Reinforcement Learning Based Angle-of-Arrival Detection for Millimeter-Wave Software-Defined Radio Systems

Marc Jean[1] and Murat Yuksel[2]

University of Central Florida, Orlando, FL 32826, USA
marc1988@knights.ucf.edu, Murat.Yuksel@ucf.edu

Abstract. Millimeter-wave (mmWave) signals experience severe environmental path loss. To mitigate the path loss, beam-forming methods are used to realize directional mmWave beams that can travel longer. Yet, advanced algorithms are needed to track these directional beams by detecting angle-of-arrival (AoA) and aligning the transmit and receive antennas. To realize these advanced beam-forming algorithms in real world scenarios, Software-Defined Radio (SDR) platforms that allow both high-level programming capability and mmWave beam-forming are needed. Using a low-cost mmWave SDR platform, we design and prototype two reinforcement learning (RL) algorithms for AoA detection, i.e., Q- and Double Q-learning. We evaluate these algorithms and study the trade-offs involved in their design.

1 Introduction

Next generation 5G networks are being deployed at millimeter-wave (mmWave) bands, beyond 22 GHz [1]. Such high frequencies enable data rates in the order of gigabits per second due to the availability of large unlicensed bandwidth. This is especially beneficial to the future highly dense Internet-of-Things (IoT) networks, which demand large bandwidth. Further, mmWave systems have small form factor and are strong candidates for the emerging intelligent surfaces for IoT devices. Recent studies showed that mmWave antennas can be designed in a flexible and conformal manner [2–4], making them suitable for wearables.

Although mmWave bands allow for high data rate, the short wavelengths are heavily attenuated by the environment [5] mostly due to absorption. Thus, transmitted signals experience severe path loss. To combat the high path loss, mmWave antenna arrays with beam-forming features are being used for generating directional beams which attains longer communication range. However, the directionality of the mmWave beams brings difficulty in mobile settings as they need constant alignment on both the mobile transmitter and receiver nodes [6]. mmWave channels can be quite complex as line-of-sight (LoS) and non-LoS (NLoS) signals can exist due to emphasized environmental effects. Characterizing mmWave channels, tracking mmWave beams, and Angle-of-arrival (AoA)

detection have been challenging [7]. Handling this complexity requires the future mmWave systems to be highly integrated with software-defined radio (SDR) platforms, where advanced algorithmic methods can be practiced.

AoA detection [8] is a critical capability that can facilitate better alignment of the mmWave beams. It is an important directional wireless capability that is used to detect signals transmitted in the environment [9]. A good estimation of the AoA enables fine tuning of the beam alignment between the transmit and receive antennas, which leads to more accurate channel state information (CSI). As a result, the received signal strength (RSS) increases, which leads to a better overall signal-to-noise ratio (SNR) and link performance.

AoA detection has been studied extensively over the years. Numerous algorithms have been used to estimate AoA using synthetic data [9]. Deep learning has been the preferred machine learning (ML) choice for AoA detection, due to robustness to environmental noise. Other methods have been shown to perform poorly in estimating AoA in noisy environments [9]. However, the deep learning methods require extensive training which is not suitable for IoT devices operating in a highly dynamic environment with almost constantly changing channel behavior. More importantly, deep learning methods may require large memory which does not fit well with hardware-constrained IoT devices like wearables.

For mmWave AoA detection, we adapt unsupervised reinforcement learning (RL) algorithms to avoid the abovementioned complexities of deep learning. Our RL-based approach to detecting AoA is compatible with mmWave SDR systems as we show it by implementation. Our approach only considers the receiver side and can passively detect AoA without help from the transmitter or any other localization system. We utilize two RL methods, Q-learning and its variant double Q-learning, for AoA detection. Q-based learning algorithms are widely used for a wide variety of applications that require fast learning capability, such as in gaming, or fast detection capability, such as detecting a drone flying through an indoor environment [10].

Our RL algorithms follow a Markovian model, using actions to explore different states of a given environment [11, 12]. The actions can be based off a greedy policy. With this type of policy, the algorithms can use prior information learned to select the best actions to take [10]. Positive actions lead to positive rewards, while negative actions are punished with negative rewards. After a certain number of iterations, the algorithms learn which actions lead to the best states and converge to a solution. The algorithms use the Bellman equation, discussed later in Section 4. Beyond the observed reward for taken actions, the equation relies on several tunable input parameters: the learning rate α, the discount factor γ, and the exploration policy ϵ. It has been shown that tuning the learning rate α and exploration policy ϵ can lead to optimum solutions [12–14]. For our study we measured the accuracy of detected AoA and convergence time, by tuning both the learning rate α and exploration policy ϵ. Further, the algorithms take a certain number of unknown iterations to converge. To tackle this problem, we use a threshold on the coefficient of variation (CoV) of RSS data samples as the criteria to detect convergence. We compare the performance of our algorithms

using Horn antennas controlled by a Pyhton-based SDR setup in connection with GNU radio [15].

Our main contributions are as follows:

- Adaptation of Q-learning and Double Q-learning methods for mmWave AoA detection.
- Tuning the hyper parameters (the learning rate α and exploration policy ϵ) of both Q- and Double Q-learning to detect AoA within 2^o's of accuracy.
- Design of a threshold-based convergence criteria for both Q- and Double Q-learning using CoV of RSS data samples.
- Implementation of a prototype of the algorithms in an affordable mmWave testbed platform using an off-the-shelf SDR platform.

The rest of the paper is organized as follows: Section 2 surveys the related literature on AoA detection and experimental mmWave SDR efforts. Section 3 presents our experimental platform and how the AoA detection algorithms are implemented in that platform. Next, Section 4 provides a detailed description of our Q-learning and Double Q-learning algorithms for AoA detection. Section 5 details experimental setup and discusses results from our experiments. Finally, Section 6 summarizes our work and outlines directions of future work.

2 Related Work

Angle (or direction) of arrival (AoA) detection/estimation has been an extensively studied problem within the context of wireless localization [16]. With the recent advent of directional beam-forming capabilities in super-6 GHz systems, AoA detection, in particular, has gained a renewed interest due to emerging applications using such systems [17].

Experimental demonstration and evaluation of AoA detection in super-6 GHz bands such as mmWave bands has been lacking. The main reason for this has been the limited availability of mmWave experimental testbeds due to the lack and high cost of mmWave hardware [18]. The U.S. National Science Foundation (NSF) is currently funding wireless communication testbed platforms to enable such experimentation. The COSMOS platform [19], for example, includes a 28 GHz phased array antenna, designed by IBM. The front end uses a custom software for steering the antenna beam with respect to azimuth and elevation angles. The AERPAW platform uses drones for 5G experimentation [20], which is the first of its kind. These platforms enable users to perform a variety of wireless communication experiments, such as, AoA detection. However, they are still being adapted by researchers. Unlike these high-end testbeds, we use a cheap SDR platform and mmWave hardware to evaluate our AoA detection mechanisms. Further, the application programming interface (API) used by these testbed platforms can limit user experimentation. For example, the AERPAW API restricts users from running on the fly experiments. As a result, users aren't able to collect or train radio frequency (RF) data on the fly. This can restrict the types of algorithms users can use on the platform.

Researchers have relied on virtual environments and simulations to perform mmWave experiments. These virtual environments have gotten more sophisticated with the usage of 3D ray tracing. In [21], 3D ray tracing is used to simulate mmWave signals in virtual environments. Users can use the open source software to design large intelligent reflective surfaces and determine AoA using compressive sensing algorithms. Although using simulation-based approaches is cost effective, they do not render the physical world and fall short of precisely modeling complicated physical communication channel dynamics in mmWave or other super-6 GHz bands.

Recently, cheaper off-the-shelf SDRs have been used to setup testbed platforms for AoA detection. The testbed platform [9] uses a Kerberos radio with four whip antennas at the receiving end. At the transmitting end a long range (LoRa) radio is used to transmit a signal at 826 MHz. LoRa is beneficial for long range communication and uses low transmit power. The transmitter includes a GPS and compass unit used to label the direction of the transmitted signal. The labeled data set is the ground truth that is trained in the machine learning (ML) algorithm. The data is trained using a deep learning convolutional neural network (CNN) model [9].

Multiple Signal Classification (MUSIC) is a widely used AoA detection algorithm and assumes that the received signal is orthogonal to the noise signal [9]. MUSIC uses this assumption to decompose the noise from the received signal into separate sub-spaces. The power spectral density (PSD) of the signal is taken as a function of angle [22]. The angular value which results in the maximum power is estimated to be the AoA. The assumption that the received signal is orthogonal to the noise signal is not valid in real world scenarios. Therefore, MUSIC does poorly in environments that involve NLoS propagation. Since mmWave signals can experience severe environmental path loss and involve multiple NLoS signals, MUSIC may not be a good choice for mmWave AoA detection.

Support Vector Regression (SVR) has also been used to estimate AoA. SVR is a supervised ML algorithm. Regression does poorly in estimating AoA from impinging signals at multiple time steps [9]. The algorithm cannot be used to determine AoA since the number of impinging signals is unknown [23]. As a result, the algorithm can be used for detecting AoA for a single source at a time. This makes SVR less robust for AoA detection in environments with multiple signal sources. Therefore, SVR is not a good choice for mmWave AoA detection.

The CNN model used in [9] adapts a hybrid configuration. A classification method is used to determine the number of impinging receive signals and two regressive heads are used to determine the AoA. The study showed that CNN outperformed the other classical ML methods, MUSIC and SVR. Further, the CNN model was able to estimate AoA within 2^o's of accuracy.

Our approach does not use a deep learning approach or supervised learning. These approaches are not the most suitable for many hardware-constrained IoT devices as the former requires large memory hardware to perform well and the latter requires availability of ground truth. Resources-constrained IoT devices like wearables do not have sufficient memory to keep trained models nor the

extensive sensing or coordination capability to obtain the ground truth in AoA. To make it more relevant to IoT devices with high resource constraints, we design RL-based AoA detection methods that do not require the ground truth and determine the AoA based only on the RSS observations at the receiver.

3 mmWave Testbed

To perform a thorough evaluation of our reinforcement learning (RL)-based AoA detection methods, we use our mmWave testbed [24] that allows beam-steering capability from Python.

3.1 Hardware Setup

The architecture of our testbed can be seen in Figure 1. The testbed uses a Universal Software Radio Peripheral (USRP) model N210. The USRP uses a Superhterodyne architecture for up-converting the transmit signal and down-converting the receive signal [25]. The architecture is built into the USRP's daughter-board to tune the signal within sub-6 GHz [25]. As a result, the USRP is only able to transmit and receive signals at a maximum frequency of 6 GHz.

To handle mmWave frequencies we connect the daughter-board to external RF mixers to further up/down convert the signal. We use Analog Devices up-converter ADMV 1013 [26] and down-converter ADMV 1014 [27]. The cost of each unit is reasonably priced at a few hundred dollars. This makes our mmWave testbed platform more affordable, compared to [18] and [19] that use RF front-ends that cost thousands of dollars. Two signal generators are used as local oscillators for mixing the signal. Two 26 GHz mmWave 15 dB gain horn antennas are used for transmission and reception. The receive horn antenna is mounted to a servo that can rotate from 0 to 180 degrees. A pulse width modulated (PWM) signal is transmitted from the Arduino micro-controller to rotate the servo at a set angular value.

3.2 Software Setup

GNU radio software is used to program the USRP device. GNU radio is a Python-based graphical interface that is open source and readily available online. As seen in 2 [24], the source block is used to generate a cosine signal at a sampling frequency of 2.5 MHz. The samples are streamed in the USRP sink block that sets the frequency of the signal to 2 GHz. The signal is then transmitted from the USRP daughter-board. The receive signal is mixed down to 2 GHz and injected into the daughter-board. The I and Q base-band data samples are streamed from the USRP source block. The samples are used to determine the RSS using the complex to mag-square block. The RSS samples are streamed into a socket using the ZMQ pub sink block. Python socket libraries are used to connect and receive the RSS data from the socket.

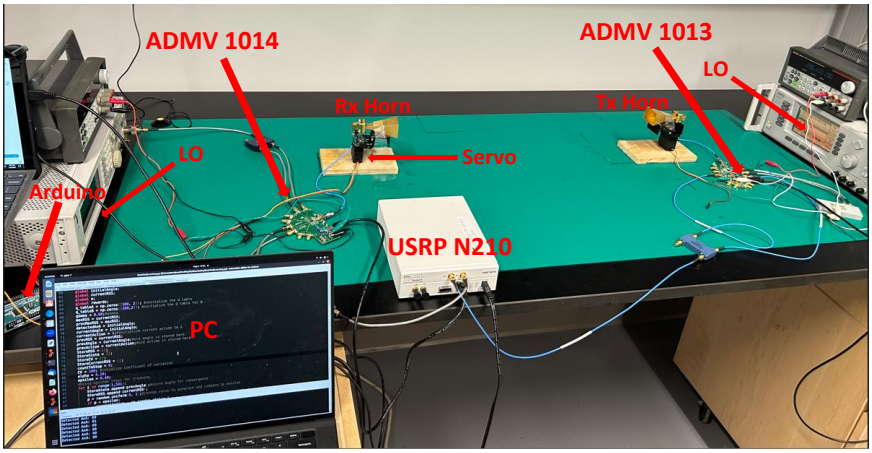

Fig. 1: Testbed Configuration

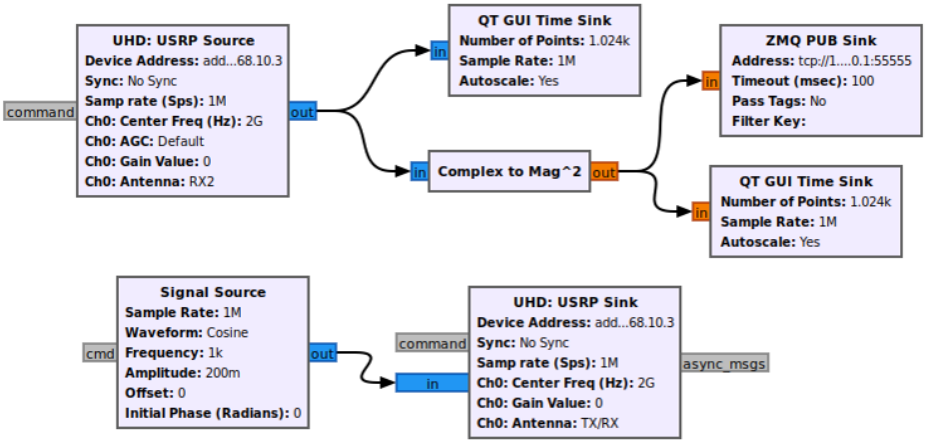

Fig. 2: GNU Radio Software Configuration

4 Q-learning and Double Q-Learning

Q-learning and its variant Double Q-learning are model-free RL algorithms. Both algorithms are based on a Markovian approach that selects random actions [11]. The block diagram in Figure 3 presents our RL model. The learning agent is located at the receiver horn antenna that is mounted onto a servo motor that can be steered. The agent can choose to take two possible actions: move left or right by one degree resolution. Since the servo can rotate up to a maximum angular value of 180 degrees, the time-variant state values can be $s_t \in (0, 180)$.

For our setup, a positive reward (i.e., when the action improves the RSS) is set as the difference of the current RSS_t and previous RSS_{t-1}, i.e., $RSS_t - RSS_{t-1}$. A negative reward (i.e., when the action reduces the RSS) is set to -5. This design incentivizes the agent to seek the angular position that maximizes the RSS, which is implied when the antenna is steered to the correct AoA.

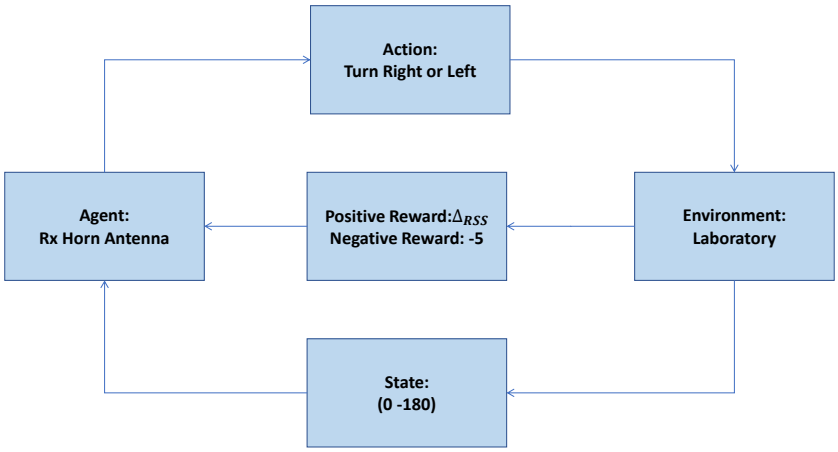

Fig. 3: RL Configuration

Q-learning uses the Bellman equation to populate the Q-table. The Bellman equation is

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha * (r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)) \quad (1)$$

which provides a mechanism to update the Q values as a function of state, action, and tunable parameters [10]. The variable s_t is our current state and a_t is our current action. When the agent performs an action the algorithm moves to the next state defined as s_{t+1} and a reward r_t is given. For our algorithm, the discount factor γ is set to 0.98 and the learning rate $\alpha \in (0, 1)$ was tuned to 0.1, 0.2, 0.3, and 0.4. Likewise, the greedy policy ϵ was tuned to the same values. Smaller values of ϵ cause the algorithm to exploit the Q table, by searching for the action that results in the largest Q value. Increasing ϵ increases the likelihood that the algorithm will explore the environment, by selecting random actions.

Our software approach is presented in Algorithm 1. At the start of each run the hyper-parameters γ, α, and ϵ are initialized. The current state s_t is initialized to a random angular value, based off our experimental scenario. For the 90^o experimental setup the random angle can be any value in [80,100] degrees and [120,140] degrees for the 130^o. The Q_{table} used to store the Q values from equation 1 is initialized to zero. An outer while loop comparing the *threshold* value and CoV is used to decide the stopping condition for the algorithm until the threshold value is met. If the value of ϵ is greater than a random number

Algorithm 1 AoA Detection with Q-learning

Input: $\alpha \in (0,1)$, the learning rate; $\epsilon \in (0,1)$, the greediness policy; $windowSize \in (5,30)$, the number RSS samples maintained for implementing convergence criterion; $threshold \in (0.1, 0.4)$ threshold for convergence criterion.
Output: $detectedAoA$, AoA detected by the algorithm.
$\gamma \leftarrow 0.98$ ▷ Initialize the discount factor
$RSS[windowSize] \leftarrow [\,]$ ▷ Initialize the array of RSS samples
$CoV \leftarrow 1$ ▷ Initialize the coefficient of variation of RSS samples
Randomize $currentAngle$ ▷ Initialize to a random angle respectively in [80,100] or [120,140] for the 90^o and 130^o scenarios
$s_t \leftarrow currentAngle$ ▷ Initialize the current state to $currentAngle$
$previousRSS \leftarrow$ Measure RSS at $currentAngle$
$RSS.append(previousRSS)$ ▷ Store RSS sample in array
$sampleCount \leftarrow 1$
$Q_{table} \leftarrow 0$ ▷ Initialize the Q-table to 0
while $threshold \leq CoV$ **do**
    $RSS.append(RSS_t)$ ▷ Store RSS data in array
    if Uniform(0,1)$< \epsilon$ **then**
        $a_t \leftarrow$ Uniform[0,1] ▷ Randomly choose to turn left (a_t=0) or right (a_t=0)
    else
        $a_t \leftarrow \max_a Q(s_t, a)$ ▷ Choose action based on maximum Q value
    end if
    if $a_t == 0$ **then**
        Turn antenna beam left by 1 degree
        $currentAngle - -$;
    else
        Turn antenna beam right by 1 degree
        $currentAngle + +$;
    end if
    $newRSS \leftarrow$ Measure RSS at $currentAngle$
    $RSS.append(newRSS)$ ▷ Store the new RSS sample and remove the oldest sample if needed
    $sampleCount + +$;
    $s_{t+1} \leftarrow currentAngle$
    $\Delta RSS \leftarrow newRSS - previousRSS$ ▷ Calculate the reward
    if $0 < \Delta RSS$ **then**
        $r_t \leftarrow \Delta RSS$ ▷ Positive reward
    else
        $r_t \leftarrow -5$ ▷ Negative reward
    end if
    $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * (r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$ ▷ Bellman eqn.
    $s_t \leftarrow s_{t+1}$ ▷ Update current state with next state value
    if $windowSize \leq sampleCount$ **then**
        $CoV \leftarrow RSS.std()/RSS.mean()$ ▷ Compute the CoV of RSS samples
    end if
end while
return s_t

in (0,1), the agent takes a random action, either turn the antenna using the servo left or right with one degree resolution. If the value of ϵ is greater than the random value, then the algorithm will exploit the Q_{table} by selecting the action that results in the largest Q value. A positive reward value is given if ΔRSS is greater than zero. If ΔRSS is less than zero, then the action resulted in a decrease in RSS. Therefore, the action yields a negative reward value of -5. The Bellman equation is updated with the reward at every iteration.

Q-learning based algorithms have to train for a certain number of iterations. Based on the number of iterations, the algorithm may or may not converge to the solution. This makes selecting the number of iterations trivial. To address this issue, we take CoV of a certain number of RSS samples (defined by $windowSize$) to detect convergence. CoV is a statistical measure of how dispersed data samples are from the mean of the sample space. It is the ratio of standard deviation and mean of a certain number of samples. In our algorithm, a window size is initialized to $windowSize \in (5, 30)$. The array RSS is used to store RSS samples as the algorithm is training on the fly. If the counter $sampleCount$ is greater or equal to the $windowSize$, then enough samples have been collected to calculate the CoV. While the CoV is greater or equal to the selected threshold value, the algorithm will continue to train. When CoV is less then or equal to the threshold, the convergence criteria is met and the algorithm breaks out of the while loop. The smaller CoV means that the RSS samples have stabilized and it is safer to stop the algorithm and return the last steering angle as the detected AoA.

Q-learning uses a single estimator $\max_a Q(s_{t+1}, a)$, the maximum next state Q_{t+1} value for all possible actions. As shown in [12], this causes the algorithm to overestimate the desired solution, by causing a positive bias. As a result, standard Q-learning can perform poorly in certain stochastic environments. To improve the performance of standard Q-learning other variants, such as, Double Q-learning [12], Delayed Q-learning [28], Fitted Q-iteration [29], and Phased Q-learning [30] were developed to improve convergence time. To reduce overestimation, the double Q-learning variant uses two Q functions $Q^A(s_t, a_t)$ and $Q^B(s_t, a_t)$ seen in equations 2 and 3 below. $Q^A(s_t, a_t)$ is able to learn from

$$Q^A(s_t, a_t) = Q^A(s_t, a_t) + \alpha * (r_t + \gamma Q^B(s_{t+1}, \arg\max_a Q^A(s_{t+1}, a)) - Q^A(s_t, a_t)) \quad (2)$$

$$Q^B(s_t, a_t) = Q^B(s_t, a_t) + \alpha * (r_t + \gamma Q^A(s_{t+1}, \arg\max_a Q^B(s_{t+1}, a)) - Q^B(s_t, a_t)) \quad (3)$$

the experiences of $Q^B(s_t, a_t)$ and vice versa. This approach has been shown in [12] to cause the algorithm to underestimate, rather than overestimate the solution with a positive bias. In conditions where Q-learning performs poorly, double Q-learning has been shown to converge to the optimum solution [12].

Our AoA detection algorithm for Double Q-learning can be seen in Algorithm 2. Like algorithm 1, the same parameters are initialized at the start of each run. Two Q tables Q^Atable and Q^Btable are initialized to zero. The same greedy ϵ and reward r_t values are also used in Algorithm 2. The variable q is set equal to a random uniform value $q \in (0, 1)$. If q is larger than the threshold value of 0.5

Algorithm 2 Double Q-learning

Same as lines 1-10 in Algorithm 1
$Q^A_{table} \leftarrow 0$ ▷ Initialize the first Q-table to 0
$Q^B_{table} \leftarrow 0$ ▷ Initialize the second Q-table to 0
while $threshold <= CoV$ **do**
  $RSS.append(previousRSS)$ ▷ Store RSS sample in array
  if Uniform(0,1)$< \epsilon$ **then**
    $a_t \leftarrow$ Uniform[0,1] ▷ Randomly choose to turn left (a_t=0) or right (a_t=0)
  else
    $Q^{AB}_{table} \leftarrow Q^A_{table}(s_t) + Q^B_{table}(s_t)$
    $a_t \leftarrow \max_a Q^{AB}_{table}$
  end if
  if $a_t == 0$ **then**
    Turn antenna beam left by 1 degree
    $currentAngle--$;
  else
    Turn antenna beam right by 1 degree
    $currentAngle++$;
  end if
  $newRSS \leftarrow$ Measure RSS at $currentAngle$
  $RSS.append(newRSS)$ ▷ Store the new RSS sample and remove the oldest sample if needed
  $sampleCount++$;
  $s_{t+1} \leftarrow currentAngle$
  $\Delta RSS \leftarrow newRSS - previousRSS$ ▷ Calculate the reward
  if $0 < \Delta RSS$ **then**
    $r_t \leftarrow \Delta RSS$ ▷ Positive reward
  else
    $r_t \leftarrow -5$ ▷ Negative reward
  end if
  $q \leftarrow$ Uniform(0,1)
  if $q < 0.5$ **then**
    $Q^A(s_t, a_t) = Q^A(s_t, a_t) + \alpha * (r_t + \gamma Q^B(s_{t+1}, \arg\max_a Q^A(s_{t+1}, a)) - Q^A(s_t, a_t))$
  else
    $Q^B(s_t, a_t) = Q^B(s_t, a_t) + \alpha * (r_t + \gamma Q^A(s_{t+1}, \arg\max_a Q^B(s_{t+1}, a)) - Q^B(s_t, a_t))$
  end if
  $s_t \leftarrow s_{t+1}$ ▷ Update current state with next state value
  if $windowSize \leq sampleCount$ **then**
    $CoV \leftarrow RSS.std()/RSS.mean()$ ▷ Compute the CoV of RSS samples
  end if
end while

then $Q^A(s_t, a_t)$. Otherwise, if q is less than the threshold of 0.5, then $Q^B(s_t, a_t)$ is selected. The threshold value is set to 0.5 to give both $Q^A(s_t, a_t)$ and $Q^B(s_t, a_t)$ equal probability of being selected for training. The same convergence criteria is used in Algorithm 1 is used in 2.

5 Experimental Evaluation and Results

To understand the efficacy of our RL-based AoA detection algorithms, we measured the average AoA error and average convergence time for different (ϵ,α) combinations at each *Threshold* value. The *Threshold* value determines how strict the convergence criterion is. Hence, smaller *Threshold* causes the AoA detection algorithms to search the AoA for a longer period of time. This enables them to find a more accurate AoA. Hence, there is a trade-off between the error in AoA estimate and the convergence time of the algorithms. For a fixed *Threshold* value, we need to find the best (ϵ,α) combination by minimizing both the AoA error and convergence time. To do so, we search for the (ϵ,α) combination that minimizes the product of the two metrics, i.e.:

$$\min_{\epsilon,\alpha} \quad <\text{AoA Error}> * <\text{Convergence Time}>. \tag{4}$$

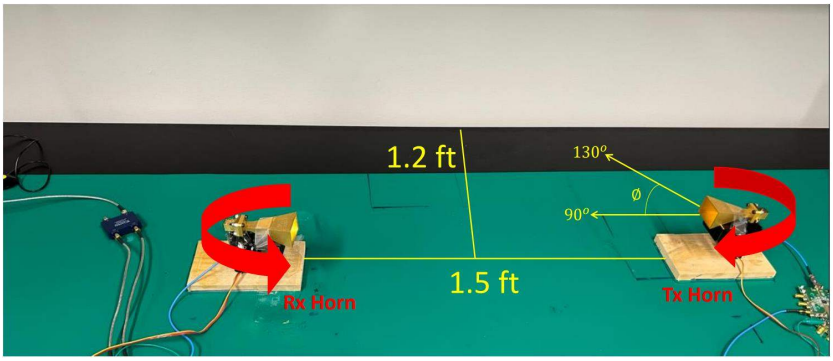

Fig. 4: Experimental Set-Up

We considered two scenarios for comparing the performance of the Algorithms 1 and 2. As seen in Figure 4, the range between the transmit and receive horn antennas is 1.5 ft. The distance to the wall to the center is 1.2 ft. In the first experiment scenario, the transmit antenna is fixed to 90^o, pointing towards the receive antenna to compose an LoS path to the receiver. The receive horn antenna is initialized to a random angular state value between 80 and 100 degrees. For the second scenario, the transmit horn antenna is rotated 130^o to the right, composing a NLoS path to the receiver. The main lobe of the transmitted signal is reflected off the wall. The initial angle is set to a random value between 120 and 140 degrees.

As previously mentioned, the hyper-parameters ϵ and α are tuned to 0.1, 0.2, 0.3, and 0.4. This was done to measure which combinations of (ϵ,α) resulted in the best performance with respect to both AoA detection and convergence time. The average AoA error and time of convergence was measured for thirteen threshold values from 0.1 to 0.4 in increments of 0.025. This was done using Q and double Q-learning for both the 90^o and 130^o experimental scenarios.

5.1 Q-Learning Results

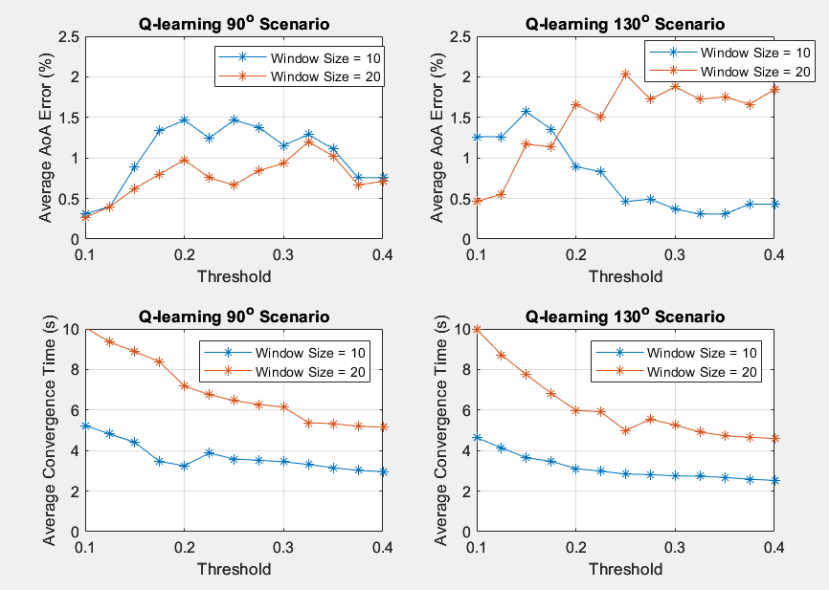

Fig. 5: Q-Learning for 90^o and 130^o scenarios

Figure 5 are graphs of *Threshold* vs. average AoA error and *Threshold* vs. average convergence time. For each combination of (ϵ, α) the average convergence time is reduced with respect to window size. A window size of 10 results in faster AoA detection compared to the window size of 20.

Both the 90^o and 130^o degree scenarios resulted in average AoA error less than 1% for some combination of (ϵ, α).This occurs for both window sizes of 10 and 20. With less than 1% AoA error, the detected AoA is within 1^o of the correct AoA.

To understand which (ϵ,α) combinations give the best results, we look at the heat map of (α, ϵ) occurrences resulting in the minimum product in equation 4 as shown in Figure 6. For the 90^o case, ϵ of 0.4 with α of 0.1, 0.2, and 0.4 are the dominate cases. Likewise, ϵ of 0.4 is also the dominant case for the 130^o scenario. The α values are also similar at 0.2 and 0.4 being the dominant cases.

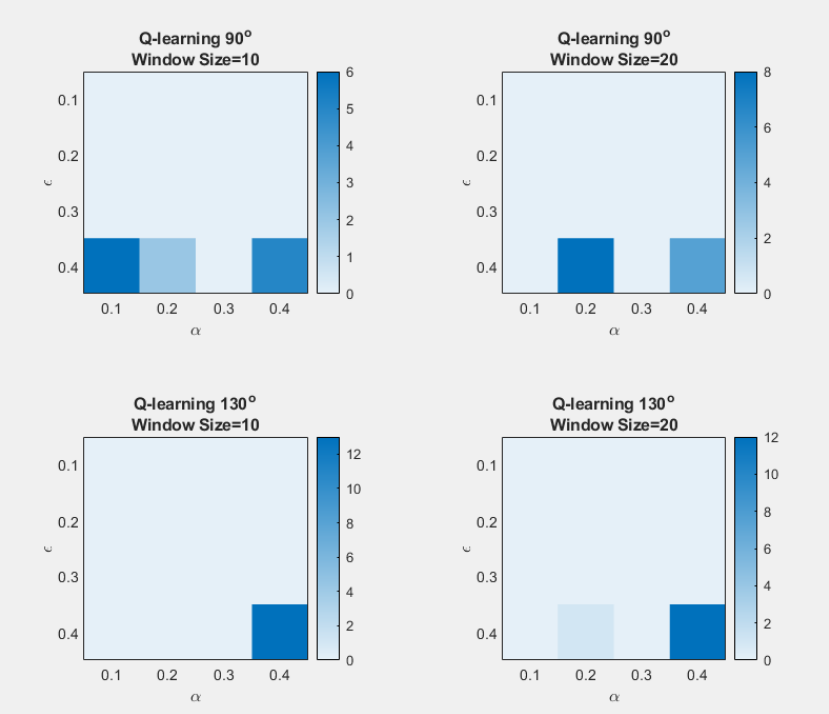

Fig. 6: Q-Learning Heat Map

For Q-learning the combination of (ϵ,α) that are most effective are (0.4,0.1), (0.4,0.2) and (0.4,0.4).

5.2 Double Q-Learning Results

Figure 7 shows the average AoA error and average convergence time against *Threshold* for Double Q-learning. Like Q-learning, it is clear that an increase in window size results in larger convergence time. For Double Q-learning, both the average AoA error and convergence time are better than Q-learning. For nearly all (ϵ, α) combinations, the average AoA error is less than 0.15% for both the 90^o and 130^o degree scenarios.

Figure 8 shows the heat map of (α, ϵ) occurrences resulting in the minimum product in equation 4 for Double Q-learning. For the 90^o scenario the (ϵ, α) combinations that occur the most often are (0.1,0.1), (0.1,0.3) and (0.1,0.4) for both window sizes. Like the 90^o case, it can also be seen that an ϵ of 0.1 does occur for the 130^o scenario for a window size of 10. However, we also see that ϵ values of 0.3 and 0.4 do occur for both window sizes. The dominant (ϵ, α) are (0.1,0.1), (0.1,0.2), (0.2,0.3), (0.3,0.2), (0.4,0.2), and (0.4,0.3).

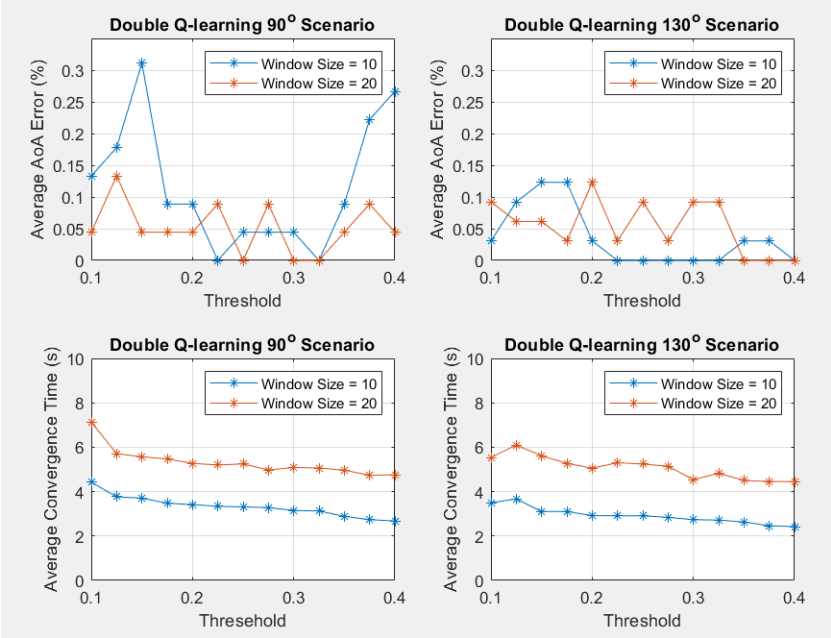

Fig. 7: Double Q-Learning for 90^o and 130^o scenarios

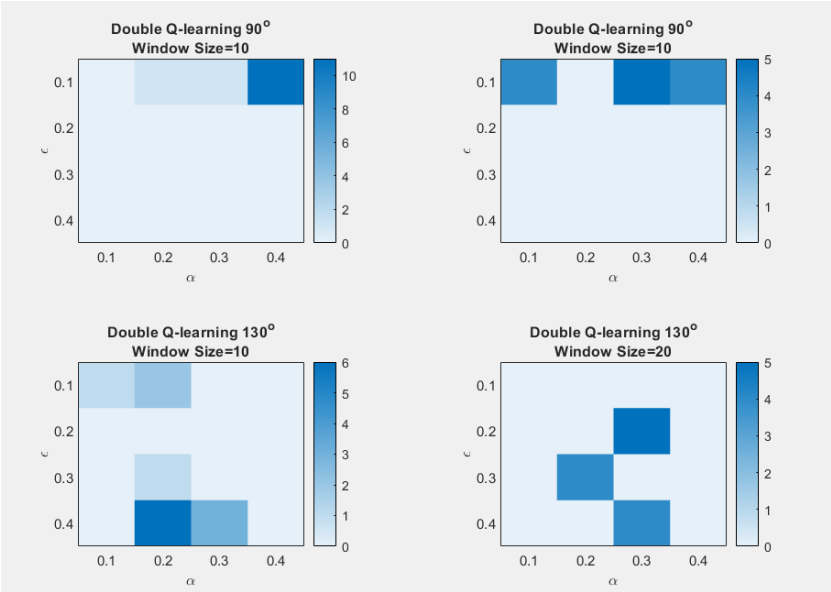

Fig. 8: Double Q-Learning Heat Map

6 Conclusion and Future Work

We presented RL-based AoA detection algorithms for mmWave systems that are operated by SDR. By adapting Q-learning and Double Q-learning to the AoA detection problem, we demonstrated the practicality of the approach and experimentally evaluated the methods in a mmWave SDR testbed. We achieved AoA detection within 2^o of the correct AoA accuracy using the RL algorithms Q- and Double Q-learning. Compared to [9], our current setup uses unsupervised learning and does not rely on labeling data sets. The setup in [9] uses GPS to label transmitted signals. GPS does poorly in indoor environments, due to low power of reception. Further, for the RL algorithms Q- and Double Q-learning, our study investigated the best combinations of hyper-parameters that minimizes the AoA detection error and convergence time of the algorithms. We showed that double Q-learning outperforms Q-learning with respect to both AoA accuracy and convergence time. Compared to [18] and [19], our mmW platform is much cheaper and allows for a more user friendly interface.

Neural networks aren't used in our Q- and Double Q-learning algorithms. We plan on implementing and testing deep learning methods utilizing neural networks for AoA detection. With the usage of cheap servos, our current system set-up is limited. We plan to equip our testbed with phased array antennas [6], which can steer antenna beams in the order of microseconds. This will result in much faster convergence time and enable more flexible beamsteering enlarging the action space for the learning algorithms. We also plan to integrate field programmable gate array (FPGA) to our setup, in order to improve hardware and software run-time. An improvement in run-time will result in better overall convergence time.

References

1. S. Seth, M. Yuksel, and A. Vosoughi, "Forming coalition sets from directional radios," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022, pp. 507–514.
2. S. F. Jilani, M. O. Munoz, Q. H. Abbasi, and A. Alomainy, "Millimeter-wave liquid crystal polymer based conformal antenna array for 5G applications," *IEEE Antennas and Wireless Propagation Letters*, vol. 18, no. 1, pp. 84–88, 2019.
3. T. Dhruva, A. Rajesh, and K. Abirami, "Design of conformal multi-band mmWave wearable antenna for 5G applications," in *Proceedings of IEEE International Conference on Smart Electronics and Communication (ICOSEC)*, 2022, pp. 573–577.
4. K. Hu, Y. Zhou, S. K. Sitaraman, and M. M. Tentzeris, "Fully additively manufactured flexible dual-band slotted patch antenna for 5G/mmWave wearable applications," in *Proceedings of IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (AP-S/URSI)*, 2022, pp. 878–879.
5. T. Bogale, X. Wang, and L. Le, "Chapter 9 - mmwave communication enabling techniques for 5g wireless systems: A link level perspective," in *mmWave Massive MIMO*, S. Mumtaz, J. Rodriguez, and L. Dai, Eds. Academic Press, 2017, pp. 195–225. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128044186000091

6. M. Jean, E. Velazquez, X. Gong, and M. Yuksel, "A 30 ghz steerable patch array antenna for software-defined radio platforms," in *SoutheastCon 2023*, 2023, pp. 856–860.
7. W. Fan, Y. Xia, C. Li, and Y. Huang, "Channel tracking and aoa acquisition in quantized millimeter wave MIMO systems," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2023.
8. H. Yazdani, S. Seth, A. Vosoughi, and M. Yuksel, "Throughput-optimal d2d mmwave communication: Joint coalition formation, power, and beam optimization," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 1539–1544.
9. Z. Dai, Y. He, T. Vu, N. Trigoni, and A. Markham, "Deepaoanet: Learning angle of arrival from software defined radios with deep neural networks," 2021.
10. M. M. U. Chowdhury, F. Erden, and I. Guvenc, "Rss-based q-learning for indoor uav navigation," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, 2019, pp. 121–126.
11. C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
12. H. Van Hasselt, "Double q-learning." 01 2010, pp. 2613–2621.
13. T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Computation*, vol. 6, no. 6, pp. 1185–1201, 1994.
14. J. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," in *Proceedings of 32nd IEEE Conference on Decision and Control*, 1993, pp. 395–400 vol.1.
15. "GNU Radio - the free and open source radio ecosystem," GNU Radio. [Online]. Available: https://www.gnuradio.org
16. F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.
17. K. Wu, W. Ni, T. Su, R. P. Liu, and Y. J. Guo, "Recent breakthroughs on angle-of-arrival estimation for millimeter-wave high-speed railway communication," *IEEE Communications Magazine*, vol. 57, no. 9, pp. 57–63, 2019.
18. A. Şahin, M. L. Sichitiu, and İ. Guvenç, "A Millimeter-Wave Software-Defined Radio for Wireless Experimentation," *arXiv e-prints*, p. arXiv:2302.08444, Feb. 2023.
19. X. Gu, D. Liu, C. Baks, O. Tageman, B. Sadhu, J. Hallin, L. Rexberg, and A. Valdes-Garcia, "A multilayer organic package with 64 dual-polarized antennas for 28ghz 5g communication," in *2017 IEEE MTT-S International Microwave Symposium (IMS)*, 2017, pp. 1899–1901.
20. "Aerpaw: Aerial experimentation and research platform for advanced wireless," https://aerpaw.org/.
21. A. Alkhateeb, "Deepmimo: A generic deep learning dataset for millimeter wave and massive MIMO applications," *CoRR*, vol. abs/1902.06435, 2019. [Online]. Available: http://arxiv.org/abs/1902.06435
22. M. Kaveh and A. Barabell, "The statistical performance of the music and the minimum-norm algorithms in resolving plane waves in noise," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 2, pp. 331–341, 1986.
23. M. Pastorino and A. Randazzo, "A smart antenna system for direction of arrival estimation based on a support vector regression," *IEEE Transactions on Antennas and Propagation*, vol. 53, pp. 2161–2168, 2005.

24. M. Jean, M. Yuksel, and X. Gong, "Millimeter-wave software-defined radio testbed with programmable directionality," in *Proceedings IEEE INFOCOM Workshop 2023*, in press.
25. "UBX 10-6000 MHz Rx/TX (40 MHz, N series and X series)," Ettus Research, a National Instruments Brand. [Online]. Available: https://www.ettus.com/all-products/ubx40
26. "ADMV1013," Analog Devices. [Online]. Available: https://www.analog.com/en/products/admv1013.html
27. "ADMV1014," Analog Devices. [Online]. Available: https://www.analog.com/en/products/admv1014.html
28. M. Kearns and S. Singh, "Finite-sample convergence rates for q-learning and indirect algorithms," in *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*. Cambridge, MA, USA: MIT Press, 1999, p. 996–1002.
29. D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J. Mach. Learn. Res.*, vol. 6, p. 503–556, dec 2005.
30. C. Szepesvári, "The asymptotic convergence-rate of q-learning," in *Proceedings of the 10th International Conference on Neural Information Processing Systems*, ser. NIPS'97. Cambridge, MA, USA: MIT Press, 1997, p. 1064–1070.