Multi-agent Reinforcement Learning for Autonomous Vehicles in Wireless Sensor Networks

Haitham Afifi, Arunselvan Ramaswamy, and Holger Karl

Abstract—We develop a Deep Reinforcement Learning (DeepRL) based multi-agent algorithm to efficiently control autonomous vehicles in the context of Wireless Sensor Networks (WSNs). In contrast to other applications, WSNs have two metrics for performance evaluation. First, quality of information (QoI) which is used to measure the quality of sensed data. Second, quality of service (QoS) which is used to measure the network’s performance. As a use case, we consider wireless acoustic sensor networks; a group of speakers move inside a room and there are microphones installed on vehicles for streaming the audio data. We formulate an appropriate Markov Decision Process (MDP) and present, besides a centralized solution, a multi-agent Deep Q-learning solution to control the vehicles. We compare the proposed solutions to a naive heuristic and two different real-world implementations: microphones being held or preinstalled.

We show using simulations that the performance of autonomous vehicles in terms of QoI and QoS is better than the real-world implementation and the proposed heuristic. Additionally, we provide theoretical analysis of the performance with respect to WSNs dynamics, such as speed, rooms dimensions and speaker’s talking time.

Index Terms—wireless sensor networks, reinforcement learning, quality of service, quality of information, unmanned aerial vehicles

I. INTRODUCTION

Autonomous vehicles – comprising examples like unmanned ground vehicles (e.g., social robots [1]) or unmanned aerial vehicles (UAV, e.g., drones [2]) – have a broad range of applications, including entertainment, industrial scenarios and ambient-assisted living. They are usually considered as robots with interactive tasks, such as human-robot interaction or robot-robot collaboration. The user’s needs and preferences may, however, change over time. Therefore, it is important for such vehicles

This work was funded by the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) within the framework of the Research Unit FOR2457 “Acoustic Sensor Network” – 282835863

H.Afifi, H. Karl and A. Ramaswamy are with the Dept. of Computer Science, Paderborn University, 33098 Paderborn, Germany (E-mail:haitham.afifi@upb.de, arunr@mail.upb.de, holger.karl@upb.de)

Fig. 1. Example of the motivating scenario

to continuously learn from their interactions, since hard-coded rules will not adapt to these changes.

Rule-based systems are normally labor-intensive and have rigid and static rules, despite working dynamic environments. As an alternative, machine learning-based approaches, in particular, reinforcement learning (RL) one are better suited to solving sequential decision making problems in dynamic environments. They consist of *agents* who refine their behavior over time, based on feedback (*rewards*) for their decisions (*actions*) in given statuses (*states*).

To illustrate our idea, we consider a room (e.g., meeting room or large conference hall) with several human speakers and a couple of such autonomous vehicles, equipped with microphones (Figure I). The objective is to move the mircophones to appropriate positions in the room to obtain high-quality recordings of the speakers (e.g., for question-and-answer sessions in conference setups). Hence, we would like to develop an intelligent control algorithm to move these vehicles so that they can best acquire audio signals when speakers change (e.g., multiple questioners), under the constraint that only a few vehicles are available. We note that our ideas may translate to analogous problems in other modalities like video streaming; but for the sake of concreteness and clarity in presentation, we limit our description to audio streams.

Depending on the problem at hand, there are multiple objectives that yield a multitude of solution behaviors. For instance, localizing a speaker as best

as possible or recording audio in stereo [3], [4] will require moving multiple vehicles towards the “active speaker” to act as a microphone array. Audio filtering, on the other hand, requires at least one vehicle close to the “active speaker” and data from other vehicles is used for noise canceling [5].

In this paper, we assume that the vehicles are connected to an access point that forwards the collected data for monitoring and/or further processing. Therefore, it is important to maintain good connectivity between the vehicles and the access point to avoid packet losses and unnecessary delays. Simultaneously, it is important to be close to the active speaker in order to pick up the best quality audio. For this, we use two quality metrics to evaluate the solution’s performance: Quality of Information (QoI) describing the utility of the collected data (e.g., audio signal-to-noise (SNR) ratio) and Quality of Service (QoS) describing the network performance (e.g., packet losses).

We observe that there is potentially a trade-off between QoI and QoS in dynamic scenarios; vehicles moving towards the speaker to improve the audio quality but at the same time moving away from the access point, which degrades the network performance. This trade-off has not yet been fully explored in the literature, especially not for groups of autonomous vehicles. A possible approach to explore this trade-off would be to formulate it using an optimization problem and try to solve it in (near) real time [6]. But this is fraught with obstacles, to name but a few: the high solution times for conventional optimization problems, the non-obvious way in which QoI/QoS interact towards overall application quality, and the difficulty to adapt to changing application objectives.

To address these challenges, we develop a dynamic programming-based multi-agent solution. In a precursor to this work, [7], we developed a centralized DeepRL-based solution and empirically showed that it outperforms standard ad hoc solutions. In this paper, we extend this work and develop a multi-agent DeepRL solution that is more scalable and practical. As in [7], we compare this solution to the solution with fixed microphone positions and when each speaker carries a microphone. Then, we compare the DeepRL solution to a heuristic one that also controls the vehicles’ movements. In addition to empirical evaluations, we present a theoretical analysis to explain how the changes in the scenario

(e.g., speed of the speaker) will impact the DeepRL training and performance.

II. RELATED WORK

The two main building blocks of autonomous vehicle systems are *scene understanding* and *decision making* [8]. Understanding the scene relies on sensors (such as cameras and microphones) to perceive the environment and localize the vehicles. This is achieved by using different techniques such as motion detection [9] semantic segmentation [10] and mapping [11]. In our work, we assume that these techniques are at hand and are error free. Decision making is further divided into two blocks [12]: planning and control. On the one hand, the planning block is essential for motion and trajectory estimation. There are many possible assumptions regarding the trajectory estimation. For instance, trajectories could be static as in self-driving vehicles, where the route to the destination is predefined and the traffic density is already known [8]. Other planning approaches [13] consider dynamics in the environment as seen in indoor environments [14]. On the other hand, the control block is used for decision making. This could be related to the speed, steering angle or braking. In our work, *we assume that the speed is constant and we control only the steering angle and braking*. Similar to the work in [15], we rely on neural networks to model the motion patterns of active speakers.

Nevertheless, this is not the only role of neural networks; they also act, in our work, as a function approximation for calculating the fitness of QoI/QoS. Accordingly, the neural networks handles the functions of both planning and control blocks. As mentioned before, calculating QoI is based on the data collected from the sensors. Meanwhile, these calculations are based on time-exhaustive algorithms and normally cannot be expressed in explicit expressions [16]. Instead, we use neural networks to implicitly extract features from these data and express the combination of current data and taken actions via some values [17].

Moreover, we focus here on papers that are concerned with data acquisition quality (i.e., QoI) and network connectivity (i.e., QoS), when controlling the vehicles. The work in `uavRelay,staticEnv` optimized the movement of UAV to maximize QoI. They assumed that the sensor and sink nodes are

fixed and the vehicle acts as a relay node for data collection and routing. In contrast, we assume that the vehicles are installed with sensors and move around for data collection, while the sink node's position is fixed. This adds an additional challenge with respect to the position of the vehicle. Furthermore, we assume that all sensor nodes have a direct connection to the access point, thus, they do not require relay nodes. Meanwhile, extending our work to a mobile sink node (with fixed sensors) is a simplified version of our problem and should be straightforward.

Social robotics environments [1] are very close to our own scenario. These robots are usually equipped with one or more sensors and move to improve data acquisition. Nevertheless, such work mainly focuses on data acquisition quality and ignores network aspects. Similarly, planning the trajectory for the vehicles has been studied in [18], [19], but the focus was to achieve low latency and low wireless interference.

Combining both QoS and QoI was modeled in [20], but under the assumption that sensor nodes do not move and we need to select a subset of these nodes for maximizing QoI with a certain QoS level. In summary, previous work aims at either optimizing data acquisition with best effort network performance or maximizing QoS for a given QoI. As opposed to previous work, we explore the trade-off between QoI and QoS.

III. PROBLEM DESCRIPTION

We consider N users in a room, with at most one active speaker at any time; the *active speaker* trajectory is stochastic and unknown.

There are M moving microphones that record audio data from a speaker and transmits them to a fixed access point via wireless channels. At time step t , microphone m takes $\delta_m(t)$ time units to transmit its audio data to the access point. The speaker's velocity is a $U(0, v_{\text{src}})$ random variable, i.e., it is sampled uniformly at random from the interval $[0, v_{\text{src}}]$. When moving, the vehicle velocities are fixed to a constant value of v_m . We assume that the talk time of each active speaker is uniformly distributed in the interval $[\tau_{\min}, \tau_{\max}]$. Hence, the average talk time τ is the same for all speakers.

At every time-step t , we associate the following two performance metrics with each microphone:

acoustic gain and network cost, to imitate both QoI and QoS, respectively. The former, $g_m(t)$, is determined by the distance between the speaker and the microphone (d_m). Then, we have:

$$g_m(t) = \begin{cases} \left(\frac{d_m(t)}{d_{\text{th}}}\right)^{-\frac{1}{3}}, & \text{if } d_m \leq d_{\text{th}} \\ \left(\frac{d_m(t)}{d_{\text{th}}}\right)^{-3}, & \text{otherwise} \end{cases} \quad (1)$$

where d_{th} is a threshold distance at which the relation between d_m and acoustic gain (g_m) changes [21]. The network cost associated with microphone m corresponds to the transmission time $\delta_m(t)$. This depends on the available data rate, which in turn depends on the wireless signal-to-noise ratio at the access point from microphone m . We rely on IEEE 802.11 PHY standard parameters [22] to map the SNR to data rate and required time to transmit a packet; for now, we ignore both propagation and medium access delays for simplicity. Hence, we may normalize the transmission time to be $w_\delta \delta_m(t)$, where w_δ is a normalization factor. Note that an extension to include the previously mentioned delays is illustrated in Section VII.

Finally, we define the *utility* of microphone m as

$$u_m(t) = g_m(t) - w_\delta \delta_m(t) \quad (2)$$

The utility value can be normalized and weighted between QoI and QoS using w_δ . Such weights are application-dependent and are chosen by an application developer. In our setup, $0 < g_m(t) \leq 1$ and $0 < w_\delta \delta_m(t) \leq 0.1$.

Note that in practice, calculating the acoustic gain relies on more complex and time consuming algorithms, which are not easy compute in real-time [23]. Instead, we use acoustic features (e.g., distance between microphones and the speaker) and let the acoustic gain be part of the reward function. The same applies to the network cost.

IV. PROPOSED SOLUTIONS

In this section, we first present the centralized and multi-agent DeepRL solutions. Next, we elucidate the baseline and heuristic solutions. In particular, we highlight the differences between them and the RL solutions.

A. Centralized DeepRL solution

We assume that the locations of the vehicles – equipped with microphones – and the active speaker are known (e.g., via one of the localization techniques mentioned in Section II). Hence, we may define a continuous observation space so that the state vector $s(t)$, at time t , contains the position of each vehicle as well as the position of the active speaker. Using the location of the vehicles and the access point, we calculate the attenuation via the log-normal shadowing model $\mathcal{N}(0, 1.6)$ [24]. As a consequence, we estimate the transmission time from each vehicle. At time t , action vector $a(t)$ determines the direction of movement of all vehicles. Specifically, each vehicle may move in one of the 8 directions: horizontal, vertical and diagonal direction. Additionally, $a(t)$ may dictate that a vehicle remains stationary, i.e., it does not move. Hence, there are 9 possible actions per each vehicle, and $a(t)$ is one among 9^M possibilities.

At time t , the intelligent agent picks an action $a(t)$ and receives feedback for it (performance measure at time t), in the form of rewards. We consider two popular reward (feedback) models. First, we consider the *joint* model that calculates the sum of the microphone utilities:

$$r_J(t) = \sum_{m \in M} u_m(t); \quad (3)$$

it is useful for stereo streaming applications. Second, we consider a *contented* model that selects the maximum utility of all microphones:

$$r_C(t) = \max_{m \in M} u_m(t); \quad (4)$$

it is relevant to noise-canceling applications.

Through the principle of dynamic programming, the problem of optimally controlling the vehicles reduces to taking a sequence of actions, $\{a(t)\}_{t \geq 0}$, such that the cumulative rewards – for r_J or r_C – are maximized over time. In other words, to maximize:

$$\sum_{t \geq 0} \gamma^t r_{J|C}(t). \quad (5)$$

This in turn boils down to calculating the optimal Q -function Q^* . We train a Deep-Q-Network (DQN) to best approximate Q^* , i.e., to minimize the squared Bellman loss function $\mathcal{L}_t$ [25]:

$$\mathcal{L}_t = (r(t) + \gamma \max_{a'} (Q(s(t+1), a'); \theta) - Q(s(t), a(t); \theta))^2, \quad (6)$$

where θ represents the vector of neural network weights. Note that we train our DQN in order to learn from past experiences, which is emulated through the use of an experience replay [25].

B. Multi-agent DeepRL solution

While the centralized solution presented in Section IV-A exhibits excellent empirical characteristics, it is not scalable or practical. To overcome this, we propose a multi-agent solution, wherein each vehicle is controlled by an autonomous agent (M agents in total). Hence, agent m only determines the direction of movement of vehicle (microphone) m . Furthermore, it takes decisions only using the *partial knowledge of the environment*. We assume that agent m only knows the positions of vehicle m and the speaker. In addition to being ignorant of the other microphone positions, it is also blind of their existence. The agents are, however, linked through the reward function. All agents obtain the same reward at time t . Through this common reward structure, our empirical results show that the agents learn to cooperate with each other.

To summarize, we define our environment as follows: we have $m = 1, \dots, M$ agents, whose observation/state $s_m(t)$ represents the position of microphone m and the active speaker position. The action per each agent $a_m(t)$ decides one out of 9 possible actions. Hence, we achieve a reduction in the size of both observation space (size = 4) and action space (size = 9), compared to the centralized solution (observation's size = $2(M+1)$ and action's size = 9^M).

We present solutions based on two multi-agent DeepRL paradigms, (a) *shared policy paradigm* [26], and (b) *separate policy paradigm* [27]. In the shared policy solution, we train a single DQN to take actions in lieu of each of the M intelligent agents. To facilitate effective training, the experiences of all the agents, at every time step, are collected in the experience replay buffer. Training using this buffer amounts to learning from the past experiences of all the agents. The solution based on shared policy is a scalable version of the centralized solution presented in Section IV-A. Note that this design facilitates cooperation between agents, through the shared experience replay buffer [28], [29].

In the solution based on the separate policy paradigm, we train a separate DQN for each of

the M agents. In particular, each agent maintains a separate experience replay buffer in order to train the associated its DQN. At time t , the actions taken by all of the M agents (guided by the DQN) results in the same feedback, i.e., all the agents obtain the same reward at time t . Cooperation between agents is indirect and is achieved through the use of the common reward structure.

In terms of implementation, the centralized solution requires centralized training and centralized execution. In contrast, both multi-agent solutions have distributed execution. Nevertheless, the shared policy relies on centralized training (i.e., partially distributed implementation), while the separate policy's training is distributed (i.e., fully distributed implementation). For further discussions regarding the performances of these two solutions, the reader is referred to Section VII.

Recall that we consider two reward models in order to account for different applications, (a) joint model and (b) contended model. In case of the joint reward model, it can be shown that solving the problem of finding the optimal policy (sequence of actions) for each agent separately, amounts to finding the optimal policy for all the M agents combined. In particular, maximizing $\sum_{t \geq 0} \gamma^t u_m(t)$, separately, for each $1 \leq m \leq M$, amounts to maximizing $\sum_{t \geq 0} \gamma^t r_J(t)$, over time. Recall that $r_J(t) = \sum_{m=1}^M u_m(t)$. However, in the case of contended rewards, one may not make such claims. This is because, unlike the joint model, the utilities are mixed in a non-linear manner in order to obtain the reward at time t . Hence, we believe that the contended model truly tests the performance of a multi-agent solution.

C. Real-world solutions (baselines)

We adopt two solutions from real-world implementations to act as baselines to evaluate our DeepRL solutions. First, a *carry-on* solution where all users carry a microphone with themselves (implying $M = N$ and perfect motion control). We still hold the assumption that only one user is speaking at a time. Second, a *fixed* set-up, where the microphones are preinstalled and cannot be moved.

Additionally, we compare the RL solution to the greedy *heuristic* one, for controlling the position-

ing of the autonomous vehicles (i.e., the microphones). The heuristic solution simply moves all microphones in the direction that maximizes the instant reward at the current state, hence, moving the microphones closer to the speaker. The performance is evaluated in a similar manner to the RL solution's reward model (cp. Section IV-A), so it is either the utilities from all microphones (joint) or the best utility (contended).

V. SIMULATION RESULTS

We start out by highlighting some simulation results, which will be put into a theoretical perspective in the following Section VI.

Our initial simulation setup consists of two moving vehicles $M = 2$, each equipped with a microphone. The centralized RL agent is trained on a speaker that moves in random trajectories while talking, where the average talking time $\tau = 10$ time steps. Then, the next speaker starts speaking from a new random position.¹ For the upcoming results (unless stated otherwise), we assume that $v_m \geq 2v_{src}$. In case of the fixed set-up baseline, the microphones are placed on a uniform equidistant grid.

A. Temporal Difference

We use the temporal difference loss $\mathcal{L}$ defined in Eq. (6) to visualize the convergence of the RL agent while learning and show the results in Fig. 2. We observe in Fig. 2 that $\mathcal{L}$ converges with increasing the training steps for both joint (Fig. 2a) and contended (Fig. 2b) models. Therefore, we conclude that our model is indeed learning.

Furthermore, we observe that changing the relative speed of the vehicles $\frac{v_m}{v_{src}}$ (i.e., the vehicles move longer distances per time step) also changes the value to which $\mathcal{L}$ converges, for the joint reward model. In other words, increasing the speed increases the convergence value. Meanwhile, the convergence for the contended model is almost the same as that of the joint value, but has smaller variance. The joint model's convergence can be improved by tuning the hyperparameters but this is not the focus of this paper. The relation between the speed and the convergence is theoretically explained in Section VI.

¹A previous speaker might have moved while being silent, or a new person starts speaking from elsewhere.

(a) joint model

(b) contented model

Fig. 2. Temporal difference error

(a) joint model

(b) contented model

Fig. 4. Centralized RL vs Heuristic

(a) joint model

(b) contented model

Fig. 3. Comparison with baseline models

B. Compare Centralized RL with Baselines

We compare the performance achieved by the trained RL agent to that of the carry-on and static real-world solutions (Sec. IV-C). For simplicity, we assume that there are two speakers $N = 2$ and only one of them is active; however, each one of them carries a microphone and moves according to its predefined path inside the room. In Fig. 3, we show the achieved reward r_t for 100 time steps per each solution. For each time step, the speaker moves or changes (i.e., another speaker starts talking) and the microphones act accordingly. For the joint model (Fig. 3a), we observe that the RL agent has on average better rewards/performance compared to both the static or carry-on solution. This is due to additional degrees of freedom for the RL agent that can control the position of the microphones to be closer to the speaker. Accordingly, we conclude that the decisions that were taken by the RL agent yield on average better performance than current baseline solutions.

In the case of the contented model (Fig. 3b), carrying the microphone has the best acoustic performance, but not necessarily the best network performance. Because microphones are wirelessly connected to the access point, the network performance degrades if the users carrying the microphones are

far away from the access point. It might hence be beneficial to sacrifice some acoustic performance in return for better network performance. We observe that the RL agent adapts itself to the movement of the speakers to achieve a better overall performance (acoustic and network). Moreover, it is scalable to any number N of speakers and still achieves on average a better performance than both carry and static solutions.

C. Compare Centralized RL with Heuristics

We now compare the trained agent in Section V-B against the greedy heuristic using the environment setups (Fig. 4). The horizontal dashed lines represent the average reward for RL (blue) and heuristic (green) solutions.²

In the joint reward model (Fig. 4a), the heuristic solution has a high variance as all vehicles follow the speaker while maintaining a good network performance, but the acoustic quality drops once the speaker changes. Meanwhile, the RL solution has smaller variance so that it chooses actions that maximize the average reward and not instant rewards.

When we use the contented reward model (Fig. 4b), the RL agent, again, has on average higher rewards than the heuristic algorithm. The reason is that the RL agent learns to keep a distance between the vehicles and only one of them needs to be close to the speaker. Accordingly, when a new speaker pops up, one of the vehicles will be able to reach the new speaker faster than the heuristic.

Another factor that impacts the reward is the relative speed $\frac{v_m}{v_{src}}$. In both reward models, the relative speed $\frac{v_m}{v_{src}} = 0.2$, but the same behavior (RL having

²We show the results in different plots from the baselines for clarity.

Fig. 5. Convergence rate with respect to the speed

higher rewards than the heuristic) was observed for other speed values.

Although the RL solution has a higher reward than the heuristic one, the latter still have better results than the baselines at high speeds. At low speeds, this highly depends on the reward model. For the joint model, the heuristic still performs better than the baselines, while for the contended model the baselines performs better than the heuristic one. These differences are due to the fact that the heuristic solution is better designed for the joint reward model.

D. Changing Vehicle Speeds for Centralized RL

One of the important things to consider when configuring a vehicle is its speed. Hence, we look into how the speed will impact the learning process of the RL agent as well as the impact on its performance.

We start with the impact on convergence point (Fig. 5) by varying $v_m \in \{0.2, 0.4, 1, 2\}v_{src}$. We observe that for low vehicle speeds, the achieved reward is lower than at higher speeds; when $v_m \geq 2v_{src}$ the RL agent has the highest rewards for both joint (Fig. 5a) and contended (Fig. 5b) reward models. Additionally, we observe that RL agents, with high-speed vehicles, have higher rate of convergence. In other words, an RL agent can achieve higher rewards at fewer training steps, compared to other trained RL agents with lower vehicle speeds.

We look further into the impact on the reward, by repeating the experiment in Section V-B for different environments with different vehicle speeds. But this time, the speakers move within different region of interests, where each region is far away from the other one, and all regions are far away from the access point. We compare the average reward (as well as 1st and 3rd quartiles) of 100 time steps of

Fig. 6. Comparing RL to baselines with respect to the speed

a trained RL model to that of carry-on and static solutions (Fig. 6).

When using the joint model (Fig. 6a), the RL solution has on average a better performance when $\frac{v_m}{v_{src}}$ is high. For low values of $\frac{v_m}{v_{src}}$, the RL performance can be even worse than either static, carry or both solutions (depicted by 1st quartile). Similarly for the contended model (Fig. 6b), we observe that the higher the speed, the better the RL performance. Meanwhile, the performance of the RL agent is still better than both static and carry solutions, even for low speeds. In contrast to Fig. 3b, note that carrying the microphones is not better than having preinstalled ones (static), when the speakers keep moving in regions that are far from the access point and have high network delays.

E. Changing Microphones Speeds for Multi-agent RL

Following up from the speed analysis, we investigate the impact of changing the vehicle's speed on the multi-agent solution. We recall that a multi-agent environment has only the contended reward model and two different policies: shared and separate.

First, we look at the convergence value of mean reward for both policies (Fig. 7). On the one hand, the shared policy is sensitive to the microphone's speed; the mean reward increases with the microphone's speed. On the other hand, the separate policy's reward is less sensitive to the microphone's speed, yet, the mean reward and microphone's speed are also directly proportional.

Furthermore, we observe that at high speeds ($\frac{v_m}{v_{src}} = 2$) the shared and separate policies have almost the same value of converged mean reward. Nevertheless, at low speeds ($\frac{v_m}{v_{src}} = 0.2$), the separate policy has higher value of reward convergence

Fig. 7. Average reward with respect to the speed

than the shared policy. Accordingly, we conclude that a task-specific policy (as in separate policy) may be better than having a global policy (as in shared policy) depending on the parameters in the environment.

This is indeed an inherited property from modular hierarchical learning [30]. The formulation with partial observations and unknown variables (e.g., the position of the other microphones) is a complex environment, which is hard to solve with a single monolithic policy [31]. Hence, hierarchical learning can be used to divide the main objective into sub-goals or sub-policies using fined-grained [32] or structured supervision [31]. In our formulation, the shared policy is equivalent to a global monolithic policy and the separate policy is equivalent to learning sub-policies. In contrast to the previous work, our sub-policies are unsupervised, i.e., all agents have the same reward functions, while they decide on their own policies.

We compare the convergence of multi-agent solutions to that of the centralized one (Fig. 8). For a clear visibility of the plots, we show the convergence at low ($\frac{v_m}{v_{src}} = 0.2$) and high ($\frac{v_m}{v_{src}} = 2$) speeds. The vertical lines represent the point of convergence for each solution.

Again, the separate policy has higher rewards than the shared one, while the centralized solution is very similar to the separate one (Fig. 8a). For a high speed (Fig. 8b), the separate and shared policies have same reward but the centralized solution performs better (i.e., higher rewards) than either of other policies.

For a complete analysis of the impact of the microphone's speed, we compare the converged mean reward of the multi-agent policies to that of the heuristic (Fig. 9). For both slow and high speeds (Fig. 9a and Fig. 9b respectively), the averaged

Fig. 8. Average reward with respect to the policy

Fig. 9. Multi-agent vs Heuristic

reward for multi-agent's policies is higher than the heuristic's one. Hence, we conclude that RL solutions are better than the heuristic one while the centralized RL solution is at least as good as the multi-agent solutions.

F. Up-scaling with multi-agent RL

Although the centralized RL solution has better results compared to multiple agents, it is not scalable; the more microphones it has, the longer the training will be (recall that action space = 9^M). Meanwhile, a multi-agent approach can better scale up with some trade-off on performance.

We show in Fig. 10 how the multi-agent solutions scale up with the number of microphones. The convergence values are very close for different numbers of microphones for both shared and separate policies. Unlike the multi-agent RL solution, the centralized one shows multiple spikes as the number of microphones increases, which clearly indicates that the agent has not converged. Due to the exponential increase in the action space with respect to the number of microphones, we drop the convergence results for 8 microphones, since it exceeds our memory (64 GB RAM) and CPU (16 cores Intel Xeon with 2.3 GHz clock frequency) resources.

Fig. 10. Temporal difference error for up-scaling

Fig. 11. Average reward with respect to the number of microphones

Next, we look at the gain in reward when increasing the number of microphones. The shared policy (Fig. 11a) converges to higher rewards as the number of microphones increase. Moreover, the reward gain diminishes as the number of microphones increases: increasing from 2 to 3 microphones yields a gain of 14, while doubling the number of nodes from 4 to 8 yields only a gain of 8.6.

A similar behavior is observed for the separate policy (Fig. 11b); the gain increases with more and more microphones, yet convergence takes longer the more microphones there are. This is due to the fact that each agent has its own buffer for experience replay, unlike the shared policy where all agents share the same experience replay. Again, the gain in reward decreases as the number of microphone increases; moving from 2 to 4 microphones gains 11.6 reward, while doubling the number of nodes from 4 to 8 gives only 8.6 gain.

VI. THEORETICAL DISCUSSION

In this section, we show how a dynamic environment impacts the behavior of both DQN and heuristic models. A generic theory of this impact is discussed in [33]. In this section, we present a

theory that is specific to our problem. We show a relation between the changing attributes of the environment and the changes in the DQN behavior. For simplicity in notation, we drop the time index from $s(t)$, $a(t)$ and $r(t)$, and merely use s , a and r , respectively. Wherever necessary, we explicitly revert back to the notation involving the time index t .

A. Impact of Environment Setup on DQN Training

As briefly stated earlier, in Section IV-A, the main objective of DQN is to find the optimal state-action function $Q^*(s, a)$ (optimal Q-function). To do this, DQN in turn finds an optimal set of weights of a neural network, θ^* , such that it closely mirrors the Q function. In other words,

$$\arg \max_a Q(s, a; \theta^*) = \arg \max_a Q^*(s, a) \quad (7)$$

Note that θ^* represents the set of neural network weights which minimize the squared Bellman loss function $\mathcal{L}$.

Now the question is, can any of the environment variables (configuration of the environment) impact the convergence of RL? If yes, how? to answer these questions, we first consider the expected value of TD error (expected value of the squared Bellman loss when picking action a in state s) as a reference:

$$\mathcal{B} = \left(r + \gamma \int \arg \max_{a'} Q(s', a'; \theta) p_\theta(s' | s, a) - Q(s, a; \theta) \right)^2, \quad (8)$$

where the expectation is taken with respect to the distribution of the next state s' . Ideally speaking, if $\nabla_\theta \mathcal{B} = 0$ for every state action pair (s, a) , then the model has converged. From Eq. 8, this seems to depend on the learning the kernel transition $p_\theta(s' | s, a)$ such that:

Fig. 12. Depicting the attributes for the environment. In this example, x_{src} and x'_{src} are the positions of two different speakers, because the distance between them is bigger than v_{src}

$$\begin{aligned} p(s'|s, a) &\approx p(s'|s, a; \theta) \\ &= p\left(s'|s, \arg \max_a Q(s, a; \theta)\right) \end{aligned} \quad (9)$$

In other words, updating θ will alter the estimate of the transition kernel. However, this is not the only parameter that influences the estimation. Next, we show how the environment variables can influence the kernel transition estimation.

Let us consider the centralized solution. A state s is defined by the position of the speaker x_{src} and the position of the microphones X_M . The speaker position x_{src} does not depend on the microphone positions X_M , but it does vice versa. Hence, Eq. (9) becomes:

$$\begin{aligned} p(s'|s, a) &= p(x'_{\text{src}}, X'_M | x_{\text{src}}, X_M, a) \\ &= p(x'_{\text{src}} | x_{\text{src}}) p(X'_M | x_{\text{src}}, X_M, a) \end{aligned} \quad (10)$$

Without loss of generality, we assume that the speakers are inside a square room, with dimension $z \times z$ (Figure 12). The initial position of the speaker is assumed to be a random variable that is uniformly distributed across the room. Then, its position changes in accordance to the following transition probability (within an area of $4v_{\text{src}}^2$):

$$p(x'_{\text{src}} | x_{\text{src}}) = \begin{cases} \frac{\|x'_{\text{src}}\|_2}{4v_{\text{src}}^2} \left(\frac{\eta}{4v_{\text{src}}^2} + \frac{1-\eta}{z^2} \right), & \|x'_{\text{src}} - x_{\text{src}}\|_2 < \sqrt{2}v_{\text{src}} \\ \frac{\|x'_{\text{src}}\|_2}{z^2}, & \text{otherwise} \end{cases} \quad (11)$$

where η is the probability that the same speaker is talking. In simple words, if the distance between the new position of the speaker x'_{src} and the old one x_{src} is less than the distance that a speaker moves within 1 time step, i.e., v_{src} , then two cases are possible. First, the same speaker is

talking and has moved with average speed v_{src} in either horizontal, vertical or diagonal directions, i.e., $p(x'_{\text{src}} | \text{same src}) = \frac{\|x'_{\text{src}}\|_2}{4v_{\text{src}}^2}$. Second, this is the position of a new speaker, which is uniformly distributed inside the room, i.e., $p(x'_{\text{src}} | \text{new src}) = \frac{\|x'_{\text{src}}\|_2}{z^2}$. If we have a counter $\omega(t)$ that resets each time a new speaker starts talking, then we can rewrite η as

$$\eta(t) = p(\tau > \omega(t)) \quad (12)$$

Hence, we have shown theoretically how the room dimensions, speaker's speed and talking time (Eq. (11) and Eq. (12)) impact the behavior of the trained model. Consequently, and without loss of generality, the system dynamics characterize the behavior of the training process. As an example, we have shown empirically how the relative speed of the autonomous vehicle, with respect to the speaker's speed, impacts the training phase (Section V-D).

Furthermore, from Eq. 8, the discount factor γ is another parameter that influence the converged RL policy. Indeed, γ is tightly related to the talking time τ and defines the scope of which the agent maximizes the rewards. To explain more in detail, we define the discounted reward $G(t)$ as

$$G(t) = \sum_{k=0}^{\infty} \gamma^k r(t+k), \quad (13)$$

then we relate it to the time horizon τ , so that

$$G(t) = \sum_{k=0}^{\infty} e^{\frac{-k}{\tau}} r(t+k) \quad (14)$$

where $\gamma = e^{\frac{-1}{\tau}}$. When $k \geq \tau$, the contributions of future rewards decreases exponentially. By picking $\tau = \infty \rightarrow \gamma = 1$, we model a system with only one speaker speaking all the time. When $\tau \approx 0 \rightarrow \gamma \approx 0$, we model a system where there are multiple speaker, each speaking for a very short duration, so that the objective is to maximize the reward for this short duration. Consequently, the discount factor should be carefully selected with respect so the talking time; otherwise, the RL solution will converge to a sub-optimal one.

B. Heuristic Sub-optimality

When considering the joint reward model, the proposed heuristic is indeed equivalent to solving an MDP associated with the discounted reward

problem with a discount factor of $\gamma = 0$. This explains the similarity in the performances of the heuristic and the RL model in Fig. 4. To elaborate more, let us substitute in Eq. (13) $\gamma = 0$, then the discounted return $Q^*(s_t, a_t) = r(s_t, a_t)$. Hence, in Q-learning we end up picking actions that maximize the rewards, i.e., $\arg\max_a r(s, a)$. This is also, roughly

speaking, the strategy of the heuristic baseline, which moves all microphones towards the speaker, regardless how long the speaker has been talking.

As the values of τ and v_{src} decrease, the speakers' positions will start bouncing inside the room (e.g., seminar rooms where the speakers normally do not change their places). This would obviously result in a poor performance using the heuristic solution. Meanwhile, the RL solution is generic and can be retrained and adapted to the changes in the environment.

VII. LIMITATIONS AND FUTURE WORK

The work in [7] was the first step towards autonomous vehicles targeting both QoS and QoI, but the proposed solution was a centralized one. In this work, we extended [7] to a decentralized multi-agent DeepRL based solution for two different reward models. Both the centralized and the decentralized solutions have their own sets of pros and cons. Additionally, they open the door to new questions that we highlight in this section.

A. Convergence Rate and Optimality

The centralized solution (2 speakers and 2 microphones) converges and achieves the best results when compared to all other presented solutions. But the problem is that when we increase the number of nodes, the action space increases exponentially. Accordingly, the model needs longer training steps for converging and in some cases (e.g., $M = 8$) huge amounts of resources. Therefore, it is not practical and can only be used as a baseline. Of course we can limit the number nodes to move (i.e., K nodes), which will decrease the size of action space from 9^M to $\frac{9^K M!}{(M-K)!}$. This will, however, limits the solution's optimality.

Alternatively, we presented two different multi-agent formulations. One common property for these formulations is that each agent takes an action independent of the actions taken by other agents.

This will avoid the need for frequent exchange of information between the agents, resulting in a quick decision making for moving the microphones. Therefore, not only do multi-agent formulation have a smaller action space, but also they have a similar communication paradigm to that of the centralized solution.

The multi-agent formulations rely on partial information of current states while the action space per agent is the same for any number of nodes. Therefore, their performances are slightly lower than the centralized solution's performance, which is the price they pay to easily upscale for different number of microphones. Nevertheless, each multi-agent solution has its own additional limitation.

B. Privacy and Online Training

On the one hand, the shared policy provides a generic model that can be used by any node in the network. Hence, newly joining nodes can use the same trained model and improve their performance over time. This requires, however, sharing experience from all other nodes, which yields two problems. First, it raises privacy concerns since experiences will be visible to other agents. Second, it will not allow online learning since all experiences need to be shared on one central server, processed and then update the trained model. The fact that there is a round trip between the agent and the central server (sending training data and receiving a new model) will slow down training process due to the communication delay. Note that the communication delay for sending the trained data is different from the transmission time for streaming the data.

C. Asymmetry and Generality

On the other hand, the separate policy learns using its own experience with no need of sharing experience with other nodes. Hence, it is easier to use for online learning without privacy concerns. A challenge here is that the learned models could be asymmetric, meaning that each agent learns to be task-specific. The problem then appears when a new agent (i.e., a new vehicle) joins the network, what is the new agent's model? Training the model from scratch is impractical because it may take long time for each agent to find its new task. Accordingly, this might introduce instabilities in the performance of the multi-agent solution during the training phase,

when switching from the old task tasks to the new ones. This of course depends on the method used such as ensemble [34], federated [35] learning or Q-mix training [36].

D. State Inaccuracies

In our formulation, we assumed a perfect knowledge about current states, which may be impractical in real-world implementation. For example, localization techniques are known to have some uncertainties about their output. In this work, we assumed that these uncertainties are very small so that we can ignore them. But even for perfect localization estimation, there may be some (network or processing) delays when getting the updates. These delays were ignored, assuming that the time interval between these updates (i.e., the time between changes in the RL's states) is bigger than processing and network delays.

Indeed, we can retrain our RL solutions with new data to include these delay, yet we need to add additional parameters to the observation space (such as uncertainties or delay) to describe input inaccuracies. The robustness and performance of the RL solutions in that case is a long discussion that we shall present in a follow up to this paper.

VIII. SUMMARY

In this paper, we presented a DeepRL solution to control autonomous vehicles within the context of data acquisition in WSNs. The results are supported with simulations and theoretical discussions. Our objective is to stream audio data in a room with multiple microphones and two to many moving speakers. Consequently, the DeepRL solution will control the vehicles –carrying microphones– to achieve high quality streams with low latency. We compared the performance of RL-controlled autonomous vehicles to current solutions (called baselines) and a heuristic one to show that the proposed solution achieves very good performance and flexibility. We supported our results with theoretical analysis and showed how the attributes in a dynamic environment (such as v_{src} , τ and room dimensions) impact the RL behavior.

In addition to a centralized DeepRL solution, we proposed two scalable multi-agent RL solutions, based on separate and shared policies. We showed that both policies scale with respect to the number of microphones, but each has its own advantages with

respect to the specific application. For the sake of testing and further comparisons, we have published our implementation as well [37].

REFERENCES

- [1] N. Akalin and A. Loutfi, "Reinforcement learning approaches in social robotics," 2021.
- [2] I. Jawhar, N. Mohamed, and J. Al-Jaroodi, "Uav-based data communication in wireless sensor networks: Models and strategies," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 687–694.
- [3] A. Brendel and W. Kellermann, "Distance estimation of acoustic sources using the coherent-to-diffuse power ratio based on distributed training," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 1–5.
- [4] J. Schmalenstroer, J. Heymann, L. Drude, C. Boeddecker, and R. Haeb-Umbach, "Multi-stage coherence drift based sampling rate synchronization for acoustic beamforming," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, 2017, pp. 1–6.
- [5] X. Xu, Z. Cao, H. Qiu, M. Wang, and X. Tang, "Unmanned aerial vehicle (uav) for collecting audio data," Feb. 13 2018, uS Patent 9,889,931.
- [6] M. Günther, H. Afifi, A. Brendel, H. Karl, and W. Kellermann, "Network-aware optimal microphone channel selection in wireless acoustic sensor networks," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Jun. 2021.
- [7] H. Afifi, A. Ramaswamy, and H. Karl, "Reinforcement learning for autonomous vehicle movements in wireless sensor networks," in *2021 IEEE International Conference on Communications (ICC): IoT and Sensor Networks Symposium (IEEE ICC'21 - IoTSN Symposium)*, Montreal, Canada, Jun. 2021.
- [8] K. Li and P. Ioannou, "Modeling of traffic flow of automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 2, pp. 99–113, 2004.
- [9] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "Modnet: Motion and appearance based moving object detection network for autonomous driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2859–2864.
- [10] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, "Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–8.
- [11] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, "Visual slam for automated driving: Exploring the applications of deep learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 360–36010.
- [12] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. K. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *CoRR*, vol. abs/2002.00444, 2020. [Online]. Available: <https://arxiv.org/abs/2002.00444>
- [13] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at <http://planning.cs.uiuc.edu/>.
- [14] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, 1999, pp. 473–479 vol.1.

- [15] O. Saha, G. Ren, J. Heydari, V. Ganapathy, and M. Shah, "Deep reinforcement learning based online area covering autonomous robot," in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, 2021, pp. 21–25.
- [16] M. Günther, H. Afifi, A. Brendel, H. Karl, and W. Kellermann, "Network-Aware Optimal Microphone Channel Selection in Wireless Acoustic Sensor Networks," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [17] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: <http://arxiv.org/abs/1509.06461>
- [18] U. Challita, A. Ferdowsi, M. Chen, and W. Saad, "Machine learning for wireless connectivity and security of cellular-connected uavs," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 28–35, 2019.
- [19] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected uavs: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, p. 2125–2140, Apr 2019. [Online]. Available: <http://dx.doi.org/10.1109/TWC.2019.2900035>
- [20] R. Katona, V. Cionca, D. O'Shea, and D. Pesch, "Virtual Network Embedding for Wireless Sensor Networks Time Efficient QoS/QoI Aware Approach," *IEEE Internet of Things Journal*, pp. 1–1, 2020.
- [21] M. B. Salin and D. A. Kosteev, "Examples of usage of nearfield acoustic holography methods for far field estimations: Part 1. cw signals," 2018.
- [22] I. S. 802.11-2016, "IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," (*Revision of IEEE Std 802.11-2012*), pp. 1–3534, 2016.
- [23] M. Günther, A. Brendel, and W. Kellermann, "Online estimation of time-variant microphone utility in wireless acoustic sensor networks using single-channel signal features," in *29th European Signal Processing Conference (EUSIPCO 2021)*, Dublin, Ireland, 2021.
- [24] I. T. Union. [Online]. Available: <https://www.itu.int/rec/R-REC-P.1238-10-201908-I/en>
- [25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction, chapter 14*, 2nd ed. The MIT Press, 2018.
- [26] J. Zhang, Y. Pan, H. Yang, and Y. Fang, "Scalable deep multi-agent reinforcement learning via observation embedding and parameter noise," *IEEE Access*, vol. 7, pp. 54 615–54 622, 2019.
- [27] M. Lanctot, V. F. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver, and T. Graepel, "A unified game-theoretic approach to multiagent reinforcement learning," *CoRR*, vol. abs/1711.00832, 2017. [Online]. Available: <http://arxiv.org/abs/1711.00832>
- [28] A. Oroojlooyjadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *CoRR*, vol. abs/1908.03963, 2019. [Online]. Available: <http://arxiv.org/abs/1908.03963>
- [29] R. Nair, "Coordinating multiagent teams in uncertain domains using distributed pomdps," PhD thesis, 2004.
- [30] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.
- [31] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," *CoRR*, vol. abs/1611.01796, 2016. [Online]. Available: <http://arxiv.org/abs/1611.01796>
- [32] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic Compositionality Through Recursive Matrix-Vector Spaces," in *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.
- [33] A. Ramaswamy, "Theory of Deep Q-Learning: A Dynamical Systems Perspective," 2020. [Online]. Available: arxiv.org/abs/2008.10870
- [34] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.
- [35] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1387–1395.
- [36] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4295–4304.
- [37] H. Afifi, "RL for autonomous vehicles," 2020. [Online]. Available: https://git.cs.uni-paderborn.de/hafifi/rl_uav