Predicting Vehicles' Positions using Roadside Units: a Machine-Learning Approach

Mamoudou Sangare*, Soumya Banerjee‡, Paul Muhlethaler*, Samia Bouzefrane†
*INRIA EVA, Centre de Recherche de Paris, 2 Rue Simone, IFF CS 42112, 75589 Paris Cedex 12
Email: {paul.muhlethaler}@inria.fr
† CEDRIC Lab, CNAM, 292 Rue Saint-Martin, 75003 Paris, France
Email: samia.bouzefrane@lecnam.net
‡ Department of Computer Science and Engeneering. Birla Institute of Technology, Mesra, India
Email: dr.soumya@ieee.org

***Abstract*—In this paper, we study positioning systems using Vehicular Ad Hoc Networks (VANETs) to predict the position of vehicles. We use the reception power of the packets received by the Road Side Units (RSUs) and sent by the vehicles on the roads. In fact, the reception power is strongly influenced by the distance between a vehicle and a RSU. To predict the position of vehicles in this context, we adopt the machine-learning methodology. As a pre-requisite, the vehicles know their positions and the vehicles send their positions in the packets. The positioning system can thus perform a training sequence and build a model. The system is then able to handle a prediction request. In this request, a vehicle without external positioning will request its position from the neighboring RSUs. The RSUs which receive this request message from the vehicle will know the power at which the message was received and will study the positioning request using the training set. In this study, we use and compare three widely recognized techniques : K Nearest Neighbors (KNN), Support Vector Machine (SVM) and Random Forest. We study these techniques in various configurations and discuss their respective advantages and drawbacks. Our results show that these three techniques provide very good results in terms of position predictions when the error on the transmission power is small.**

I. Introduction

Vehicular Ad Hoc NETworks (VANETs) use Wireless Access in Vehicular Environments (WAVE) [1] to exchange data between the vehicles themselves and the vehicles and the infrastructure. For the access, WAVE is based on the IEEE802.11p access protocol to provide communication between vehicles (V2V), and between vehicles and roadside infrastructure (V2I) in new Intelligent Transportation Systems (ITS).

One of the prominent applications of VANETs communication concerns safety applications; these applications rely on the periodic transmission of packets. There are two main types of applications: Car Awareness Messages (CAMs) [2] and Decentralized Environmental Notification Messages (DENMs) [3]. CAMs are used to periodically send information about the vehicles' velocities and positions, while DENMs are sent when a hazardous event occurs on the road. Another use of VANETs communication concerns the transmission of information or advertising to the vehicles. Information sent to the vehicle's driver can concern the status of the traffic, whereas data sent to the passengers can concern advertising or entertainment, often infotainment.

Another possible application of VANETs is to use them as a positioning system. The safety messages that are periodically broadcasted by the vehicles allow one to build huge data bases with positioning information since in most cases the vehicles know their positions as they are equipped with a Global Positioning System (GPS). Moreover, there is a strong correlation between the position of a vehicle and the signal strength of the messages it exchanges with the roadside units. Therefore when a vehicle can not rely on its GPS to give its position, it can ask the roadside units in its vicinity for an estimation based on the power with which they receive its messages. For instance, we can assume that even though a vehicle does not have any information from its GPS concerning its speed and position, it continues to send its CAMs carrying no (or very little) position information. These messages can be used by the roadside units to establish the vehicle's position. An approach with machine-learning techniques such as: K Nearest Neighbors (KNN), Support Vector Machine (SVM) and Random Forest is thus possible. The aim of this paper is to adapt these three techniques to the positioning in VANETs with roadside units and to study and compare their performances to predict the position of a vehicle on a road when it is within reach of three roadside units. We consider different configurations and compute the prediction error in different situations.

Here, the contributions are as follows:

- We use the reception power to predict a vehicle's position.
- We propose and adapt three learning techniques to the positioning of vehicles : K Nearest Neighbors (KNN), Random Forest and Support Vector Machine (SVM).
- A simple simulation tool is developed to produce data with the positions of vehicles and the different power of messages sent by vehicles and received at the base stations.
- We analyze and compare the performance of k Nearest Neighbors (KNN), Random Forest and Support Vector Machine (SVM) with the given dataset.

The remainder of this paper is organized as follows: Section II reviews related work; Section III describes the three machine-

learning techniques: K Nearest Neighbor (KNN), Support Vector Machine (SVM) and Random Forest. In Section IV, the simulation scenario and numerical results are presented. Different machine learning techniques and their optimizations are discussed. Three methods and their performances are compared. Finally, Section V concludes the paper.

II. Related work

In the field of outdoor positioning systems, we mainly find the following techniques:

- Time-Of-Arrival (TOA)-based techniques. They are based on the measurements of the distance between base station and the receiver of a signal and a triangulation allows the position to be computed. These measurements are not acceptable for VANETs, as the precision required for these measurements are not achievable. These techniques also require a perfect synchronization between the clocks of the base stations and the receivers. This implies the use of atomic clocks, which are very expensive. TOA is precisely the basis of GPS and other similar positioning systems.
- Techniques based on the round trip delay [4]. The receiver sends a small packet to the base station and waits for a reply. The time elapsed is proportional to the distance between the base station and the receiver. By simple triangulation the receiver can compute its position if it can receive three round trip delays from three different base stations. To obtain a precise estimation of the location, the distance between the receiver and the base station must be large. However, like TOA schemes, techniques based on round trip delay require very accurate delay evaluation, which is very difficult if dedicated transmission modems are not used. Thus this technique is generally not suitable for VANETs which use off-the-shelf IEEE 802.11p communication units.
- Signal-strength-based techniques. The receiver computes an estimate of its location based on the received signal strength from several wireless access points. In this case, the access points must be carefully positioned to cover the roads. Reported results based on this technique show poor accuracy [5], [6].

The GPS or other positioning systems are the prominent positioning techniques in vehicular networks [4]. But GPS has three main drawbacks: limited accuracy, incomplete coverage and security problems. It has a limited accuracy that is almost 20 meters. This is really unsuitable for most VANET applications, e.g., lane tracking, collision avoidance, autonomous driving, etc. According to GPS device vendors, the best announced accuracy is plus/minus 5 meters. Moreover, this accuracy is claimed to be reached for only 95% of the cases and for the other 5%, the accuracy may be greatly less. Thus relying on GPS is unacceptable in the critical applications requiring an accurate positioning system. GPS also has an incomplete coverage. The ideal situation is when the GPS signal can be received in vehicles from four different satellites and this is very unlikely to occur everywhere, even if the obvious case of tunnels is left out. GPS simulators (devices generating GPS signals) can create fake signals, these signals are generally sent with a high power. The GPS receiver usually considers the strongest signal and thus locks on the fake signal. Thus, any jammer equipped by this kind of GPS simulator can completely distort the evaluation of the vehicles' positions.

Numerous papers concerning positioning with GPS use an enhancement referred to as the Differential GPS (DGPS). This technique relies on installing ground stations whose locations are precisely known and which assist the location process. However, DGPS and similar techniques do not work in tunnels, underground, and in highly dense built-up areas, because the signal cannot be received or only received very weakly. The technique presented in [7] does not use any radio ranging scheme. Rather, a Cooperative Positioning (CP) method is presented to improve the relative positioning between two vehicles within a VANET, by fusing from different sources. The proposed method which fuses the available low-level GPS data does not depend on any radio ranging technique. The performance of this scheme is studied by analytical and experimental results. Although the principles of the proposed method are similar to those of differential solutions such as differential GPS (DGPS), the authors claim that their technique outperforms DGPS by about 37% and 45% in accuracy and precision of relative positioning, respectively.

There exist several other studies that focus on VANET [8-15]. Most of them use the Received Signal Strength and provide a framework for using the GPS. However, the lack of accuracy in the results obtained makes the methods unsuitable for some VANET applications.

III. The machine learning schemes

We use three widely accepted machine learning techniques: K Nearest Neighbors (KNN), Random Forest and Support Vector Machine (SVM). These three techniques and their adaptation to perform positioning in VANETs are recalled below.

In machine-learning schemes, we have a vector $X_j = \{x_j^1, \ldots, x_j^n\}$ of observations and these observations of X_j are linked to the variables Y_j. The problem is to infer Y_j knowing the vector X_j. In general (but not always) we have to train the algorithm. The algorithm must in this case work on a given number of observations $\{Y_j, X_j\}_{1 \leqslant j \leqslant K}$ to build a model which will be used to perform the predictions. Building this model is equivalent to computing a function $\hat{Y} = f(X)$. Then, given an observation X_i the model can compute $\hat{Y}_i = f(X_i)$. When Y_i is known we can compute the prediction error $\epsilon_i = Y_i - \hat{Y}_i = Y_i - f(X_i)$.

With the same situation, machine learning can perform classification; in this case X belongs to a class Y_l for $l \in \{1, \ldots, p\}$ [1]. When we have an observation X_i, the prediction algorithm will have to predict the most probable class given the observation

[1] We can observe that regression and classification problems are very close.

X_i. In this paper, the issue is a positioning problem, and thus it comes down to a regression problem.

A. k Nearest Neighbors (KNN)

K Nearest Neighbors (KNN) is one of the simplest machine learning algorithms which was first described, to the best of our knowledge, in [8]. As previously stated, we have a given number of observations $\{Y_j, X_j\}_{1 \leqslant j \leqslant K}$ where X_j is usually a vector and Y_j is a real number.

Assuming that we have an observation X_i, we want to predict Y. The KNN algorithm must select the k nearest observations of X_i in $\{Y_j, X_j\}_{1 \leqslant j \leqslant K}$.

Let $i_1, \ldots, i_k$ be the k values which provide the k minimum values of the function

$$g(j) = d(X_j - X_i).$$

In other words $i_1, \ldots, i_k$ are the indexes of the k minimum values of $g(j) = d(X_j - X_i)$. These minimum values can be equal if there are multiple values of X_j at the same distance from X_i.

We have at least the three possibilities: Euclidean,Manhattan, Minkowski for the distance, the most often used being the Euclidean distance.

The value predicted for Y_i will be the mean value of the k values Y_j for the k nearest neighbors of x_i.

$$\hat{Y_i} = \frac{1}{k} \sum_{1}^{k} Y_{i_k}$$

B. Random Forest

We still have a given number of observations $\{Y_j, X_j\}_{1 \leqslant j \leqslant K}$ where X_j is usually a vector and Y_j is real number. The first step in random forest is to create a tree. Using the observations $\{Y_j, X_j\}_{1 \leqslant j \leqslant K}$ we build different sets using different splitting criteria which operate on the vectors $\{X_j\}_{1 \leqslant j \leqslant K}$. Each criterion allows the initial subset to be divided into two subsets. For instance, in Figure III.1 the criterion $X_j < A$ provides the first splitting of the observations. The following two criteria complete the selection tree which ends with four final leaf nodes.

Suppose now that we have a vector X_i and that we want to predict $\hat{Y_i}$. We will use the previous selection tree and determine in which final node the vector X_i is classified. Let us assume that X_i is classified in node 3 as are $X_{i_1}, \ldots, X_{i_k}$. In that case, the prediction of $\hat{Y_i}$ will simply be:

$$\hat{Y_i} = \frac{1}{k} \sum_{j=1}^{k} Y_{i_k}$$

The idea of Random Forest is to correct the error obtained in one selection tree by using the predictions of many independent trees and by using the average value predicted by all these trees. This technique first introduced in [9] is called Random Forest.

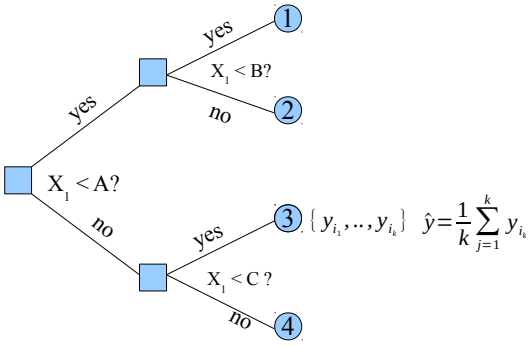

Fig. III.1. Regression tree .

C. The Support Vector Machine regression technique

Generally the positions y_i and the related values x_i (x_i is usually a vector) +are known (thus we know $(y_i, x_i)_{1 \leqslant i \leqslant N}$) and we have to predict the positions using other values: $(x'_i)_{1 \leqslant i \leqslant N}$. We assume that

$$y_i = w^T \phi(x_i) + b \tag{III.1}$$

where w and b are two unknown vectors and $\phi(x)$ an unknown function of a vector x.

To solve these equations, we introduce the following convex optimization problem:

$$\text{minimize} \quad \frac{1}{2}||w||^2$$

$$\text{subject to} \ -\epsilon \leqslant w^T \phi(x_i) + b \leqslant \epsilon. \tag{III.2}$$

Solving this problem gives

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \phi(x_i)^T \phi(x) + b$$

This formula is called the Support Vector expansion. The complexity of the function representation only depends on the dimensionality of the input space. $\phi(x_i)^T \phi(x)$ can be any kernel function, usually a Gaussian kernel is used. α_i, α_i^*, b can be obtained by the Karush-Kuhn-Tucker (KKT) conditions. More details can be found in [10] a tutorial on SVM by Smola.

IV. Numerical results

A straight road of length 600 m is considered to obtain the numerical results. The position on the road is given by $x \in [0, 600]$. We assume that we have three roadside units located at $x = 0m$, $x = 305m$ and $x = 600m$.

We assume that the signal strength received by the vehicles depends on the distance between the vehicles and the road side units. We consider that there are no obstacles on the road to hinder free propagation. Thus the power received is

$$P = \frac{P_0}{r^\beta} \quad with \quad \beta \in [2, 4]$$

We measure the power received in dB thus

$$P^{dB} = 10\frac{log(P)}{log(10)}.$$

Moreover, errors in the measurements are taken into account; we assume a Gaussian noise of zero mean and with a variance 0.05. This can also be interpreted by a log-normal fading which would affect the reception.

The data base is obtained by 20 different measurements at each location of the vehicle, the locations being 15 meters apart. Thus the data consist of 780 sets with three different powers, each of them corresponding to the power received by the three roadside units.

Even if it were possible to do otherwise, for the sake of simplicity we assume that the vehicles send beacons which are received by the three roadside units. The RSUs fuse these data and perform the machine-learning process. The location of the vehicles established can then be sent to them by one of the RSUs.

A. KNN algorithm

For the KNN algorithm, we use the data set directly derived from the power measurements in dB; we do not perform any data processing before using the KNN algorithm. The position error for the KNN algorithm is shown in Figure IV.1; we have set $k = 10$ which seems to be optimal with the data set we have. The code of KNN is found in the R software [11] and in its KNN library. We use the Euclidean distance. The approximation is usually very good with an error of around $5m$ except at the middle of the road where the error can reach $30m$. This is probably because the algorithm confuses a location before the roadside unit in the center of the road with a location after this roadside unit. We observe that the filtering of the measures has a great impact on the quality of the prediction: on average the error is divided by 2. We note however that near the middle roadside unit, the filtering makes little difference. It can be observed that the filtering we have done (averaging over 20 measurements) is very efficient since it is even better than when no error on the measurements is assumed.

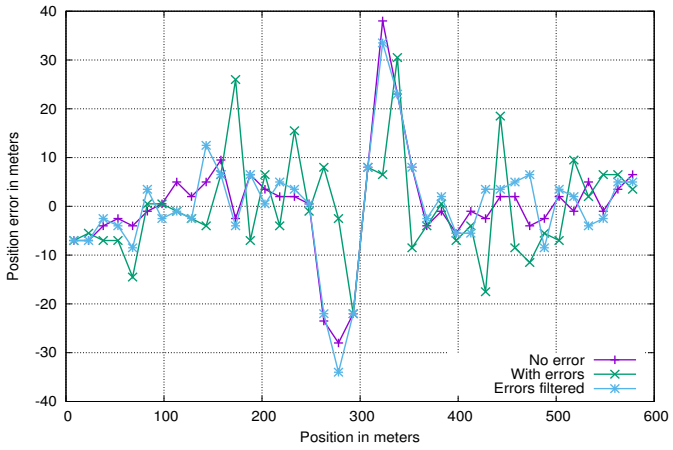

Fig. IV.1. Position errors versus position $x \in [0, 600]$ on the roads with the k nearest neighbors technique.

B. Random Forest algorithm

For the Random Forest algorithm, we use the data set directly derived from the power measurements in dB; we do not process the data before using the algorithm. The position error for the Random Forest algorithm is shown in Figure IV.2. We observe that the Random Forest algorithm offers a good estimation of the location of the vehicle, the prediction being better when we are not at the beginning or at the end of the road where the error tends to increase. As for KNN, filtering the measures improves the prediction but the improvement is less noteworthy, the mean progression in the average error is only 6%. The Random Forest algorithm seems to be able to operate well even with noisy measurements.

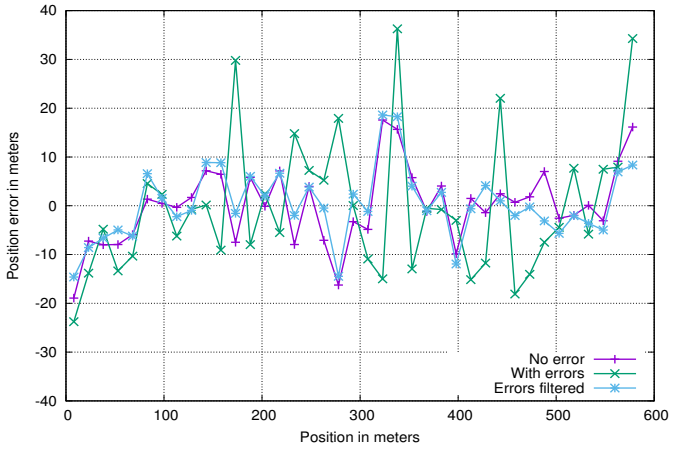

Fig. IV.2. Position errors versus position $x \in [0, 600]$ on the roads with the Random Forest technique.

C. Support Vector Machine algorithm

The libsvm library [12] is used. For the Support Vector Machine algorithm, the data set (powers in dB) is not directly processed. A linear transformation of these powers is performed so that the minimum power becomes 0 and the

maximum power 1. The following values for the parameters are used: $C = 10$, $\epsilon = 10^{-6}$ and an exponential kernel. This means that we have to significantly increase the penalization of not respecting the bounds for the estimation since the default value for C is 1.

Figure IV.3 shows that the error can be large (up to $80m$) when the power received is not filtered . The errors are very significantly decreased when the powers are filtered (at most $20m$). The error is minimum in the center of the road and near to the beginning (at $x = 50m$) and to the end of the road (at $x = 500m$). We note that there is no significant difference when there is no error on the measurements and when the errors are filtered.

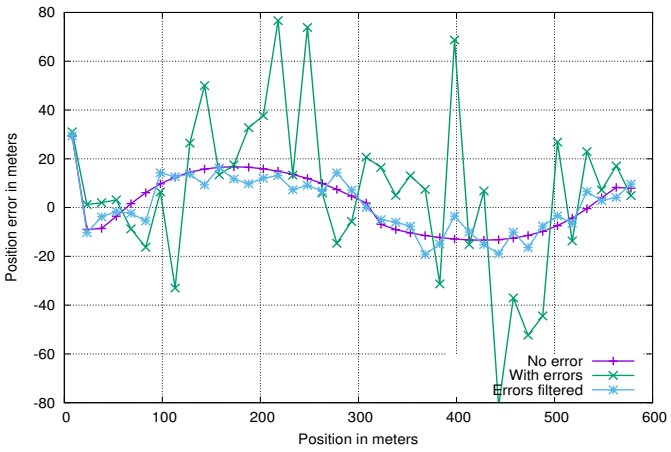

Fig. IV.3. Position errors versus position $x \in [0, 600]$ on the roads with the support vector technique.

D. Results with more fading

In the following, the results of our three algorithms when the power received is more affected by the fading are presented. A Rayleigh fading (of rate 1) is assumed which means that there is no prominent direct path between the vehicle and the roadside units. The power received consists in the random combination of many independent paths. In these conditions, the predictions without filtering the measurement lead to really poor results. Thus they are not included in our presentation.

We observe that the predictions are much less precise than with log-normal fading. Here the predictions are mostly within the interval $[-50m, 50m]$, unlike before, most of the predictions were within the interval $[-20m, 20m]$,

We note that for KNN and Random Forest the predictions are more accurate when the vehicle is in the intervals $[100m, 200m]$ and $[400m, 500m]$, in other words centrally located between two RSUs.

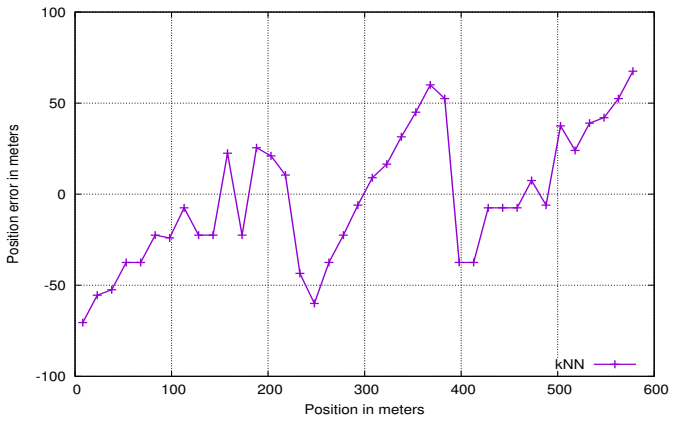

Fig. IV.4. Position errors versus position $x \in [0, 600]$ on the roads with the k nearest neighbors technique.

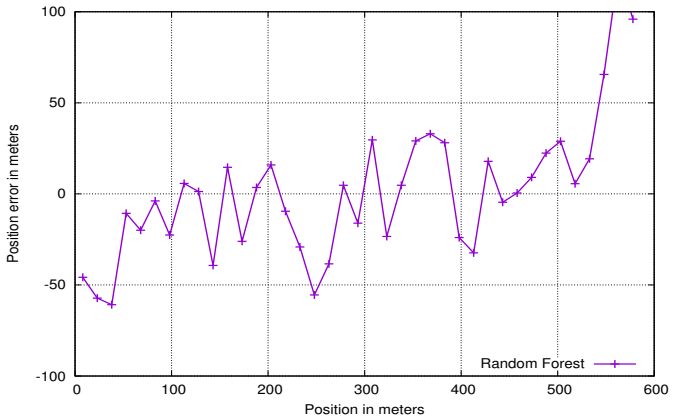

Fig. IV.5. Position errors versus position $x \in [0, 600]$ on the roads with the Random Forest technique.

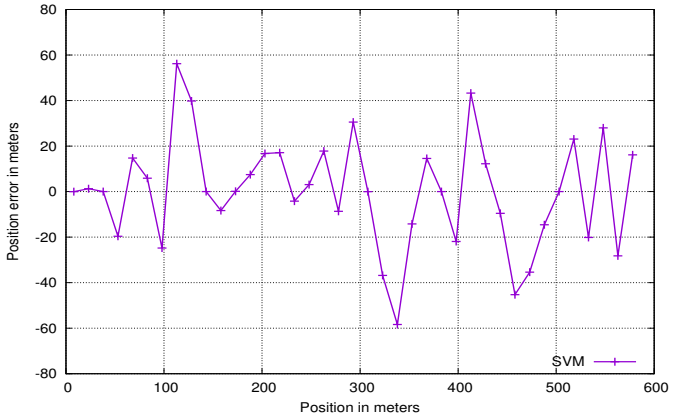

Fig. IV.6. Position errors versus position $x \in [0, 600]$ on the roads with the support vector technique.

E. Comparison : KNN, Random Forest and Support Vector Machine

1) Direct link and log-normal fading: The comparison between KNN, Random Forest and Support Vector Machine is

carried out on the average prediction error and is reported in Table I. We first consider the performance when the measurements are filtered. We note that KNN is the most accurate algorithm with an average error of $5.37m$ before Random Forest with an average error of $7.49m$ and SVM with and an error of $9.68m$. The results are very similar when we do not consider any measurement errors, in particular the ranking between our three schemes remains the same.

When the errors are not filtered, the ranking is different. Random Forest comes first with an average error of $8.05m$ then KNN (average error $10.66m$) and finally SVM (average error $24.67m$).

TABLE I
MEAN ERROR VERSUS PREDICTION TECHNIQUES (DIRECT PATH PROPAGATION)

Predicton Error (in meters)	KNN	RF	SVM
Power: Exact measurement	5.93	6.62	10.50
Power: Measurement with error	10.66	8.05	24.67
Power: Measurement with error filtered	5.37	7.49	9.68

2) No prominent path and Rayleigh fading (no direct path Rayleigh fading): The comparison between the three techniques is given in Table II. In contrast to the results we have obtained with log-normal fading we observe that with Rayleigh fading, it is the Support Vector Machine technique that provides the best average error with a mean accuracy of $16.17m$. In second position we have the Random Forest technique with an average position error of $27.91m$ and in third position the KNN technique with $31.07m$.

TABLE II
MEAN ERROR VERSUS PREDICTION TECHNIQUES (NO DIRECT PATH PROPAGATION)

Predicton Error (in meters)	KNN	RF	SVM
Measurement with error filtered	31.07	27.91	16.17

V. CONCLUSION

In this paper, we present three machine-learning techniques to predict the position of a vehicle using the reception power of packets sent to fixed nodes whose positions are precisely known.

We have studied the KNN technique, the Random Forest technique and the Support Vector Machine technique. The simplest method is the KNN technique; in the data set the scheme selects the k closest samples of the actual measurement. In the Random Forest, we use a classification tree to generate different classes according to a random classification tree. The location in each class is supposed to be the average location of the points in this class. The tree is then used in prediction; the location predicted being that of the training samples at the same leaf of the random trees. The Support Vector Machine is an approximation technique which usually uses kernels as base functions. The main goal is to maintain the sample and its approximation with a bounded error as much as possible. In general, the base functions are exponential functions.

The numerical experiments presented in this paper demonstrate that a precise prediction can only be obtained when there is a main direct path of propagation. In contrast, with Rayleigh fading, the accuracy obtained is much less striking. We also observe the great importance of filtering the measurements. Except for Random Forest, the average prediction error is divided by more than two when the filtering is used. With Rayleigh fading, the filtering is mandatory to obtain acceptable results.

With a main direct path (and thus log-normal fading) we observe that the KNN technique offers the best prediction in terms of mean error, closely followed by the Random Forest technique and somewhat further by the SVM scheme. We observe the same ranking when we do not consider any power measurement errors. Random Forest is very robust against measurement errors. The mean prediction error ($8.05m$) is only improved by 6% when filtering is used. With Rayleigh fading, SVM is by far superior to Random Forest and KNN. As a future work, we plan to extend our study to neural network techniques.

REFERENCES

[1] *Task Group p. IEEE 802.11p*, Wireless Access in Vehicular Environments (WAVE) Draft Standard, 2007.

[2] ETSI EN 302 637-2, "Intelligent Transport Systems (ITS) - Vehicular Communications - Basic Set of Applications - Part 2 : Specification of Cooperative Awareness Basic Service," *History*, vol. 1, pp. 1–44, 2014.

[3] ETSI EN 302 637-3, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service," vol. 2, pp. 1–73, 2014.

[4] M. Porretta, P. Nepa, G. Manara, and F. Giannetti, "Location, location, location," *IEEE Vehicular Technology Magazine*, vol. 3, 2008.

[5] B. C. Liu and K. H. Lin, "Sssd-based mobile positioning: On the accuracy improvement issues in distance and location estimations," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1245–1254, March 2009.

[6] S. A. E. Mohamed, "Why the accuracy of the received signal strengths as a positioning technique was not accurate?" *International Journal of Wireless and Mobile Networks (IJWMN)*, vol. 3, no. 3, pp. 69–82, June 2011.

[7] N. Alam, A. T. Balaei, and A. G. Dempster, "Relative positioning enhancement in vanets: A tight integration approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 47–55, March 2013.

[8] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475879

[9] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[10] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[11] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013. [Online]. Available: http://www.R-project.org/

[12] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199