Wi-Fi Throughput Estimation for Vehicle-to-Network Communication in Heterogeneous Wireless Environments

1st Daniel Teixeira
University of Porto
Instituto de Telecomunicações
Porto, Portugal
danielfilipeteixeira24@gmail.com

2nd Rui Meireles
Vassar College
Poughkeepsie, NY, USA
rui.meireles@vassar.edu

3rd Ana Aguiar
University of Porto
Instituto de Telecomunicações
Porto, Portugal
anaa@fe.up.pt

***Abstract*—Vehicles increasingly need to be connected to networking infrastructure, to support applications such as over-the-air updates, edge computing, and even autonomous driving. The ubiquity of Wi-Fi networks makes them a good candidate for opportunistic vehicular access. However, that ubiquity also creates a problem of choice. In a heterogeneous Wi-Fi environment, with multiple different networks available, it becomes important for vehicles to be able to pick the best-performing one. Focusing on delay-insensitive traffic, we equate network performance with throughput, and aim to estimate it to inform network selection. Throughput estimation is traditionally done by injecting probe traffic, which induces congestion. We provide a solution that avoids this by using only passive measurements of variables such as signal strength to estimate throughput. Taking real-world training data collected in a diverse vehicular Wi-Fi communication scenario, with IEEE 802.11n, ac, and ad networks, we used Symbolic Regression (SR) and Unscented Kalman Filter (UKF) to develop a computationally inexpensive throughput estimation model, UKF-SR. Using a separate testing dataset, we compared the proposed UKF-SR model against traditional linear and support-vector regression, decision tree, random forest, and shallow neural network models. UKF-SR was competitive with even the most complex models. It yielded the lowest Root-Mean-Square Errors (RMSE) for 802.11n and ac, by 4.71 % and 27.59 %, respectively, and was within 1 % of the best-performing model for 802.11ad.**

***Index Terms*—Heterogeneous wireless networks, symbolic regression, vehicular offloading, throughput estimation.**

I. Introduction

Vehicle-to-Network (V2N) communication is increasingly important. It enables myriad applications that promote road safety, efficiency, and entertainment. For example, Tesla vehicles upload information they collect to the cloud to help develop autonomous driving algorithms [1]. And they also download media for streaming, traffic information, and software updates for the vehicles' subsystems.

This work is a result of project FLOYD (POCI-01-0247-FEDER-045912), funded by the European Regional Development Fund (FEDER), through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) and by Portuguese National Funds (OE), through Fundação para a Ciência e Tecnologia, I.P.; and UIDB/50008/2020, funded by the applicable financial framework (FCT/MCTES) (PIDDAC).

Such traffic is typically sent over a cellular connection. But, with ever increasing capacity demands — Cisco reports compound annual traffic growth rates between 7 and 30 % [2] — there is a need for alternatives. The availability of Wi-Fi Access Points (APs) usable from roadways, particularly in urban scenarios [2], [3], makes them a compelling proposition.

However, the abundance of Wi-Fi networks also makes us have to choose which one to use at any given moment, which can be challenging. First, different networks may use different IEEE 802.11 standards, which strike different balances between bandwidth and coverage range. Second, the high degree of mobility associated with vehicular environments limits the amount of time during which each network can be used.

One reasonable goal, particularly for delay-insensitive traffic, is to pick the network that maximizes the amount of data that can be offloaded [4], [5], which equates to cumulative throughput over time. Determining this for a given network can be decomposed into two subproblems:

1) **Estimate** the throughput that the network can currently provide.
2) **Predict**, based on past throughput estimates and other features, how throughput will evolve in the near future.

We focus on the first step, throughput estimation. Previous works have focused on measuring throughput by actively injecting probe traffic into the network [4], [6], introducing unwanted congestion. In contrast, we explore the possibility of leveraging passively-observable variables such as distance to the AP and received signal strength.

Our approach was to take real-world experimental Wi-Fi V2N communication performance data, and then apply Symbolic Regression (SR) to obtain a throughput estimation model. We then refined the model by incorporating an Unscented Kalman Filter (UKF). We tested our SR and UKF-SR models against standard machine learning models Multiple Linear Regression (MLR), Support-Vector Regression (SVR), Decision Tree (DT), Random Forest (RF), and Shallow Neural Network (SNN), on a separate dataset we collected for that very purpose. Our models outperformed the competition while

at the same time being computationally inexpensive and thus suitable for deployment in embedded devices.

In summary, we make the following contributions:

- Describe how we collected network performance data in a realistic Wi-Fi-diverse vehicular scenario to test our estimation models (§III).
- Detail how we used Symbolic Regression (SR) to estimate Wi-Fi network throughput (§IV).
- Present how an Unscented Kalman Filter (UKF) can be used to improve SR estimation, yielding what we call a UKF-SR model (§V).
- Report on the results of the evaluation of our SR and UKF-SR models against standard machine learning models from the literature (§VI).

II. Related Work

In the literature we can find two main types of throughput prediction problems: (i) predict future throughput (for time $t+1, \ldots, t+n$) based on past values and channel quality metrics, and (ii) predict current throughput (time t) using passive measurements. We distinguish the two by referring to the former as prediction, and to the latter as estimation.

Liu & Lee [6] categorized throughput prediction methods as: (i) formula-based, using mathematical expressions; (ii) history-based, predicting throughput using past measurements; or (iii) machine learning (ML) model-based.

Wei *et al.* [7] developed a two-staged prediction mechanism for mobile networks. The algorithm first identifies the user's mobility pattern, and then applies a neural network regression model specific to said pattern. The throughput prediction is based on past values, but also integrates signal strength. The underlying assumption is that past throughput is available to use in future-value prediction. But measuring throughput ordinarily implies injecting large amounts of traffic into the network, rendering it unusable from congestion.

Li *et al.* [8] proposed a packet-pair-based algorithm for measuring throughput, where only a small number of probes are required. However, the algorithm assumes that network conditions are stable, which is not the case in vehicular environments. Furthermore, to generate probes a user must be connected to an AP, while in the network selection problem the decision must be made prior to connection establishment.

The alternative is to rely on passive measurements to estimate throughput, thus not generating additional traffic. A seminal work in throughput modeling and estimation is that of Mathis *et al.* [9], where the authors introduced a mathematical TCP throughput model, function of packet loss probability. However, it does not apply to wireless environments, where losses do not necessarily imply congestion.

Samba *et al.* [10] explored throughput estimation in cellular Long Term Evolution (LTE) networks by leveraging several data sources, such as context information (e.g., distance and speed) and physical layer measurements, such as signal strength. By feeding these data into a random forest algorithm the authors obtained a model capable of explaining 84 % of the variation in throughput.

In this paper we propose two throughput estimation models for Wi-Fi access networks and analyze their ability to adapt to different environments. We include mobility features such as speed and distance to the AP, collected in a real-world vehicular setup. The models only use client-side passive measurements, accessible prior to establishing a connection.

III. Datasets

To develop a good throughput estimation model, it is essential to use data collected in a realistic environment. We had access to such a dataset, created in the context of related prior work [4]. We decided to use it for model training, and collect a new dataset for testing. This let us assess how well the model generalizes across different environments. In this section we describe both datasets, highlighting their differences.

A. Testing dataset

1) Experimental Setup: We drove a vehicle around a stationary access point, while transmitting data through three different networks, each using a different Wi-Fi standard — IEEE 802.11n, ac, and ad, in parallel.

Fig. 1a depicts the location where the experiments took place — Rua D. Frei Vicente da Soledade e Castro in Porto, Portugal. This street runs roughly east-west, and is lined by trees on both sides. It also sits lower than the surrounding terrain, by roughly 2 m. Weather was cloudy but dry, with a temperature of around 25 °C.

The AP remained stationary at the location shown, while the client vehicle did a total of 10 laps around the indicated circuit, traveling counterclockwise. Speed varied between 0 and 40 km/h. Vehicular traffic was substantial, which added to the environment's multipath and line of sight obstructions.

Fig. 2 illustrates the vehicles and hardware devices used in the experiments. We placed a TP-Link Talon AD7200 [12] router on the roof of each vehicle. These routers feature 802.11n, ac, and ad interfaces, making them ideal for our purposes. Given that the original firmware is not user-programmable, we installed LEDE-AD7200, a modified version of the Linux-based LEDE operating system specifically created for the AD7200 routers by the Talon Tools project [13].

Taking advantage of the programmable environment, we wrote simple data producing and consuming applications and deployed them on the routers themselves. Pseudo-random data was sent over the three radio interfaces, from the mobile client to the static AP. Tab. I shows the channels used, all of which are independent.

TABLE I: Wi-Fi channels used for the experiments.

Standard	Channel #	Center freq. (GHz)	Bandwidth (MHz)
802.11n	6	2.437	20
801.11ac	40	5.2	40
802.11ad	2	60.48	2160

Throughput was measured by the receiving AP and sent as feedback to the client over the 802.11n network, as it featured the longest range. This information was saved on a database,

(a) Testing mobility pattern. Access Point GPS coordinates: 41.177, -8.59585. (b) Training. GPS: 41.112, -8.631.

Fig. 1: Experiment locations and client mobility patterns used for testing and training. Imagery ©2022 CNES / Airbus, IGP/DGRF, Maxar Technologies, Map data ©2022 Google.

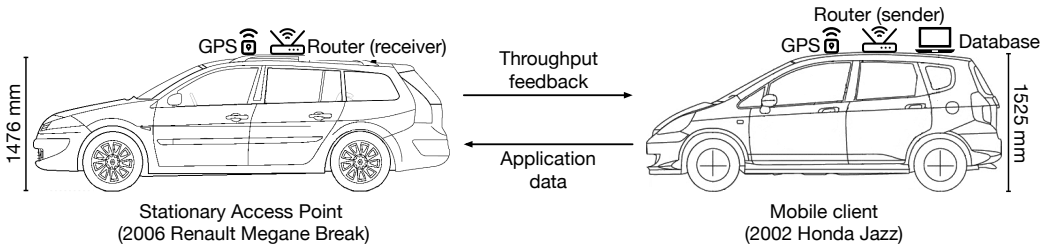

Fig. 2: Experimental setup. Vehicles are drawn to scale. Blueprints courtesy of getoutlines.com [11].

which, due to memory constraints, ran on an external device connected to the client router through Gigabit Ethernet.

Although throughput was measured in the client-AP direction, it is also valid for the reverse direction. We can assume channel symmetry because both AP and client used the same exact communication devices, with the same transmission power and antennas.

Location and time information for the mobile client was provided by a MikroElektronika GNSS 5 click receiver [14], which features a 10 Hz GPS module. However, to increase the number of packets used to compute each throughput data point, our entire analysis used a granularity of 1 Hz. Therefore, we downsampled the GPS information, using the first sample in each second to represent that entire second. Due to lack of additional GNSS 5 devices, the AP was equipped with a commodity 1 Hz GPS receiver.

2) Collected data: Using the described setup, we collected a dataset with over 3750 samples (1250 per Wi-Fi standard). Each sample pertains to a specific (timestamp, network) combination. The timestamp resolution is 1 Hz. A sample contains the mobile client's location coordinates, heading, and speed, along with the measured signal quality, throughput, and transmission data rate. We call this the Teixeira dataset. It is freely available to download on Zenodo [15].

Fig. 3 summarizes the collected data through some of its metrics. Namely, measured throughput, vehicle speed, and GPS horizontal dilution of precision (HDOP). Isik *et al.* [16] suggest the use of dilution of precision values to validate the integrity of GPS estimates. Values below 1 are considered "ideal", and below 2, "excellent". Following these guidelines, we consider the collected GPS data trustworthy, as HDOP values were consistently below 1.

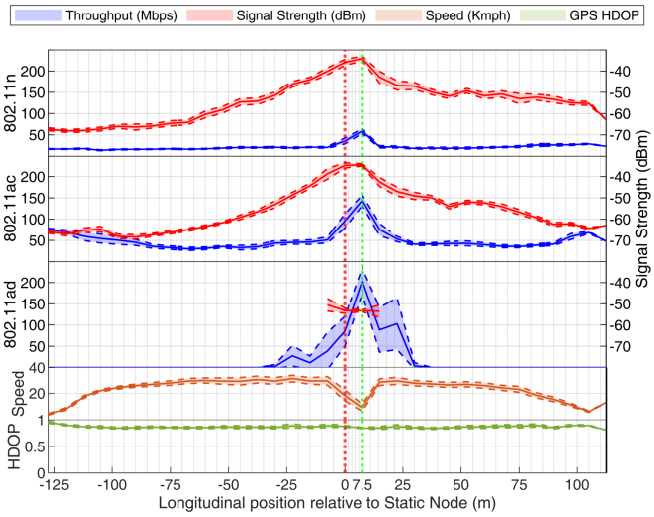

Fig. 3: Mean throughput, RSSI, speed, and GPS horizontal dilution of precision as a function of the client's longitudinal position relative to the AP (negative for West and positive for East). The red vertical dotted line denotes the AP's position; the green vertical dotted line indicates the client's initial (and final) position. Ranges represent the 95 % confidence intervals.

The mobile client's speed ranged between 0 and 40 km/h, with average values in the range of 5 to 30 km/h. The lowest speed values were observed at both extremities, where we had to slow down to perform a u-turn, and at the initial/final position, where the vehicle remained stationary for a period

of time at the experiments' beginning and end.

Throughput varied significantly across the different networks, with a maximum average throughput of 50 Mbps for 802.11n, 150 Mbps for 802.11ac, and 200 Mbps for 802.11ad.

The 802.11n and ac networks remained connected throughout the experiments, even at distances beyond 125 m. On the other hand, 802.11ad was only able to communicate at distances of less than 25 m. This is due to its use of 60 GHz frequencies, which suffer from much greater attenuation than the 2.4 and 5.2 GHz used by 802.11n and ac, respectively.

802.11n's throughput did not vary significantly for distances above 25 m. 802.11ac's throughput on the other hand, initially decreased with distance, but then increased as the vehicle scrubbed speed near the turnaround points — from 25 to 75 Mbps. This behavior reveals an inverse relationship between speed and throughput, suggesting that 802.11ac is more sensitive to mobility than 802.11n.

Signal strength was similar for both 802.11n and ac, peaking at −30 dBm and dropping to between −60 and −70 dBm as the vehicle moved away from the AP. Like throughput, 802.11ad's signal strength was only reported for very short distances.

B. Training dataset

The dataset we used for training, which we call the Meireles dataset, was described in detail in an earlier work [4]. IT is also freely available for download on Zenodo [17].

Just like the testing dataset, it contains network performance data collected in a V2N communication scenario where a vehicle drove a circuit around an access point, exchanging data with it over three different networks, one IEEE 802.11n, one ac, and one ad, in parallel. It contains essentially the same time-indexed mobility and throughput information as the testing dataset, with the same 1 Hz resolution.

The experimental setup used was also very similar to the one used to collect the testing dataset. The same Talon AD7200 routers were used for IEEE 802.11ad communication (the devices used for 802.11n and ac differed), and the devices were also mounted on the roof of two vehicles (although of different makes and models).

The main differences between the two datasets are the environment where they where collected, and the client's mobility pattern. As per Fig. 1a, in the testing dataset the client drove back and forth along a tree-lined road.

In the training dataset, the AP remained at the corner of an intersection as the client approached and left the intersection in every possible direction combination. Fig. 1b depicts the environment and a simplified version of this mobility pattern.

Another difference between the two datasets is the direction of network traffic. In the testing dataset, data flowed from client to AP. In the training dataset, it flowed in the reverse direction. However, since the devices were the same on both ends, the channel can be considered symmetric.

IV. Symbolic Regression-based Estimation

Symbolic Regression (SR) is a data-driven technique used to uncover mathematical expressions that can be used to model a certain feature from a dataset [18]. The mathematical expressions produced can vary in complexity, from simple and intuitive models to complex ones that may be hard to interpret.

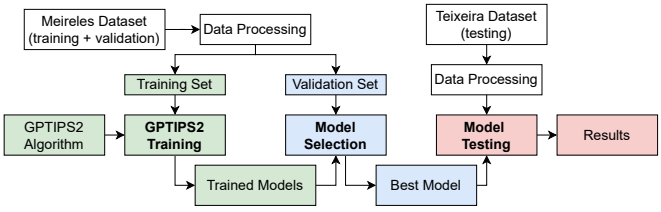

Fig. 4: Symbolic regression pipeline methodology for training, validating, and testing the discovered models.

In this section we leverage SR to develop throughput estimation models based on a Wi-Fi vehicular dataset for IEEE 802.11n/ac/ad access networks.

A. Methodology

To perform SR, we relied on the GPTIPS2 algorithm [18]. GPTIPS2 is a genetic-programming-based algorithm that searches for linear combinations of mathematical functions and input variables. It formulates an equation similar to a linear model, but its terms can be non-linear. These terms are represented by tree structures, where leaves are variables and constants, and intermediate nodes are mathematical operators, e.g., plus, minus, cosine, etc.

To develop the estimation models, we followed the pipeline in Fig. 4, where models go through training, validation, and testing phases. We used the Meireles dataset (§III-B) for training and validation, applying a random 70-30 % split, and the Teixeira dataset (§III-A) for testing. Using two different datasets let us analyze how well the models generalize across environments. To ensure that the models are kept simple and do not over-fit the data, we limited their structure to six terms, each represented by a tree with maximum depth of four. The algorithm was executed 25 times and in each run it evolved 300 models through 100 generations, resulting in a set of 7500 trained models. At the end of each generation, the algorithm runs a tournament selection with a size of 30, to select the best models for cross-over. To choose a model, we plotted the Pareto front with regards to the models' goodness-of-fit, based on the validation set, and their expressional complexity. The expressional complexity is calculated as the sum of the number of nodes of all sub-trees [19]. The chosen model is the one among those in the Pareto front that exhibits the best trade-off between performance and complexity.

B. Estimation Models

We performed SR for each Wi-Fi standard separately, following the previously described methodology. The resulting estimation models for $tput_n$, $tput_{ac}$, and $tput_{ad}$ are described by the following equations:

$$\begin{aligned} \hat{tput}_n = {} & 0.7111 * RSSI - 2.479 * N_{users} \\ & + 11.88 * e^{-N_{users}} + 62.02 \end{aligned} \tag{1}$$

$$\begin{aligned}\hat{tput}_{ac} &= 1.409 * RSSI - 2.667 * v \\ &+ 44311 * \cos(\frac{N_{users}}{RSSI}) \\ &- 11.14 * \cos(N_{users}) - 36.77 * d^{\frac{1}{4}} - 44022\end{aligned} \tag{2}$$

$$\begin{aligned}\hat{tput}_{ad} &= 67.75 * {N_{retries}}^{\frac{1}{4}} - 88.18 * \tanh(v) \\ &- 6.348 * \sqrt{N_{retries}} \\ &- 75.37 * e^{-(v+N_{beacons})^3} + 88.83\end{aligned} \tag{3}$$

, where $\hat{tput}_n$, $\hat{tput}_{ac}$, and $\hat{tput}_{ad}$ are the throughput estimates in Mbps, $RSSI$ is the received signal strength in dBm, d is the vehicle-AP distance in meters, v is the vehicle's speed in m/s, N_{users} is the number of users connected to the AP, and $N_{beacons}$ and $N_{retries}$ are the number of beacon frames received and re-transmissions in the last second, respectively.

The training and validation datasets use RSSI to represent signal quality for 802.11n and ac, but Signal-to-Noise Ratio (SNR) for 802.11ad. The testing dataset uses RSSI for all standards. Therefore, if we were to train a model with SNR we would be unable to test it. This is why, unlike the other equations, the one for $\hat{tput}_{ad}$ doesn't take signal quality into account. Instead, it uses the number of beacons and re-transmissions as proxies. The use of these variables in place of RSSI or SNR may lead to poor performance. To circumvent this limitation and improve the model's accuracy, we generated synthetic RSSI values for 802.11ad using a simplified log-distance path loss model:

$$\begin{gathered}RSSI = RSSI_0 - 10\,\gamma\,\log_{10}\left(\frac{d}{d_0}\right) + X \\ X \sim \mathcal{N}(0,\,\sigma)\end{gathered} \tag{4}$$

, where d_0 is the reference distance (i.e., one meter), $RSSI_0$ is the RSSI at d_0, γ is the path loss exponent, and X is a zero-mean Gaussian random variable with σ standard deviation, reflecting the effect of fading. The path loss exponent and fading standard deviation were set to $\gamma = 1.59$ and $\sigma = 2.1$, respectively, following the results of Karedal et al. for suburban environments [20]. $RSSI_0$ was measured to be −53 dBm for our TP-Link Talon AD7200 access points.

Another issue is the proportion of 802.11ad throughput samples that pertain to periods of disconnection - which given ad's short range, is the vast majority. The large amount of zero-throughput samples will influence the model's training phase, forcing the model to not only learn how to estimate throughput, but also to classify periods of connection versus disconnection (i.e., zero throughput versus non-zero throughput). However, we can easily determine when the vehicle has a viable connection with an AP, and the model should not focus on any other task apart from estimating throughput. Therefore, we re-ran the SR algorithm with the generated RSSI values and trained it using only non-zero throughput samples. Throughput for 802.11ad can then be estimated as:

$$\begin{aligned}\hat{tput}_{ad} &= 0.7334 * RSSI + 47.74 * \sin(v * RSSI) \\ &- 112.6 * \tanh(v)^{1/4} \\ &- 115.8 * \tanh(\cos(v)) * \log(N_{users})^2 + 387.9.\end{aligned} \tag{5}$$

V. Estimation Refinement With Kalman Filters

When estimating throughput through ML or SR models, or manually-discovered formulas, we are subjected to two primary sources of noise: (i) process noise, due to inaccuracies in fitting the data; and (ii) measurement noise, resulting from inherent errors in a model's inputs, such as RSSI or distance.

In this section we improve the SR models' performance by minimizing errors using recursive Bayes filters.

A. Recursive Bayes Filters

Recursive Bayes filters are methods that combine a theoretical model of how a state estimate (e.g., vehicle's speed and direction) evolves over time, with a measurements model that defines what measurements we should expect given an estimated state. The most popular recursive Bayes filter is the classic Kalman Filter (KF), an optimal state-estimator for linear systems with additive Gaussian noise. Intuitively, a KF applies a weighted average over the estimated state and the expected state from the measurements, where the weight is defined as the Kalman gain.

Our goal in using a Kalman filter is to build a state-space model that takes advantage of vehicle distance, direction, and speed to estimate throughput using the SR model. But, if we include Eqs. (1), (2), and (5) in a state-space model, we will have a non-linear system, breaking KF's linearity assumption. The Extended Kalman Filter (EKF) is a non-linear version of the Kalman Filter, where a first-order Taylor series approximation is used to linearize the system. But the use of a first-order approximation may lead to the filter diverging from an acceptable estimate. An alternative to EKF has been proposed by Julier & Uhlmann [21]. It uses an unscented transformation to linearize the system using a small set of chosen points. The Unscented Kalman Filter (UKF) is equivalent to a third-order Taylor-series expansion, severely reducing the divergence probability relative to the EKF algorithm.

The UKF algorithm consists of a prediction step and a correction/filtering step. In the prediction step, a state-space model is used to predict the state for time t — defined by the mean x_t^- n-dimensional vector and the covariance matrix P_t^- — based on a set of *sigma* points that characterize the state from $t-1$. This can be expressed mathematically as:

$$\begin{aligned}x_t^- &= \sum_{i=0}^{2n} w_i^{(m)}\, f(\mathcal{X}_i) \\ P_t^- &= \sum_{i=0}^{2n} w_i^{(c)}\,(f(\mathcal{X}_i) - x_t^-)\,(f(\mathcal{X}_i) - x_t^-)^T + Q_t\end{aligned} \tag{6}$$

, where $\mathcal{X}$ is a set of $2n$ *sigma* points, $w^{(m)}$ and $w^{(c)}$ are the mean and covariance weights, f is the function for the state-space model, and Q_t is the process noise matrix. In some

systems we may extend the formulation to include control inputs. The $sigma$ points $\mathcal{X}$ and their weights represent the non-linear distribution for the state in time $t-1$ (see [21]).

In the correction step, UKF uses the set of measurements for time t and the measurement model to correct the predicted state from Eq. (6). The corrected state, described by the mean vector x_t and covariance matrix P_t, is given as:

$$\begin{aligned}
\mathcal{Z}_t &= \sum_{i=0}^{2n} w_i^{(m)}\, h(\mathcal{X}_i) \\
S_t &= \sum_{i=0}^{2n} w_i^{(c)}\, (h(\mathcal{X}_i) - \mathcal{Z}_t)\, (h(\mathcal{X}_i) - \mathcal{Z}_t)^T + R_t \\
T_t &= \sum_{i=0}^{2n} w_i^{(c)}\, (\mathcal{X}_i - x_t^-)\, (\mathcal{X}_i - x_t^-)^T \\
K_t &= T_t\, S_t^{-1} \\
x_t &= x_t^- + K_t\, (z_t - \mathcal{Z}_t) \\
P_t &= (I - K_t\, T_t)\, P_t^-
\end{aligned} \tag{7}$$

, where $\mathcal{Z}_t$ and S_t are the mean and covariance of the expected measurement's distribution given $\mathcal{X}$ *sigma* points, h is the measurements model, R_t is the measurement noise matrix, T_t is the covariance matrix between the *sigma* points and the predicted state, z_t is the measurements' vector, and K_t is the Kalman gain.

B. Filtered Estimation Models

To model our system using UKF, our state must represent the set of input features for the SR models in Eqs. (1), (2), and (5), as well as other available attributes associated with said features. Considering that the testing dataset was collected with a single connected user, the SR models used in the UKF are simplified versions where $N_{users} = 1$. That said, our state vector x_t is defined as:

$$x_t = \big[lon,\, lat,\, v,\, \theta,\, d,\, RSSI,\, tput\big] \tag{8}$$

, where lon and lat are geographical coordinates, and θ is the vehicle's direction. The corresponding state-space model f is:

$$f(x_{t-1}) = \begin{cases}
lon_t & = lon_{t-1} + \Delta_{lon} * (180/\pi) \\
lat_t & = lat_{t-1} + \Delta_{lat} * (180/\pi) \\
v_t & = v_{t-1} \\
\theta_t & = \theta_{t-1} \\
d_t & = \text{haversine}(lon_t,\, lat_t) \\
RSSI_t & = f_{RSSI}(d_t,\, v_t) \\
tput_t & = f_{tput}(d_t,\, v_t,\, RSSI_t)
\end{cases} \tag{9}$$

, where

$$\Delta_{lon} = \frac{\cos(\theta_{t-1}) * v_{t-1}}{R * \cos(lat_{t-1} * \frac{\pi}{180})},\ \Delta_{lat} = \frac{\sin(\theta_{t-1}) * v_{t-1}}{R}$$

, with R being the earth's radius, approximated as 6371 km.

In Eq. (9), throughput, represented by f_{tput}, is given by the simplified (i.e., $N_{users} = 1$) SR models.

Speed and direction of movement are assumed constant, and used to predict the vehicle's current position from the previous one. The distance between the vehicle and AP positions is computed using the Haversine formula.

RSSI is given by f_{RSSI}, which predicts the current RSSI based on the predicted speed and distance. For 802.11ad this function corresponds to the log-distance path loss model defined in Eq. (4), used to generate synthetic RSSI values. For 802.11n and 802.11ac, we used the previously described SR methodology to create two prediction models that dictate how distance and speed influence RSSI:

$$\begin{aligned}
RSSI_n = 0.050 * v - 2.739 * \log(d^3) \\
- 1.090 * \sqrt{v + d} - 23.22
\end{aligned} \tag{10}$$

$$RSSI_{ac} = -16.24 - 12.43 * \log(d). \tag{11}$$

To analyze these models' performance, we compared them against the log-distance path loss model of Eq. (4). The Root-Mean-Squared Errors (RMSE) for the log-distance path loss model were 32.46 and 25.80 dBm for 802.11n and ac, respectively, but only 7.01 and 8.44 dBm for the SR models.

The measurement model h was implemented as an identity function, i.e., measured values were kept unchanged. The measurements vector z_t has the same structure as x_t in Eq. (8).

C. Noise Matrices

UKF is regulated by a process-noise matrix Q_t and a measurement-noise matrix R_t, which define the variance of state and measurements. Incorrectly defining these matrices may result in poor performance and even divergence.

While there are multiple ways to estimate noise matrices, we followed a simple approach, relying on domain knowledge and manual tuning. For simplicity, we expressed Q_t and R_t as diagonal matrices. This approach is bound to result in sub-optimal performance, but serves as a fast way to implement the system and demonstrate its potential.

The manual tuning of the noise matrices was conducted based on the testing dataset [15], which introduces biases when measuring the model's accuracy. However, we expect the introduced bias to be negligible since our approach will naturally lead to a poorer choice of matrices compared to more statistically-advanced approaches, such as adaptive filters or the Autocovariance Least-Squares method proposed by Odelson *et al.* [22].

VI. Evaluation

In this section we test the models' performance using the Teixeira dataset, and compare our SR and UKF-SR models with against classic ML methods.

A. SR and UKF-SR Models

To assess how SR and UKF-SR models estimate throughput, we tested them using the testing dataset described in §III-A.

Fig. 5 compares the actual throughput measurements with estimated throughput using the SR and UKF-SR models (with both predicted x_t^- and corrected x_t values). The results show that UKF-SR's predictions of Eq. (6) are less accurate than those made by SR alone, due to the simple state-space model used and its dependence on values from time $t-1$.

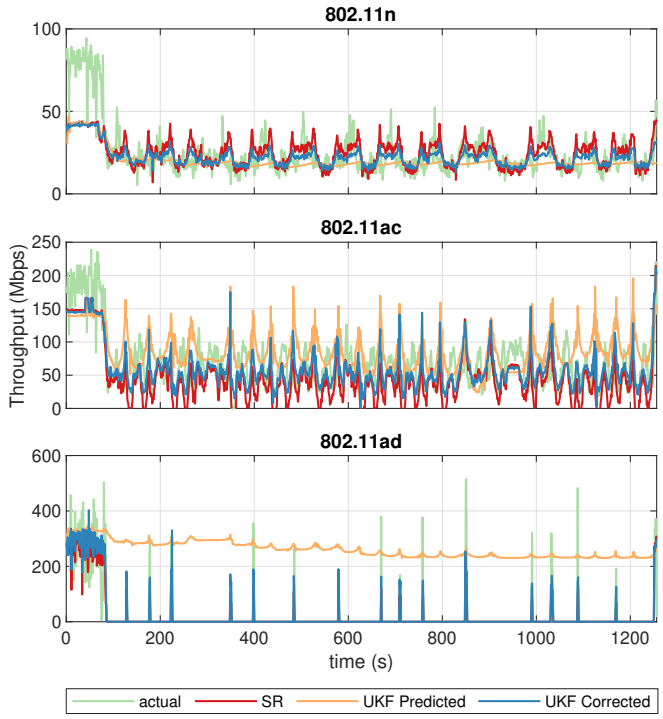

Fig. 5: Estimated throughput using SR and UKF-SR models; "UKF Predicted" corresponds to x_t^- in Eq. (6), and "UKF Corrected" to x_t in Eq. (7).

But, through the measurements model, the predicted throughput is accurately corrected. By leveraging Kalman filtering, we improved our estimation results by 8.33 % for 802.11n, 9.85 % for 802.11ac, and 13.94 % for 802.11ad.

B. Versus Machine Learning Models

To put our models' performance into perspective, we benchmarked them against common machine learning models available in popular software packages. The chosen models were Multiple Linear Regression (MLR), Support-Vector Regression (SVR), Decision Trees (DT), Random Forests (RF), and Shallow Neural Networks (SNN). They were trained following a similar approach to that used for SR, where models go through separate training, validation, and testing phases. But, while for SR the validation step consisted on manually choosing a model among those in the Pareto front, for ML we used Bayesian optimization to maximize the models' performance by tuning their hyperparameters. These vary between models, e.g., the number of neurons in a layer for SNN, and the regularization value for and SVR model. The MLR model is an exception, since it features no hyperparameters.

In Fig. 6 we show the empirical cumulative distribution function (ECDF) plots for the models' absolute errors. For 802.11n, the ECDF error curves are similar among all models, with SR, depicted in orange, managing to compete with Bayesian-optimized ML, and surpassing SNN, RF, and DT models for errors smaller than 10 Mbps. The UKF-SR model, highlighted in red, was the best-performing model, improving upon the SR predictions for over 90 % of errors. Considering 802.11ac, two distinct groups can be observed: a better-performing group consisting of SR, UKF-SR, RF, DT, and Gaussian SVR models; and a worse-performing group composed of Linear SVR, MLR, and SNN.

For 802.11ad, the models were trained using only samples collected during client-AP connection periods, and tested considering both connected and disconnected periods. Synthetic RSSI values were used, as per §IV. Due to the large proportion of zero-throughput samples from disconnected periods, around 90 % of predictions were error-free. Thus, in Fig. 6 we analyze the other 10 % of estimates, related to connected-period samples. Performance across models varied less than for 802.11ac, with the SR model showing good results, but being surpassed by SNN, DT, and Gaussian SVR. However, despite being based on SR, UKF-SR competed with the more computationally demanding algorithms quite well.

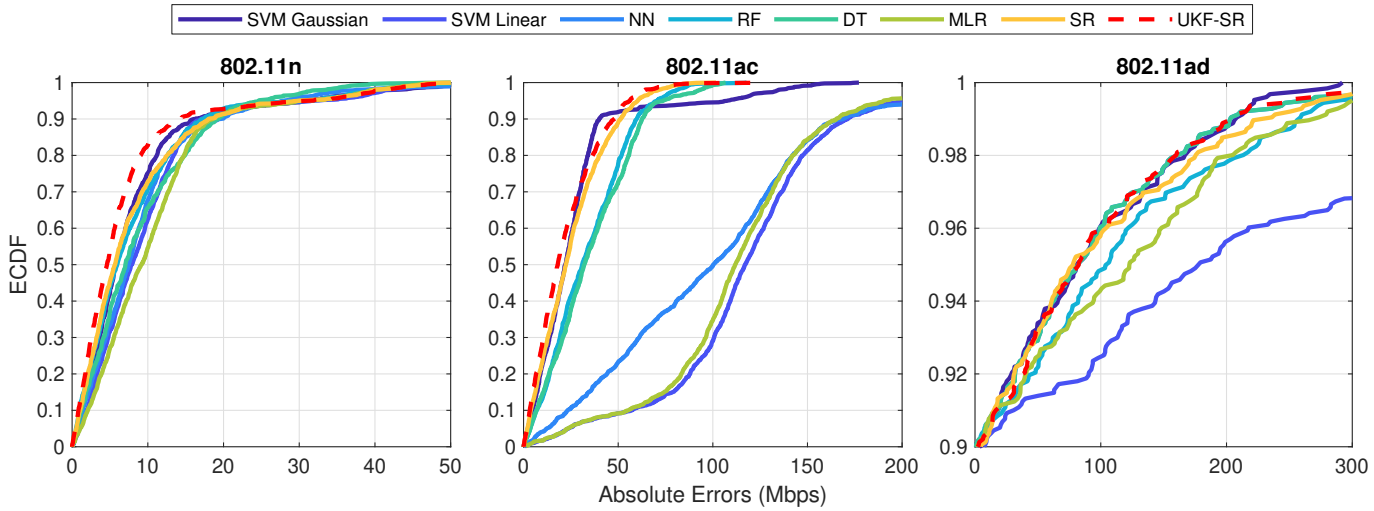

Fig. 6: ECDF plots of the absolute out-sample errors for the ML, SR and UKF-SR models.

TABLE II: Performance results of the ML, SR and UKF-SR models for 802.11n/ac/ad, considering RMSE.

Models	802.11n		802.11ac		802.11ad	
	RMSE	%	RMSE	%	RMSE	%
MLR	13.77	(+19.43)	124.94	(+339.47)	51.90	(+32.70)
RF	12.46	(+08.07)	39.26	(+38.09)	50.92	(+30.20)
SNN	13.07	(+13.36)	129.02	(+353.82)	42.41	(+08.44)
DT	**12.10**	(+04.94)	41.77	(+46.92)	42.41	(+08.44)
SVR Gauss	12.76	(+10.67)	39.62	(+39.36)	**38.70**	(-01.05)
SVR Linear	13.40	(+16.22)	130.38	(+358.60)	116.09	(+196.83)
SVR 2D	14.42	(+25.07)	39.74	(+39.78)	—	—
SVR 3D	21.29	(+84.65)	39.95	(+40.52)	40.92	(+04.63)
SR	12.49	(+08.33)	**31.23**	(+09.85)	44.56	(+13.94)
UKF-SR	**11.53**	—	**28.43**	—	**39.11**	—

Table II shows the different models' RMSE in both absolute form, and as a percentage of a UKF-SR baseline. UKF-SR was the most accurate model for 802.11n and ac, and the second-best for 802.11ad, with RMSE values of 11.53, 28.43, and 39.11 Mbps, respectively. For 802.11n, DT was second-best, with an error 4.94 % higher than UKF-SR, followed by the RF model with 8.07 %. SR was the second-best model for 802.11ac, with an error 9.85 % higher than UKF-SR. The third-best, RF, was significantly worse, with an error 38.09 % higher than UKF-SR. For 802.11ad, UKF-SR was outperformed by the Gaussian SVR model, although by a very small margin of 1.06 %. The third-best performing model for 802.11ad was SVR 2D, with an error 4.63 % higher than Guassian SVR.

VII. Conclusions

In this paper we explored the problem of throughput estimation for Wi-Fi access networks in vehicular environments, using nothing but passively-observable variables.

We used Symbolic Regression (SR) to create an initial model, which we then extended using an Unscented Kalman Filter (UKF), resulting in a model we call UKF-SR. Both models were trained, validated and tested using data collected from real-world vehicular experiments featuring a diversity of Wi-Fi networks. Benchmarking against common machine learning models showed the pure SR model to be competitive, and the UKF-SR model to generally outperform them.

Given their simplicity, SR models are specially suited for embedded systems, such as those that, due to CPU and memory limitations, are unable to leverage more advanced machine learning algorithms.

In the future, we plan to further refine the UKF-SR model by implementing automated approaches for estimating UKF's process and measurement noise matrices. We predict that this will make the model better adapt to different environments and thus improve upon its already promising results.

In practice, throughput is also heavily influenced by the number of stations sharing the access medium. In this in mind, we also plan on extending our investigations to multi-user scenarios.

References

[1] M. Harris, "The radical scope of Tesla's data hoard," *IEEE Spectrum*, October 2022.

[2] "Cisco annual internet report (2018-2023)," 2020, (accessed on 2022-12-11). [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] J. G. P. Rodrigues, A. Aguiar, and J. Barros, "SenseMyCity: Crowdsourcing an Urban Sensor," 2014. DOI: 10.48550/ARXIV.1412.2070

[4] R. Meireles, A. Rodrigues, A. Stanciu, A. Aguiar, and P. Steenkiste, "Exploring Wi-Fi Network Diversity for Vehicle-To-Infrastructure Communication," in *2020 IEEE Vehicular Networking Conference (VNC)*, 2020. DOI: 10.1109/VNC51378.2020.9318407

[5] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, "Can accurate predictions improve video streaming in cellular networks?" in *Proc. of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015. DOI: 10.1145/2699343.2699359

[6] Y. Liu and J. Y. B. Lee, "An Empirical Study of Throughput Prediction in Mobile Data Networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015. DOI: 10.1109/GLOCOM.2015.7417858

[7] B. Wei, W. Kawakami, K. Kanai, J. Katto, and S. Wang, "TRUST: A TCP Throughput Prediction Method in Mobile Networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018. DOI: 10.1109/GLOCOM.2018.8647390

[8] M. Li, M. Claypool, and R. Kinicki, "WBest: A bandwidth estimation tool for IEEE 802.11 wireless networks," in *2008 33rd IEEE Conference on Local Computer Networks (LCN)*, 2008. DOI: 10.1109/LCN.2008.4664193

[9] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, 1997. DOI: 10.1145/263932.264023

[10] A. Samba, Y. Busnel, A. Blanc, P. Dooze, and G. Simon, "Instantaneous throughput prediction in cellular networks: Which information is needed?" in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017. DOI: 10.23919/INM.2017.7987345

[11] Outlines, "Car blueprints and vector drawings," 2022, (accessed on 2022-06-16). [Online]. Available: https://getoutlines.com

[12] TP-Link, "AD7200 Multi-Band Wi-Fi Router Datasheet," 2018, (accessed on 2022-11-13). [Online]. Available: https://static.tp-link.com/2018/201809/20180912/AD7200%202.0%20Datasheet.pdf

[13] D. Steinmetzer, D. Wegemer, and M. Hollick, "Talon Tools: The Framework for Practical IEEE 802.11ad Research," 2017, (accessed on 2022-05-30). [Online]. Available: https://seemoo.de/talon-tools

[14] "MikroElektronika GNSS 5 click," 2022, (accessed on 2022-06-22). [Online]. Available: https://www.mikroe.com/gnss-5-click

[15] D. Teixeira, R. Meireles, and A. Aguiar, "WiPerf Vehicular Wi-Fi Performance Monitoring Dataset," 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6761916

[16] O. K. Isik, J. Hong, I. Petrunin, and A. Tsourdos, "Integrity Analysis for GPS-Based Navigation of UAVs in Urban Environment," *Robotics*, vol. 9, no. 3, 2020. DOI: 10.3390/robotics9030066

[17] R. Meireles, A. Rodrigues, A. Stanciu, A. Aguiar, and P. Steenkiste, "A Dataset for Exploring Wi-Fi Network Diversity in Vehicle-to-Infrastructure Communication," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.6884094

[18] D. P. Searson, "GPTIPS 2: An Open-Source Software Platform for Symbolic Data Mining," *Handbook of Genetic Programming Applications*, 2015. DOI: 10.1007/978-3-319-20883-1_22

[19] G. F. Smits and M. Kotanchek, *Pareto-Front Exploitation in Symbolic Regression*. Springer US, 2005.

[20] J. Karedal, N. Czink, A. Paier, F. Tufvesson, and A. F. Molisch, "Path Loss Modeling for Vehicle-to-Vehicle Communications," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 1, 2011. DOI: 10.1109/TVT.2010.2094632

[21] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, I. Kadar, Ed., vol. 3068, International Society for Optics and Photonics. SPIE, 1997. DOI: 10.1117/12.280797

[22] B. J. Odelson, M. R. Rajamani, and J. B. Rawlings, "A new autocovariance least-squares method for estimating noise covariances," *Automatica*, vol. 42, no. 2, 2006. DOI: 10.1016/j.automatica.2005.09.006