COMPARISON OF CLASSIFICATION TECHNIQUES FOR WALL FOLLOWING ROBOT NAVIGATION AND IMPROVEMENTS TO THE KNN ALGORITHM

Sarah Madi and Riadh Baba-Ali

LRPE, USTHB, BP 32 El Alia, Bab Ezzouar Algiers, 16111, Algeria

ABSTRACT

Autonomous navigation is an important feature that allows the robot to move independently from a point to another without a tele-operator. This feature makes mobile robots useful in many tasks that require transportation, exploration, surveillance, guidance, inspection ...etc. Furthermore, autonomous robots deal with real time environments that tend to be complex, non-linear and partially observed. They also operate with limited memory resources and tight time constraints. In this paper, we present an investigation related to mobile robot navigation. We first compare a group of classification algorithms using real traces of wall following robot navigation. Then we focus on the k Nearest Neighbors (KNN) algorithm to improve it and help it be more applicable in autonomous robot navigation. We applied a Nearest Neighbor set reduction technique to help reduce the high running time of KNN. The results indicate that KNN is a competing algorithm especially after decreasing the running time significantly by a factor of 19 and combining that with the KNN's existing features. Results are further improved by applying an attribute selection method.

KEYWORDS

Machine learning, wall following Robot navigation, CNN, Supervised Learning, KNN.

1. INTRODUCTION

When the complexity and non-linearity increase in a mobile robot system; it becomes important to start thinking away from traditional control methods for system representation and decision taking. According to [1], many traditional techniques have failed to address the above-mentioned problems due to the non-linearity and the dynamic characteristics of the task. One can say that in order to control the actions of amobile robot; means of supervised learning algorithms are needed [1]. Recently, methods to gain knowledge directly from data captured have gained more interest. They allow straightforward and accurate system modeling in addition to the avoidance of pre-programming of all possible scenarios. This is more useful especially when a mobile robot is present in unstructured and uncertain environments[2]. One of the common tasks a mobile robot should learn to be autonomous is wall following. The robot should follow a wall with all its curvatures, corners and turns while keeping a fixed distance to the wall whether it is to the left or the right of the robot. It lays the groundwork for more complex problem domains, such as maze solving, mapping, and full coverage navigation (i.e., vacuuming and lawn mowing applications)[3].

Intelligent control is the branch of control where the designer inputs the system targeted behavior and then the intelligent control system provides the knowledge based on certain rules related to the choice of the method [4]. Usually, branches of artificial intelligence are the main inspiration to intelligent control as will be discussed later. Intelligent control systems shine 'in areas that are highly non-linear, or when classical control systems fail, or the representation of the system is difficult or impossible to obtain' [4]. This discussion about converting data or experience into knowledge or a computer program that perform some tasks is known as machine learning. In machine learning technologies, control strategies are emergent rather than predetermined [3].

Learning systems can be categorized based on their feedback into three main categories; supervised learning, unsupervised learning and reinforcement learning. In supervised learning, algorithms use labeled examples that consist of inputs along with their desired output for training. Algorithms learn by comparing actual outputs with correct outputs to find errors and modify the knowledge gained accordingly. The structure that is discovered by studying the relationship between inputs and outputs is referred to as a model. Two main supervised learning models are mentioned in the literature; classification models and regression models [5].

Applying classification techniques is successful and promising in the field of robotics, however, mobile robots tend to have limited memory resources that cannot hold the required big training files to classify and obtain the next action to be taken. Additionally, autonomous navigation has tight time constraints that has to be respected in order to have a smooth and a practical operation. A mobile robot should take the appropriate action, given the current situation before arriving at a new case keeping in consideration the sampling rate of the attached sensors. In this work, we focus mainly on the KNN algorithm in the field of autonomous robots; specifically in wall following applications and we proposed the following contributions:

- We compared some of the main classification techniques using real traces of wall following robot navigation datasets in order to highlight the main advantages and disadvantages.
- We addressed the main disadvantages of KNN algorithm using two important techniques; set reduction and attribute selection.
- We implemented the Condensed Nearest Neighbor set reduction technique using Java and then successfully used it to reduce the training files; hence, less running time better accuracy.
- Further improvements are witnessed by combining both set reduction and attribute selection techniques.

This paper is organized as follows: Section 2 contains a brief explanation about the main algorithms used in this research. Section 3 presents the related work. Section 4 describesthe wall following problem in addition to the datasets used in this paper and how it was collected. The subsequent section discusses the results obtained and the improvements applied to enhance the results. Finally, section 6 concludes the paper.

2. BACKGROUND

In this section, the algorithms that are deployed in this research are briefly explained. The algorithms presented next are related to classification methods and include: decision trees, Neural Networks, Naïve bayes, JRipper, Support Vector Machines and k-Nearest Neighbors.

2.1. Decision Trees Induction C4.5

This is one of the most important supervised learning techniques that is presented in the shape of branches, nodes and leafs and decisions are taken from the root of the tree to the leaf. At the end of each branch is a leaf that presents the result obtained. Intermediate nodes in the tree contains a test on a particular attribute that distribute data points in the different sub-trees. The objective from the learning phase is to identify groups of examples as consistent as possible while in the test phase the new example is assigned to the majority class of the leaf based on a score that corresponds to the proportion of training examples in the leaf that belong to the same class [6].

Decision trees are widely used because of their flexibility and applicability to wide range of problems. The resulting set of rules or tree paths are mutually exclusive and exhaustive i.e. every instance is only covered by a single rule [7] .

2.2. Neural Networks

A Neural Network is a computation algorithm inspired by the structure of neural networks in the brain, where it consists of a large number of computing devices /nodes /neurons connected to each other through links to consist a network. Each neuron will receive a weighted sum of the neurons outputs connected to its incoming links [8]. The weights can take new values through the learning process up until 'the optimal weights are achieved and stored as strengths of the neurons interconnections'. The main features of Neural Networks are their ability to utilize a large amount of sensory information, their capability of collective processing and finally their ability to learn and adapt to changes and new information. According to [9] the two most Neural Networks types suitable for decision and control purposes are the multilayer perceptron and radial basis functions.

2.3. Support Vector Machines

Support vector Machines are a set of supervised learning methods that are applied in linear and non-linear classification problems, where they build a model that assigns new examples to one category or the other. They work by 'mapping the input vectors into a high dimensional feature space and constructing the optimal separating hyper-plane through structural risk minimization'

[10]. An SVM model is a representation of the examples as points in the space such as they are divided by a clear gap as wide as possible. As [6] explains, 'the main idea is to maximize the distance between the separating hyper-plane and the closest training example'. After that, new examples are mapped to the same space where it is predicted that they belong to the gap they fall in [11]. According to [7] three main benefits are associated with SVMs. First, they are effective in high dimensional space. Second, they are memory efficient as they use a subset of training point in the support vectors or the decision functions. Finally, they are considered versatile since they can hold different kernel functions to be specified as decision functions. Some of the popular kernel functions are linear, polynomial and RBF. On the other hand, SVMs inherently do binary classification; which means they only support two-class-problems where real life problems usually have more than two classes. Other procedures can be used to extend them to multiclass problems such as one versus the rest and one versus one approaches [12].

2.4. JRipper

JRipper is a supervised learning rule based learning algorithm that stands for Repeated Incremental Pruning to Produce Error Reduction. It builds a set of rules that identify the classes, keeping in mind minimizing the error. It consists of two stages: the building stage and the optimization stage. Some references such as [13] add a third stage which is the delete stage. In this algorithm, the training set is split into a growing set and a pruning set, where at first, an initial rule set is formed over the growing set using some heuristic method (building stage). This overlarge initial rule set is repetitively simplified using one of the pruning operators such as deleting single conditions or any single rule. At each simplification stage, the pruning operator that yields the highest reduction of error is chosen. The process continuous until applying any pruning operator increases the error on the pruning set (optimization stage) [14]. Another simple introduction to JRipperis given in [15] where they mention that all examples at start are treated as a class, and rules that cover all members of that class are found. After that, the algorithm moves to the class and does the same until all classes are covered. According to the creator of this algorithm, it was designed to be fast and effective when dealing with large and noisy datasets compared to decision trees [16].

2.5. Naïve Bayes

The Naive Bayes classifier is a classical demonstration of how generative assumptions and parameter estimations simplify the learning process [8]. It assumes that the explanatory variables are conditionally independent to the target variable. This assumption contributes to reducing the training time complexity and helps the algorithm compete on numerous application areas. The performance of this classifier depends 'on the quality of the estimate of the univariate conditional distributions and on an efficient selection of the informative explicative variables'. This approach as other approaches has some advantages such as its low training and prediction time complexity in addition to its low variance. It shines in classification problems where only few training examples are available [6].

2.6. K Nearest Neighbors

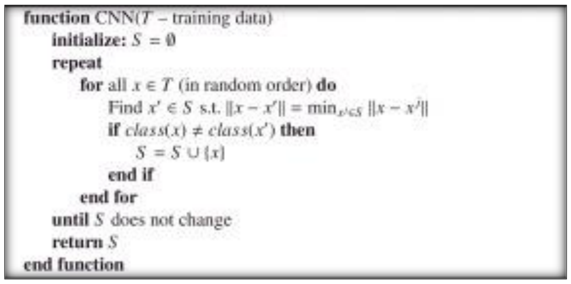
K Nearest neighbors (KNN) is one of the oldest and most widely used methods for object classification. According to [7] they are of benefit when there is little or no prior knowledge about the distribution of the data points. The nearest neighbors are calculated using a type of distance functions. The value of k specifies how many nearest neighbors are to be considered to define the class of a sample data point or a query. As [17] mentions in a comparison table between all nearest neighbors algorithms; KNN has some main advantages such as: a) fast training time, b) easy to learn c) high algorithm simplicity, d) robust to noisy training data points and finally, e) effective if the training dataset is large. KNN is a powerful classification method that allows the classification of an unknown instance using a set of classified training instances. The authors of [18] mentioned that KNN is one of the simplest classifier methods that is able to solve certain complexities such as time and computational complexities. It has the ability to memorize training data and process it during run time. However, the process of computing distances is computationally high especially in large datasets, which leads to memory limitations.

2.6.1. Set Reduction and Condensed Nearest Neighbors (CNN)

To overcome some of the disadvantages related to the KNN algorithm, we propose using a set reduction technique. Set reduction is a preprocessing technique that allow reducing the size of the dataset. Its main objective is to reduce the original dataset by selecting the most representative data points. This way, it is possible to avoid excessive storage of data points and excessive running time for supervised classification [19].

A nearest neighbor technique that can be used for set reduction is called the Condensed Nearest Neighbor (CNN) that stores the data points one by one then eliminates the duplicate, noisy and redundant ones. In other words, it removes the data points that do not add more knowledge and show similarity with other training data points [17]. This way, the size of the training dataset will decrease, the query time and memory requirements will improve which leads to system improvements.

This technique is based on finding a subset S of the whole training file T such that it is small and accurate in classifying T. through the process of finding S, it must assure that all examples in T are correctly classified regardless of it being the minimal subset. The pseudo-code of CNN is presented in figure (1). This subset S could be used instead of the original file T without major losses in performance. CNN inspired researchers to develop more set reduction techniques that help reducing data as much as possible [20]..



```
function CNN($T$ – training data)
    initialize: $S = \emptyset$
    repeat
        for all $x \in T$ (in random order) do
            Find $x' \in S$ s.t. $\|x - x'\| = \min_{x' \in S} \|x - x'\|$
            if $class(x) \neq class(x')$ then
                $S = S \cup \{x\}$
            end if
        end for
    until $S$ does not change
    return $S$
end function
```

Figure 1. CNN pseudo-code [20]

3. RELATED WORK

In this section, the recent work related to combining machine learning with autonomous mobile robots will be listed with a brief description.

The work of [21] used the nearest neighbor classification and both regression and support vector regression for their wall following robot. The authors of [22] present an incremental decision tree algorithm that is applied in mobile robot navigation. Their algorithm is fast, memory efficient and was tested and compared to previous similar work for evaluation purposes.

The authors of [23] present an application of SVM in robot navigation, in particular, an obstacle avoidance algorithm that do some learning through an SVM classifier. Experimental and simulation results are presented to validate their model.

The work of [24] attempted to develop a neural network based controller for a wall following robot. Their primary focus was to control the robot to take decisions of changing direction based on sensor readings dataset. The experimental result shows that the proposed algorithm can control the robot with 92.67% accuracy and can take decision within one second. The same group again proposed the work in [1] and attempted to use a new neural network training algorithm based on gravitational search and then a feed forward neural network for automatic robot navigation of wall following robots.

4. PROBLEM AND DATA SET DESCRIPTION

Wall Following is a task where a mobile robot should follow a wall with all its curvatures, corners and turns while keeping a fixed distance to the wall to the left or the right of the robot (figure 2). It is usually combined with obstacle avoidance for the robot to achieve its task with no collisions, which may affect the performance. Robots are usually equipped with two sensors; one in the front and the other to side of the wall to be followed. Wall following is usually selected to test new or existing algorithms since it is a fairly simple problem to set up and evaluate and complex enough to represent a non-linear control problem [21].

For a learning algorithm to be more practical, we mention some of the important points the authors of [25] discussed and concern us in this study. Since the decision the robot should take will affect the navigation and manipulation tasks, processing should happen fast almost in real time. Additionally, mobile robots have limited memories that usually cannot hold the required training files especially with algorithms that require huge dataset to get the better model. Even if the memory is sufficient to hold the required training set, the time required to train the model and process the training set will still be an issue giving the time constraints.

As a result, and in order for us to study each of these classification algorithms under the same conditions, a data set related to robot navigation is chosen as a benchmark as will be discussed shortly. The k nearest neighbor technique will be the focus as we will study its competence with other algorithms and prove its great usability.

The authors of [26] prepared their own data by guiding the robot through a certain algorithm. A C++ routine was implemented to guide the behavior of following walls in a known situation. The algorithm is responsible for generating the decision the robot should take along its navigation. The information was collected as their robot SCITOS G5 navigate through the room during four rounds. The collection of data points was performed at a rate of nine samples per second.

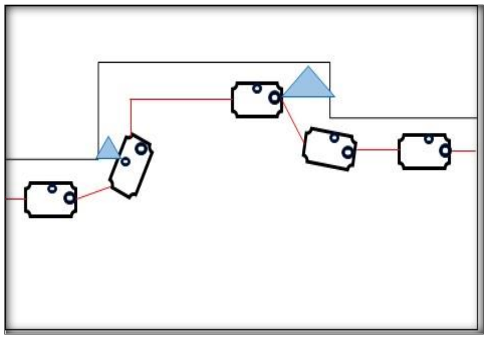


Figure 2. Wall following scenario

Their generated database contain 5456 examples and it has three versions divided in three files. The first file contain sensor information from two sensors as in 2 attributes and one class, the second file contain information from four sensors as in 4 attributes and 1 class and the last file contain sensor information from 24 sensors as in 24 attributes and 1 class. The class contains four robot decisions; move forward, slight right turn, sharp right turn and slight left turn. This dataset is widely used in research that is related to robot navigation, machine learning, and classification algorithms such as [27], [1] and [28]. It is available in the UCI machine learning repository.

Table (1) presents a brief description on the used datasets and examples from each file.

Table 1. Dataset Description And Examples

Dataset	**Number of attributes**	**Number of examples**	**Instance form**
Sensor 2 file	2 attribute, 1 class	5456 instances	1.687,0.449,Slight-Right-Turn
Sensor 4 file	4 attributes, 1 class	5456 instances	1.32,0.499,2.83,0.71,Move-Forward 1.897,1.413,1.981,1.785,Slight-Left-Turn
Sensor 24 file	24 attributes, 1 class	5456 instances	0.481,0.515,5.018,3.664,2.956,2.927, 2.947,2.993,1.697,2.622,1.651,1.641, 1.645,1.269,0.765,0.592,0.489,0.482, 0.495,0.531,0.462,0.499,0.483,0.473,Sharp-Right-Turn

Usually the third file is the most complicated file as it contains many attributes. In order to reach a model/decision/classification, the processing is a bit complicated to include all attributes, especially the ones that contribute to the decision. This complication will be noticed later in the results were the performance will decrease when using this file. The experiments were held on an Intel core i7, 64-bit operating system 2.20 GHz processor and 4GB RAM.

5. RESULTS

The results section will start by presenting the comparison held between the classification algorithms. Then, the main disadvantages and notes will be highlighted. Finally, improvements to the KNN algorithm in focus are applied through set reduction and attribute selection techniques.

5.1.Comparison of Classification Algorithms using WEKA

Table (2) presents the comparison held between the different algorithms introduced earlier using the wall following robot navigation data set. The criterion included in the comparison are the accuracy, training time, and testing time.

Table 2. Comparison between algorithms using weka

File used	Classifier Name / Evaluation Criteria	Decision Trees J48	SMO	JRipper	Neural network	Naïve Bayes	K Nearest Neighbors
Sensor 2 file	Accuracy	100%	77.20%	99.90%	91.83%	90.50%	98.80%
	Training time	0.15	0.32	0.25	9.1	0.06	0.01
	Test time (running time)	0.17	0.25	0.01	0.02	0.12	3.91
Sensor 4 file	Accuracy	100%	77.30%	99.90%	97.47%	89.10%	97.20%
	Training time	0.18	0.59	0.36	14.8	0.02	0.01
	Test time (running time)	0.14	0.15	0.02	0.07	0.13	4.05
Sensor 24 file	Accuracy	99.60%	71.40%	98.80%	87.92%	52.40%	88.17%
	Training time	0.39	9.39	2.46	81.88	0.07	0.01
	Test time (running time)	0.03	0.09	0.02	0.12	0.42	6.88

This table presents the results of running the robot navigation datasets in the WEKA software [29] for comparison purposes. WEKA is a data mining, machine learning tool which was first implemented in the University of Waikato, New Zealand.

As far as accuracy concerns, decision trees, JRip and KNN were the best algorithms to perform in almost all cases. On the other hand, KNN outperformed them all in terms of training time which is a point counted for this algorithm. However, the same algorithm recorded high testing time as expected after [17] discussed some disadvantages for the KNN algorithm such as high running/test time and memory limitations. From this result, the idea of applying the set reduction method using CNN came.

5.2. CNN Implementation and Dataset Reduction

We implemented and tested the CNN algorithm on Java based on the pseudo code presented earlier. The three datasets were successfully reduced using our implemented algorithm. The reduced files were again applied in WEKA to obtain the new results and compare them with the original KNN results. Table (3) presents the development of the KNN algorithm after applying the CNN technique to reduce the training files.

Note that the number of instances after reduction in the first file was 169/5456 with a file reduction rate of 97%, in the second file it was 369/5456 with a file reduction rate of 93% and in the third file, it was 952/5456 with a file reduction rate of 82%. These tremendous file reduction operations resulted in excessive training and test times reduction.

Some points should be noted regarding the performing of these tests. For the original algorithm tests and comparisons in table (2), results were obtained using the cross-validation method. As for the KNN after applying the set reduction with CNN (table (3)); the testing was done using the original files as test files as they will contain more information than the current reduced training file and their results will present an indication for the robot performance in real life after facing new situations.

Table 3.Comparison between knn performances before and after set reduction

File	Evaluation Criteria	K-nearest neighbors	KNN with reduced data set CNN
Sensor 2 file	Accuracy	98.80%	97.58%
	Training time	0.01	0
	Test time (running time)	3.91	0.2
Sensor 4 file	Accuracy	97.20%	96.04%
	Training time	0.01	0
	Test time (running time)	4.05	0.44
Sensor 24 file	Accuracy	88.17%	90.52%
	Training time	0.01	0.01
	Test time (running time)	6.88	2.18

Moving to the results, the reduction in test time can be noticed clearly as the reduction exceeded 90% in the first two files and exceeded 68% in the third file. The accuracy experienced some drop as well but it was very low between 1% and 1.5% in the case of the first two files. On the other hand, the accuracy increased in the case of the third file by 2% which is important.

As for the training time, which was originally excellent, it was either less or the same. The main explanation is related to the reduction in the training files. This reduction led to a simpler file construction, faster training, thus faster testing, classification, learning and performing.

To get a better look for the performance improvements of the KNN algorithm let us calculate the following performance metrics; reduction rate for test time, the resulted increment rate for the KNN performance and the speed up factor using the following equations:

$$Reduction\,rate\ = \left(\frac{oldtime - \ newtime}{oldtime}\right) * \ 100 \qquad (1)$$

$$KNN\ performance\ increment\ rate\ = \left(\frac{oldtime - \ newtime}{newtime}\right) * \ 100 \qquad (2)$$

$$Speed\ factor\ = \frac{oldtime}{newtime} \qquad (3)$$

It can be noticed from table (4), that in case of the 2 sensors file, the reduction witnessed intest/running time, helped increase the performance of KNN with more that 1000% and the algorithm became faster with a speed factor of more than 19 times. This is considered as a huge and very noticeable improvement of the KNN algorithm. The reduction in size for the 4 sensors and 24 sensors files, reduced the test/running time and improved the KNN performance by more than 800% and 200% correspondingly. The KNN algorithm became faster with a speed factor of more than 8 times.

Table 4. Improvements in the performance for the knn algorithm

File	**Test time reduction rate**	**KNN performance increment rate**	**Speed factor**
Sensor 2 file	94.8%	1855%	19 times
Sensor 4 file (after CNN)	89.1%	820%	9 times
Sensor 4 file (After attribute selection)	91.6%	1091%	12 times
Sensor 24 file (after CNN)	68.3%	215%	3 times
Sensor 24 file (after attribute selection)	87.2	682%	8 times

5.3. Attribute Selection for 4 and 24 Sensor Files

To further improve the results related to the multi-attribute data sets, the 4 and 24 sensors datasets; an option in the WEKA software was applied to select the most meaningful attributes from the dataset. This option evaluates the worth of the subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. After that, it will search the space of attribute subsets by greedy hill climbing augmented with a backtracking facility.

In case of the 4 sensors file, two attributes out of four were chosen and the related file was modified to include the two selected attributes plus the class. Table (5), presents the improvements in accuracy and in running /test time. As for the 24 sensors' file, seven attributes out of 24 were selected and the dataset after the CNN reduction was modified to include them in addition to the class. The results improved further as noticed from table (5). The accuracy witnessed an increment by more than 2% compared with the previous step. The training time kept its excellent low levels. As for the test/ running time, it kept on decreasing further and therefore more improvement in the performance. The test time deceased by 0.1 s from the previous step in case of the 4 sensors file and by 1.3 s in case of the 24 sensors file.

The improvement in the performance of the 24 sensors file is remarkable where the overall accuracy between the original training file and reduced training file in both instances and attributes increased by 4 %. The total reduction in test/running time for the same file was 6 s with a speed factor of 8 times. More on the improvements after attribute selection is presented in table (4).

Table 5. KNN results after removing redundant attributes

File	Evaluation Criteria	KNN with original files	KNN with reduced dataset CNN	KNN with CNN and reduced attributes
Sensor 4 file	Accuracy	97.20%	96.04%	98.69%
	Training time	0.01	0	0
	Test time (running time)	4.05	0.44	0.34
Sensor 24 file	Accuracy	88.17%	90.52%	92.57%
	Training time	0.01	0.01	0
	Test time (running time)	6.88	2.18	0.88

6. CONCLUSION

In this paper, we focused on the robot navigation field where real traces of robot navigation datasets were used to test multiple classification algorithms using WEKA. The results showed some limitations in terms of test/ running time in the KNN algorithm. We contributed to enhance the KNN performance by applying further techniques and ideas. We successfully implemented and tested a set reduction technique; the condensed Nearest Neighbor (CNN) algorithm. The KNN algorithm was tested again to notice important improvements in the testing time for all three files and the accuracy in one file. We were able to speed up the KNN algorithm 19 times more with a performance increment rate of more than 1000%. This is due the reduction in training files that reached up to 97% which led to reduction in the test time for up to 94%. To further reduce the testing time and improve the performance, especially in multi-attribute datasets, the files of 4 sensors and 24 sensors were pre-processed again to remove the redundant attributes. Both accuracy and testing time improved using the WEKA's attribute selection feature. We witnessed improvements in the classification accuracy by 4% in case of the sensor 24 file with a test/running time reduction that reached to 87% from 6.88 s to 0.88s.

We conclude that KNN is an old yet still a promising classification algorithm. Moreover, set reduction and attribute selection are complementary and effective tools that are combined with KNN to maintain/improve accuracy and improve testing time and speed. This way, the major drawbacks of KNN will be solved while maintaining the great advantages of fast training time and learning. For future work, we aim at a set reduction technique that combines both instance and attribute selection in one algorithm and apply that on big datasets.

REFERENCES

[1] T. Dash, T. Nayak and R. R. Swain, "Controlling Wall Following Robot Navigation Based on Gravitational Search and Feed Forward Neural Network," in Proceedings of the 2nd International Conference on Perception and Machine Intelligence, Kolkata, 2015.

[2] D. Nguyen-Tuong and J. Peters, "Model Learning for Robot Control: A Survey," cognitive processing, vol. 12, no. 4, pp. 319-340, 2011.

[3] R. A. Dain, "Developing Mobile Robot Wall-Following Algorithms Using Genetic Programming," Applied intelligent, vol. 8, no. 1, pp. 33-41, 1998.

[4] B. Smith, "Classical versus Intelligent Control," 2002. [Online]. Available: https://www.engr.mun.ca/~baxter/Publications/ClassicalvsIntelligentControl.pdf. [Accessed 20 12 2016].

[5] O. Maimon and L. Rokach, "Introduction to Supervised Methods," in Data Mining and Knowledge Discovery Handbook, 2nd ed., Springer US, 2010, pp. 149-164.

[6] V. Lemaire, C. Salperwyck and A. Bondu, "A Survey on Supervised Classification on Data Streams," Lecture Notes in Business Information Processing, pp. 88-125, 16 04 2015.

[7] I. Muhammad and Z. Yan, "SUPERVISED MACHINE LEARNING APPROACHES: A SURVEY," ICTACT JOURNAL ON SOFT COMPUTING, vol. 5, no. 3, pp. 946-952, 2015.

[8] B.-D. Shai and S.-S. Shai, Understanding Machine Learning:From Theory to Algorithms, New York: Cambridge University Press, 2014.

[9] S. G. Tzafestas, Introduction to Mobile Robot Control, Elsevier, 2014.

[10] N. A. Syed, H. Liu and K. K. Sung, "Handling Concept Drift in Incremental Learning with Support Vector Machines," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, California, 1999.

[11] H. Hormozi, E. Hormozi and H. R. Nohooji, "The Classification of the Applicable Machine Learning Methods in Robot Manipulators," International Journal of Machine Learning and Computing, vol. 2, no. 5, pp. 560-563, 2012.

[12] P. X. Huang and R. B. Fisher, "Individual feature selection in each One-versus-One classifier improves multi-class SVM performance," in Proceeding of the International Conference on Pattern Recognition, Stockholm, 2014.

[13] W. Shahzad, S. Asad and M. A. Khan, "Feature subset selection using association rule mining and JRip classifier," International Journal of Physical Sciences, vol. 8, no. 18, pp. 885-896, 2013.

[14] "Data Mining Algorithms In R/Classification/JRip," Mediawiki, 09 01 2016. [Online]. Available: https://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Classification/JRip. [Accessed 12 03 2017].

[15] A. Rajput, R. P. Aharwal , M. Dubey , S. Saxena and M. Raghuvanshi , "J48 and JRIP Rules for E-Governance Data," International Journal of Computer Science and Security (IJCSS), vol. 5, no. 2, pp. 201-207, 2011.

[16] S.Vijayarani, and M.Divya, "An Efficient Algorithm for Generating Classification Rules," International Journal of Computer Sci ence And Technology, vol. 2, no. 4, pp. 512-515, 2011.

[17] N. Bhatia and Vandana, "Survey of Nearest Neighbor Techniques," International Journal of Computer Science ans Information Security, vol. 8, no. 2, pp. 302-305, 2010.

[18] B. Ilias, S. A. Abdul Shukor, A. Adom, N. Abd Rahim, M. F. Ibrahim and S. Yaacob, "Indoor mobile robot localization using KNN," in Proceedings of the 2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Malaysia, 2016.

[19] A. Miloud-Aouidate and A. R. Baba-Ali, "IDS false alarm reduction using an instance selection KNN-memetic algorithm," International Journal of Metaheuristics, vol. 2, no. 4, pp. 333-352, 2013.

[20] S. Garcíaa, . J. Luengoa and F. Herreraa, "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining," Knowledge based systems, vol. 98, pp. 1-29, 2016.

[21] C. Hassall, R. Bhargava and T. Trappenberg, "A Robust Wall-Following Robot That Learns by Example," in Proceeding of the Dalhousie Computer Science In-House Conference (DCSI), Halifax, 2012.

[22] E. Swere, D. Mulvaney and I. Sillitoe, "A Fast Memory-Efficient Incremental Decision Tree Algorithm in its Application to Mobile Robot Navigation," in Proceeding of International Conference on Intelligent Robots and Systems, 2006.

[23] A. Shankar, M. Vatsa and P. Sujit, "Collision avoidance for a low-cost robot using SVM-based monocular vision," in Proceeding of the IEEE International Conference on Robotics and Biomimetics (ROBIO), 2014.

[24] T. Dash, S. R. Sahu, T. Nayak and G. Mishra, "Neural Network Approach to Control Wall-Following Robot Navigation," in Proceedings of the 2014 IEEE International Conference on Advanced Communication Control and Computing Technologies, Ramanathapuram, 2014.

[25] R. E. Karlsen, S. Hunt and G. Witus, "Robot Training Through Incremental Learning," Proceedings of SPIE - The International Society for Optical Engineering 8045, 23 5 2011.

[26] A. L. Freire, G. A. Barreto, M. V. Veloso and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study," in 6th Latin American Conference: Robotics Symposium (LARS), 2009.

[27] M. O. Karakus and O. Er, "Learing of Robot Navigation Tasks By Probabilistic Neural Network," in Proceeding of the Second International Conference on Advanced Information Technologies and Applications , 2013.

[28] S. Madi and A. R. Baba-ali, "Classification Techniques for Wall-Following Robot Navigation: A Comparative Study," in Proceedings of the International Conference on Advanced Intelligent Systems and Informatics AISI2018, Cairo, 2018.

[29] M. L. G. a. t. U. o. Waikato, "Weka 3: Data Mining Software in Java," University of Waikato, [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.