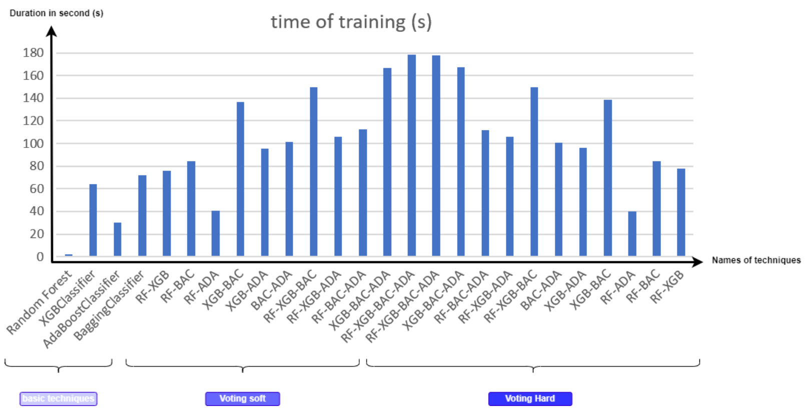


Figure 8. Time of training phase (s)

6.2. Result validation

Taking into consideration the metrics discussed in the previous section, the Table 2 illustrates the four basic algorithms used. As you can see in the Table 2, the technique succeeded with a percentage of 77.50% in the validation environments, followed by the xgboost technique with 75%. Despite the good score of the bagging technique during the training phase, it failed to navigate in most environments. Taking into account the other metrics, the random forest technique did not take much time in the environments, and the average number of pixels traversed is almost 66 pixels in each environment. On the other hand, the bagging technique needs time, but if we see the average number of pixels, we conclude that the technique generally fails when it approaches the goal.

The majority of the techniques generated have succeeded in going through the totality of the environments except for some hard voting techniques. The Table 3 show that combining random forest with a bagging classifier has succeeded with 8.75% in the validation environments, this percentage can be explained by the weakness of the bagging technique that has not succeeded in classifying the actions efficiently. The first score is for the two techniques voting hard (RF-XGB and RF-XGB-BAC-ADA) with 87.5%, these two techniques have succeeded with a high percentage even if in environments seen for the first time.

Table 2. Validation results for the four basic techniques

	XgBoost	AdaBoost	BaggingClassifier	Random Forest
TR	75%	71.25%	3.75%	77.5%
TE	0.45	0.89	1.54	0.27
TP	60.16	77.80	57	65.7

Table 3. Validation results for combinations

		Voting Hard			Voting Soft	
Combinaison	TR (%)	TE	TP	TR (%)	TE	TP
RF-ADA	82.50	1.76	81.34	83.75	1.52	68.77
RF-BAC	8.75	1.72	58.85	40	2.44	59.96
RF-XGB	87.50	1.82	71.98	85	1.7	66.7
XGB-ADA	78.75	2.19	74.52	75	1.8	60.16
XGB-BAC	6.25	2.42	58.2	62.5	3.04	58.84
ADA-BAC	8.75	3.02	61	40	2.6	65.11
RF-BAC-ADA	38.75	3.18	65.19	51.25	3.00	60.04
RF-XGB-ADA	85	3.05	64.45	82.5	3.18	67.69
RF-XGB-BAC	63.75	4.08	59.96	78.75	4.07	57.88
XGB-BAC-ADA	37.5	4.21	60.1	62.5	4.28	58.7
RF-XGB-BAC-ADA	87.5	5.98	66.02	81.25	5.37	58.84

The only difference between these two techniques is the other two metrics, as you will notice the first technique voting hard RF-XGB didn't take much time in the environments and succeeded in almost two seconds, but the second technique took almost 6 seconds because the combination of four techniques, which will take time to predict an action. For the last metric, the second technique generates an almost optimal trajectory of 66 pixels on average, and the first technique is almost 78 pixels on average. The Figure 9 shows some examples of navigating some generated patterns you see that most model don't hit obstacles but instead fall into loops at dead ends.

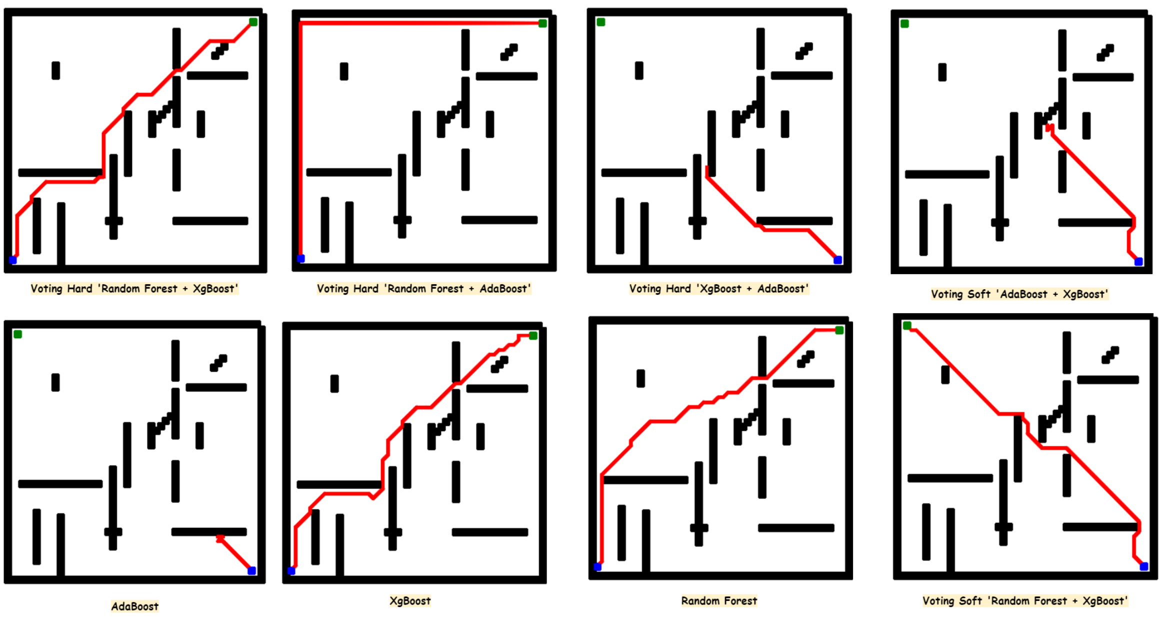

Voting Hard 'Random Forest + XgBoost'
Voting Hard 'Random Forest + AdaBoost'
Voting Hard 'XgBoost + AdaBoost'
Voting Soft 'AdaBoost + XgBoost'
AdaBoost
XgBoost
Random Forest
Voting Soft 'Random Forest + XgBoost'

Figure 9. Navigation examples of some generated technique

7. CONCLUSION AND FUTURE WORK

In this paper, a new approach is proposed to solve several problems and automate the autonomous navigation of mobile robots. Our motivation is to make the robot autonomous and robust in challenging situations and sometimes for users who do not have programming skills. The approach is based on the exploitation and use of learning by demonstration, a field that has attracted more interest in recent years. We tried to minimize the navigation process and make the robot intelligent in unexpected situations. The Hard voting technique, if we combine random forest with Xgboost or all four algorithms, has shown satisfactory results with the same success rate of 87.5%. The only difference is in the other two metrics, namely, the average time to travel and the middle pixel traveled. In our next work, we will try to merge the scalar data with the visual data since the latter exploits the camera sensor, which is characterized by its low cost, and we can also get more information about the environment.

REFERENCES

[1] T.-D. Do, M.-T. Duong, Q.-V. Dang, and M.-H. Le, "Real-time self-driving car navigation using deep neural network," *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, 2018, pp. 7-12, doi: 10.1109/GTSD.2018.8595590.

[2] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 1–22, 2020, doi: 10.1109/TITS.2019.2962338.

[3] A. Kumar, T. Saini, P. B. Pandey, A. Agarwal, A. Agrawal, and B. Agarwal, "Vision-based outdoor navigation of self-driving car using lane detection," *International Journal of Information Technology*, vol. 14, pp. 215–227, Aug. 2021, doi: 10.1007/s41870-021-00747-2.

[4] I. B. Saksvik, A. Alcocer, and V. Hassani, "A deep learning approach to dead-reckoning navigation for autonomous underwater vehicles with limited sensor payloads," *arXiv-Computer Science*, pp. 1-9, 2021.

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001, doi: 10.1177/02783640122067453.

[6] A. Jawad, J. Tarik, W. Abderrahim, S. Idrissi, E. B. Meryem, and S. Ihssane, "A new sampling strategy to improve the performance of mobile robot path planning algorithms," *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, May 2022, pp. 1–7, doi: 10.1109/iscv54655.2022.9806128.

[7] C. Lamini, S. Benhlima, and A. Elbekri, "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180–189, 2018, doi: 10.1016/j.procs.2018.01.113.

[8] C. Lamini, Y. Fathi, and S. Benhlima, "Collaborative Q-learning path planning for autonomous robots based on holonic multi-agent system," *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, Rabat, Morocco, 2015, pp. 1-6, doi: 10.1109/SITA.2015.7358432.

[9] T. A. Teli and M. A. Wani, "A fuzzy based local minima avoidance path planning in autonomous robots," *International Journal of Information Technology*, vol. 13, no. 1, pp. 33–40, Feb. 2021, doi: 10.1007/s41870-020-00547-0.

[10] L. Petrović, "Motion planning in high-dimensional spaces," *arXiv-Computer Science*, pp. 1-6, 2018.

[11] A. Waga, C. Lamini, S. Benhlima and A. Bekri, "Fuzzy logic obstacle avoidance by a NAO robot in unknown environment," *2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS)*, Fez, Morocco, 2021, pp. 1-7, doi: 10.1109/ICDS53782.2021.9626718.

[12] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," *Springer Handbook of Robotics*, pp. 1371–1394, 2008, doi: 10.1007/978-3-540-30301-5_60.

[13] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, May 2020, doi: 10.1146/annurev-control-100819-063206.

[14] M. Wigness, J. A. Rogers, and L. E. Navarro-Serment, "Robot navigation from human demonstration: learning control behaviors," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1150-1157, doi: 10.1109/icra.2018.8462900.

[15] B. C.-Tondreau, G. Warnell, E. Stump, K. Kochersberger, and N. R. Waytowich, "Improving autonomous robotic navigation using imitation learning," *Frontiers in Robotics and AI*, vol. 8, Jun. 2021, doi: 10.3389/frobt.2021.627730.

[16] A. G. E., K. Nazari, H. Hashempour, and F. Zhong, "Deep-LfD: Deep robot learning from demonstrations," *Software Impacts*, vol. 9, Aug. 2021, doi: 10.1016/j.simpa.2021.100087.

[17] M. A. Hussein, Y. Mohammad, and S. A. Ali, "COLD: A ROS package for continuous learning from demonstration teaching a robot to write," *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug. 2017, pp. 651-657, doi: 10.1109/icma.2017.8015893.

[18] L. E. Kavraki, P. Svestka, J.-C. . Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996, doi: 10.1109/70.508439.

[19] N. A. Zainuddin, Y. M. Mustafah, Y. A. M. Shawgi, and N. Khair, "Autonomous navigation of mobile robot using kinect sensor," *2014 International Conference on Computer and Communication Engineering*, Sep. 2014, pp. 28-31, doi: 10.1109/iccce.2014.21.

[20] K. Wu, W. Han, M. A. Esfahani, and S. Yuan, "Learn to navigate autonomously through deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 5342–5352, 2022, doi: 10.1109/tie.2021.3078353.

[21] I. M. A.-Adwan, "Intelligent path planning approach for autonomous mobile robot," *Intelligent Path Planning Approach for Autonomous Mobile Robot*, vol. 33, no. 6, pp. 1423-1428, doi: 10.20965/jrm.2021.p1423.

[22] M. Arduengo, A. Colomé, J. L.-Prat, L. Sentis, and C. Torras, "Gaussian-process-based robot learning from demonstration," *Journal of Ambient Intelligence and Humanized Computing*, Feb. 2023, doi: 10.1007/s12652-023-04551-7.

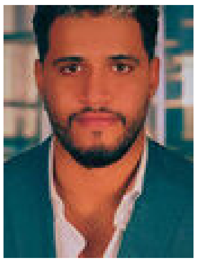
[23] C. Xia, "Intelligent mobile robot learning in autonomous navigation," Ph.D. Thesis, Department of Engineering of Beihang University, École Centrale de Lille, France, 2023. Accessed: Jul. 15, 2023. [Online]. Available: https://core.ac.uk/download/pdf/46808467.pdf

[24] N. P.-Higueras, F. Caballero, and L. Merino, "Teaching robot navigation behaviors to optimal RRT planners," *International Journal of Social Robotics*, vol. 10, no. 2, pp. 235–249, Nov. 2017, doi: 10.1007/s12369-017-0448-1.

[25] D. Garcia, E. Gorrostieta, E. V. Soto, C. R. Rivero, and G. D. Delgado, "Learning from demonstration with gaussian process approach for an omni-directional mobile robot," *IEEE Latin America Transactions*, vol. 16, no. 4, pp. 1250–1255, Apr. 2018, doi: 10.1109/tla.2018.8362164.
[26] C.-Y. Tsai, H. Nisar, and Y.-C. Hu, "Mapless LiDAR navigation control of wheeled mobile robots based on deep imitation learning," *IEEE Access*, vol. 9, pp. 117527–117541, 2021, doi: 10.1109/ACCESS.2021.3107041.
[27] Md. K. Hasan, Md. A. Alam, D. Das, E. Hossain, and M. Hasan, "Diabetes prediction using ensembling of different machine learning classifiers," *IEEE Access*, vol. 8, pp. 76516–76531, 2020, doi: 10.1109/access.2020.2989857.
[28] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992, doi: 10.1109/21.155943.

BIOGRAPHIES OF AUTHORS

Abderrahim Waga obtained his baccalaureate in mathematical sciences and after, he started the Faculty of Sciences ibn Tofail in Kenitra to follow a fundamental license course in mathematics and computer science. The next destination will be the Moulay Ismaïl Faculty of Sciences in Meknes to follow the master's course in computer networks and embedded systems. He was enrolled in doctoral training within the same faculty in order to study the problem of navigation of mobile robots using deep learning techniques. He can be contacted at email: a.waga@edu.umi.ac.ma.



Ayoub Ba-Ichou is a Ph.D. student in the Faculty of Science at Moulay Ismail University. He earned his master's degree in bioinformatics from the Faculty of Science in Meknes in 2020. His primary research interests lie in the fields of bioinformatics and intelligent systems. He can be contacted at email: b.ayoub.info@gmail.com.

Said Benhlima has received Ph.D. in Computer, Science and Robotics from ENSAM Paris in 1993. He has been responsible for several projects and courses in Computer Science and Robotics. He is a full professor with the Moulay Ismail University. He can be contacted at email: s.benhlima@umi.ac.ma.

Ali Bekri received his Ph.D. in science from The Katholieke Universiteit of Leuven in 1998. He worked as a research assistant at the Catholic University of Louvain. He is currently a professor at the Moulay Ismail University. His main research interests are bioinformatics, intelligent systems and their applications. He can be contacted at email: a.bekri@umi.ac.ma.

Jawad Abdouni is a Ph.D. student at the Advance Systems Engineering Laboratory of the National School of Applied Sciences of Ibn Tofail University. He obtained his engineering degree in electromechanics from the Ecole Nationale Supérieure des Mines in Rabat in 2017. His field of research mainly focused on trajectory planning algorithms in autonomous navigation systems. He can be contacted at email: j.abdouni@enim.ac.ma.