Creation of a Cognitive Radar with Machine Learning: Simulation and Implementation

Mark A. Kozy

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Richard M. Buehrer, Chair
Jeffrey H. Reed
John M. Ruohoniemi

March 23, 2019
Blacksburg, Virginia

Keywords: Cognitive radar, machine learning, reinforcement learning, tracking radar, software defined radio

Creation of a Cognitive Radar with Machine Learning: Simulation and Implementation

Mark A. Kozy

(ABSTRACT)

In this thesis we address radar-communication coexistence by modelling the radar environment as a Markov Decision Process (MDP), and then apply Deep-Q Learning to optimize radar performance. The radar environment includes a single point target and a communications system that will potentially interfere with the radar. We demonstrate that the Deep-Q Network (DQN) we construct is able to successfully avoid interfering with the communication system to improve its performance. We also show that the DQN method outperforms previous methods in terms of incorporating memory and handling new situations. In this thesis we also address the application of the MDP into a software defined radio (SDR) USRP X310 by utilizing the software LabVIEW to communicate with and control the SDR.

Creation of a Cognitive Radar with Machine Learning: Simulation and Implementation

Mark A. Kozy

(GENERAL AUDIENCE ABSTRACT)

In this thesis we develop methods for creating and implementing algorithms for a cognitive radar. A cognitive radar is a radar that is able to sense its environment and avoid any other communication system that may interfere with its operation. We discuss the predictive methods we used to sense and avoid the other communication systems as well as how we implemented this using a software defined radar based on the USRP X310.

Acknowledgments

I would like to thank the U.S Army Research Laboratory (ARL) for their sponsorship of this project. I would specifically like to thank Anthony Martone and Kelly Sherbondy for their guidance and mentorship throughout the duration of this project.

I would also like to thank everyone in Wireless@VT for their support in this project. Finally, I would like to recognize and thank my advisor Dr. R. Michael Buehrer whose help was instrumental in the success of this project.

Contents

List of Figures

List of Tables

List of Abbreviations

DQN Deep-Q Network

EM Electromagnetic

FCC Federal Communications Commission

LFM Linear Frequency Modulated

MDP Markov Decision Process

PRF Pulse Repetition Frequency

RFI Radio Frequency Interference

SDR Software Defined Radio

SINR Signal to Interference and Noise Ratio

USRP Universal Software Radio Peripheral

Chapter 1

Introduction

1.1 Executive Summary

One of the most precious commodities for radio applications is the electromagnetic (EM) spectrum [1]. This is especially important for radar applications since the spectrum is being opened for sharing/coexistence between systems, and the potential for interference continues to grow as the EM spectrum becomes more and more congested [2]. Radar transmissions may directly or indirectly interfere with communication systems and vice versa, thereby degrading radar performance [1]. Cognitive radar, first introduced by Haykin [3], has the potential to mitigate this problem and can readily be added to modern digital radars [4].

We say that a system is cognitive if it is capable of four fundamental functions that are basic to human cognition [5]:

1. The perception-action cycle
2. Memory
3. Attention
4. Intelligence

The perception-action cycle implies that the cognitive system has two functional parts: the

perceptor and actuator [5]. The cycle begins by the system observing the environment and using this information to choose an action based upon that information. The benefit resulting from the perception-action cycle is the maximization of information gained from the environment [5]. In the case of the cognitive radar problem we are investigating, the environment that is being observed is the frequency spectrum and the action that must be taken is choosing a sub band(s) for the radar to operate.

The model used by previous research [6] was the Markov Decision Process (MDP) with reinforcement learning in order to identify which actions were optimal so as to minimize mutual interference with communication systems. MDPs are widely popular in artificial intelligence for modelling sequential decision-making scenarios with probabilistic dynamics [7]. Communication systems can be modelled as finite state machines [8], which makes MDPs ideal for modelling them. While previous research has shown promising results using this variation of the MDP, this thesis examines the strengths and weaknesses of this model when implemented into a real, software defined system, in addition to comparing the MDP approach with a slightly different approach.

The second model presented in this work uses Deep-Q Networks (DQN) to allow the radar to learn the optimal action to take when in a congested spectrum. The system is still being modelled as an MDP with reinforcement learning, but now instead of directly calculating the transition and reward matrices and using policy iteration to solve for the optimal behavior, a deep neural network is used to directly estimate the optimal policy. DQNs have demonstrated the ability to master difficult control policies in Atari 2600 computer games using only the raw pixels as input [9]. The goal of utilizing this model is to reduce the computational complexity resulting from using the MDP and reinforcement learning approach as well as reducing the effect of other weaknesses that occur due to the MDP model.

The MDP and DQN models are compared in this work and their strengths and weaknesses are discussed as well as the potential impact of implementing these two approaches

into a real software defined system.

1.2 Thesis Overview

Chapter 2 discusses what a cognitive radar is and how it fits into the state of the art. It also includes a description of MDP's (Markov Decision Processes) and reinforcement learning, the model used to predict potential interference in the environment. Previous work done in this field is also discussed as well as the software defined system that is being used to implement the cognitive radar.

Chapter 3 describes how the MDP was implemented on the SDRadar platform and the results obtained from this implementation. It also discusses the strengths and weaknesses of this approach and presents a possible solution to the weaknesses.

Chapter 4 describes Deep-Q Networks (DQN) and the results obtained from the simulated application of DQN's in the cognitive radar in order to solve the radar/communication system coexistence problem.

Chapter 5 discusses and compares the MDP and the DQN and their respective capabilities. This section also discusses the results from the implementation on the SDRadar platform and the potential for implementing the DQN onto the system as well.

This work resulted in a conference paper presented at the 2019 IEEE Radar Conference. This work also resulted in a joint conference paper with Pennsylvania State University presented at 2019 SPIE Spectrum Sharing Conference. There are two additional journal papers currently in progress that will are based on work.

Chapter 2

Background

2.1 Cognitive Radar

2.1.1 Radar Basics

In order to discuss a Cognitive Radar, first a basic understanding of radars is required. Radar stands for Radio Detection and Ranging. It is an electromagnetic system used to detect the location and speed of an object [10]. It works by radiating energy toward the said object and analyzing the reflected signal [10]. The distance of an object is determined by the length of time it took for the reflected signal to return to the radar and the speed of the object is determined by analyzing any frequency shifts that were imparted upon the reflected signal. A visual representation of how the radar works is given in Figure 2.1.

The six main parts of the radar are a transmitter, waveguides, antenna, duplexer, receiver and received signal processing [10]. The transmitter is where the waveform is generated and is amplified as necessary. The waveguides are transmission lines for the waveforms to pass through in order to be transmitted. There are different types of antenna that have different benefits and their purpose is to send the signal out into space to be reflected back by an object and analyzed. The duplexer allows for bi-directional waveform travel in a system. This allows the antenna to both transmit and receive the generate waveform. The

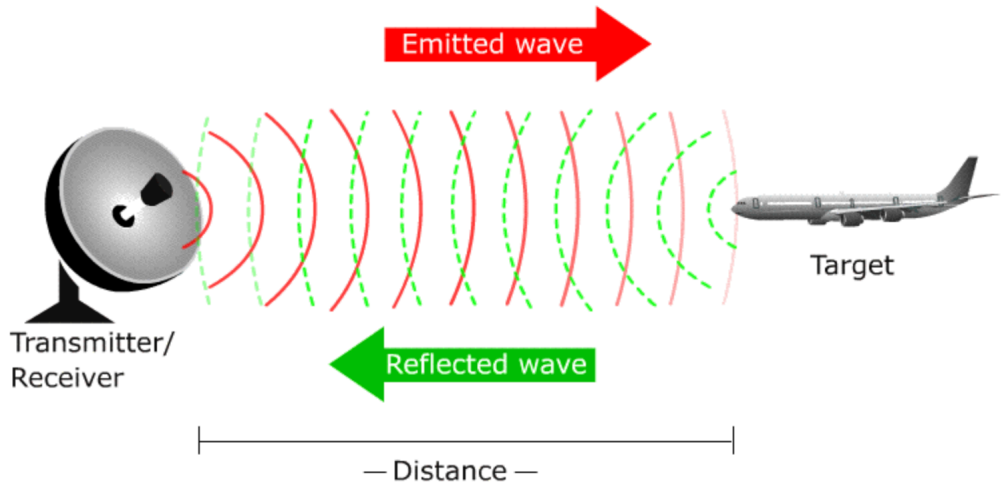

Figure 2.1: A basic radar setup where the waveform is transmitted by the radar, reflected back to the radar by an object, received, and processed to determine the range and velocity of the target.

received signal processing includes filtering, energy detection and comparing to a threshold. The threshold decision is simply choosing a value to compare to the received signal after filtering and energy detection. If the received signal energy is above said threshold then it is determined that the radar is sensing an object. If the signal energy is below the threshold then it is assumed that there is no object present in the observed environment.

There are two primary types of waveforms that are used in radar: pulsed radar waveforms and continuous radar waveforms. The continuous waveform is only used for speed measurement [10]. Without any pulses, there is no way to determine when a certain signal is being reflected, and if we don't have this time information then we cannot determine the distance the target is from the radar using the equation:

$$R = \frac{ct}{2} \tag{2.1}$$

where R is the range of the target, c is the speed of light and t is the time from when the radar waveform was transmitted to when it was received. Using pulsed waveforms we are able to determine both the range and the speed of a target. A pulsed radar sends high power and high frequency pulses toward the target and waits for a reflected waveform before transmitting the next pulse [10]. It can clearly be seen how, using 2.1, the range can be calculated for each pulse. By altering the pulse repetition frequency, different parameters can be measured more accurately. Low pulse repetition frequency is ideal for range measurements since it gives the pulse more time to be sent out and hit a potential target. Having high pulse repetition frequency (PRF) allows for the possibility that the first pulse will not return to the receiver before the next pulse is transmitted resulting in a range ambiguity [11]. For this reason low PRF reduces range ambiguity. High PRF (approaching continuous radar) also reduces Doppler (or speed) ambiguity since this is determined from analyzing the frequency of the returning waveform. If the waveform is coming in more frequently, we have more information and pulses within a time period to analyze.

There are two main radar parameters that are used in this paper to evaluate the performance of the radar: Signal to interference plus noise ratio (SINR) and bandwidth. Both of these parameters provide benefits when maximized. However, there is a trade-off to trying to maximize both simultaneously. Operating in wider bands allows for better range resolution on the target [11], which means we can determine the location of the target with greater accuracy. This is seen in the following equation where range resolution, ΔR, is defined as:

$$\Delta R = \frac{c}{2B} \tag{2.2}$$

where B is the bandwidth of the transmitted waveform. However, by operating in more

bands, thereby increasing bandwidth, the chances of other communication or radar systems interfering with this system are increased.

The SINR is a primary factor in radar performance [11] since this metric needs to be above the chosen threshold for a target to be detected. SINR is generally represented by two different equations, one for signal to noise ratio (SNR) and one for signal to interference ratio (SIR). SNR is defined as the signal power, P_r, divided by the noise power, P_n [11]. We define the signal power, P_r, using the radar range equation:

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{2.3}$$

where the parameters are defined in the following way.

- P_t is the transmit power.
- G_t is the gain of the transmit antenna.
- G_r is the gain of receive antenna.
- λ is the wavelength of the transmitted waveform in meters.
- σ is the mean radar cross section (area of target able to reflect back the waveform) of the target in square meters.
- R is the range from the target to the radar in meters [11].

Additionally, we define the noise power, P_n, to be

$$P_n = kT_0FB \tag{2.4}$$

where the parameters are defined in the following way.

- k is Boltzmann's constant (1.38 $\times 10^{-23}$ watt-sec/K).
- T_0 is the standard temperature (290 K).
- F is the noise figure of the receiver system.
- B is the receiver bandwidth in Hz [11].

Since the SNR is defined as the ratio of signal power to noise power, this results in the following equation for the SNR.

$$SNR = \frac{P_tG_tG_r\lambda^2\sigma}{(4\pi)^3R^4kT_0FB} \tag{2.5}$$

SNR only considers interference from thermal noise. SINR considers interference from other sources as well as the noise. SINR is shown to be

$$SINR = \frac{S}{N + C + J} \tag{2.6}$$

where S is the power of the signal, N is the power from the thermal noise, C is the power

from any clutter present and J is the power from jamming noise [11]. We will ignore power from any present clutter and only consider the noise and jamming (communication system) powers to be the primary contributors to interference. Generally one of these interference types tend to dominate and the equation can be simplified to S/N, S/C or S/J [11]. We begin to see the need for predictive methods to determine where interference will appear so that it can be avoided in an effort to maximize both bandwidth and SINR simultaneously.

2.1.2 Cognition and Artificial Intelligence

As stated previously, a system is considered cognitive if it is capable of four fundamental functions that are basic to human cognition [5]. Those functions being the perception-action cycle, memory, attention and intelligence.

A visual representation of the perception-action cycle taken from [5] is shown in Figure 2.2. The perception-action cycle is the circular flow of information that takes place between an organism and the environment [12]. On the right hand side of the figure we see the perception of the environment that leads to giving feedback to the system. The actuator then takes an action based upon the perceived information. This action affects the environment which is then observed by the perceptor and the cycle repeats.

The system must also have memory in order for it to learn from the environment and be able to predict the consequences of any action taken [5]. The memory is included in the perception of the environment as we are attempting to obtain more information about the environment. The system having attention simply means that it is able to focus on the aspects of the environment that are important to the goals of the system. As for intelligence, it is the multiple levels of feedback in the system that allow for the making of intelligent

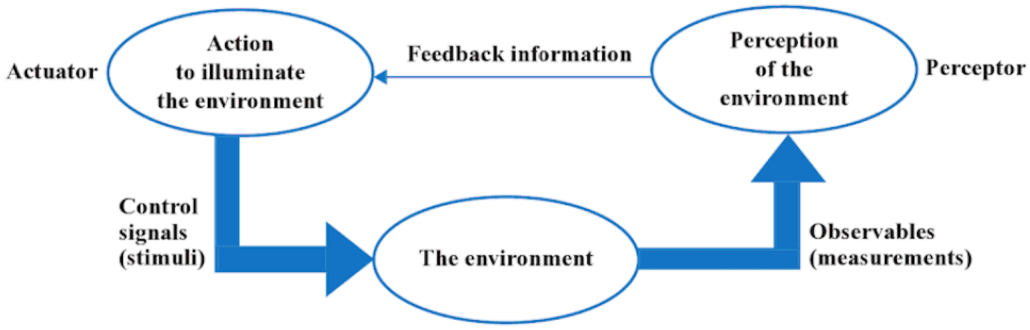

Figure 2.2: Generic form of a perception-action cycle. This perception-action cycle is at the heart of many cognitive radar (and cognitive radio) implementations.

decisions in a potentially unknown environment [5]. This learning takes place after taking an action and observing the effect this has on the environment.

The purpose of artificial intelligence is to build intelligent agents capable of cognitive reasoning [13]. These four functions of cognition are fundamental in creating an intelligent agent. An agent is described as anything that is able to perceive its environment through the use of sensors and act upon that environment through actuators [13]. A rational agent is an agent that does the right thing [13], where what is ”right” is defined by the user. In order to measure how ”right” an action is, there needs to be some performance measure. This performance measure embodies the criterion for success of an agent's behavior [13].

In order to fully describe the system, Table 2.1 first describes the task environment. The first step in designing an agent should always be to fully specify what is known as the task environment [13]. This task environment is what helps realize the system and determine what metrics need to be used to evaluate performance. The cognitive radar will be used as the example for this task environment and will be fully described.

The performance measures include bandwidth, SINR, missed opportunities and collisions. We wish to maximize both bandwidth and SINR as discussed previously and also

Table 2.1: The PEAS (Performance, Environment, Actuators, Sensors) for the cognitive radar project

Agent Type	Cognitive Radar
Performance Measure	Bandwidth, SINR, Missed Opportunities, Collisions
Environment	Any potential landscape a radar could operate in (interference and target info), currently a controlled environment in a lab
Actuators	The radar transmission (the center frequency and bandwidth of the transmitted signal)
Sensors	The radar's receiver for observing radar and spectrum data

minimize missed opportunities and collisions. Missed opportunities are defined as sub bands that the radar is not operating in and where there is no interference present, and as a result the radar should have occupied that sub band. Collisions are defined as events where the radar operates in the same sub band as some interference. The presence of missed opportunities indicate that the bandwidth is not being maximized to the fullest extent and the presence of collisions indicate that the SINR is not being maximized to its fullest extent.

The environment of the system includes the radar environment and the frequency spectrum environment. The radar environment is comprised of the target being observed and the various radar parameters outlined in section 2.1.1 including any observed clutter. The frequency spectrum environment includes any potential interferers within the specific frequency bands of operation along with their respective parameters (time-frequency patterns or statistics).

The actuator in this case is the transmitter of the radar. The radar system should be able to observe the environment and then choose the correct center frequency and bandwidth that will provide the desired effects.

The sensor is the receiver of the radar. This receiver is built to receive both radar data

and spectrum data for separate processing as they have different performance metrics.

It is also significant to note that task environments can be further analyzed and categorized in order to completely specify the task environment. The list of further classifications include fully observable vs. partially observable, deterministic vs. stochastic, episodic vs. sequential, static vs. dynamic, discrete vs. continuous and single agent vs. multiagent. The cognitive radar task environment is further classified in Table 2.2 and explained in the following paragraphs.

Table 2.2: The task environment and its characteristics

Agent Type	Cognitive Radar
Observable	Fully observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Discrete
Agents	Multiagent

This environment is fully observable since we are able to observe every band in every time slot. The state being observed may not always necessarily be the true state of the system due to the noise and the choice of threshold. A band could be occupied by a low power interferer but the sensor may not register this band as being occupied, however we are still able to fully observe the environment (even if the measurements are noisy).

The environment is also stochastic in nature. This is due to the stochastic nature of the interference that the radar is trying to avoid. Additionally, the target's movements that the radar is tracking are also considered stochastic. In both cases if a certain case is observed, the next state is not entirely certain.

This task environment is also considered to be sequential since the current decision could affect all future decisions. Choosing to operate in a band with interference could cause

the SINR to drop to a point where the target is not detected and is no longer able to be tracked. Clearly short term decisions may have long term consequences.

This is also a dynamic environment because the frequency spectrum and target are both continuously changing while the radar is observing them and deliberating on the next best action to take. In cases where the interference is static or nonexistent and there is no target to track, the system may be considered to be semidynamic as the radar's performance still has the potential to change.

In this system, time is handled by turning it into discrete states and there are a finite number of interference states and actions that the radar can take. This environment is therefore considered to be discrete in nature.

This task environment has the potential to be either single agent, multiagent, competitive multiagent or even cooperative multiagent depending on the other agents (or lack thereof) present. If there is relatively simple interference that does not change based upon the radar's actions then it could be considered single agent. If there is another cognitive system in the environment that is trying to thwart the attempts of the cognitive radar to detect targets then it would be considered a competitive multiagent environment. If this other cognitive system is instead trying to mitigate mutual interference with the radar then it would be considered a cooperative multiagent environment. In the case of this cognitive radar environment, it would be accurate to say that it has the potential to behave as all of these options, however we only consider the single agent case.

The first step in designing an agent is to specify the task environment as fully as possible [13]. Once the task environment has been thoroughly explored, it is possible to start forming the design of the agent. Our radar would be considered a simple reflex agent where the agent selects its action based upon the current percept [13]. The goal of the agent currently being developed is to maximize bandwidth and to maximize SINR, so the agent should choose an action that does this based upon the current state. Since these are

competing goals, the agent requires some sort of learning to function properly. This is where reinforcement learning is introduced.

The agent needs to know that something good has happened when it chooses a beneficial action and that something bad happens when it chooses a detrimental action. This type of feedback is called a reward or reinforcement [13]. The task of reinforcement learning is to use observed rewards to learn an optimal or near optimal policy for the environment [13]. Rewards play a large role in defining policies for Markov Decision Processes which will be further discussed in Section 2.2. For now it is sufficient to state that the user develops a reward structure that gives positive rewards for desired behavior, and negative rewards for undesired behavior. With this method the reflex agent is able to learn a policy that maps state directly to the actions that resulted in the greatest overall average reward.

The implementation of cognition into radar leads to the concept of a cognitive radar (which has already been briefly described in the discussion on task environments and learning agents) to be discussed in Section 2.1.3.

2.1.3 Cognitive Radar

Current research into cognitive radar is split into two main areas: enhancing radar functionality and performance, and spectrum sharing [6]. This paper focuses on the spectrum sharing aspect as applied to target tracking radar.

There are three ingredients basic to the constitution of a cognitive radar: 1) intelligent signal processing, which builds on learning through interactions between the radar and the environment; 2) feedback from the receiver to the transmitter; and 3) preservation of the information content of radar returns [3]. With current software defined systems, it is entirely feasible to create a cognitive radar that is able to learn from the environment and adapt to perform better than a typical radar. Not only is it feasible, but it is also becoming necessary

for cognitive systems to be created in order to optimize the use of the frequency spectrum. A challenge for radar designers both in the private sector and the government is to design radars according to the standards set forth by entities such as the FCC who regulate the EM spectrum in the United States [1]. These restrictions limit the flexibility of the radar and lead to inadequate performance of the radar itself [1]. As the frequency spectrum becomes more congested and more bands are being opened up for uses other than radar, the need to share the spectrum becomes more apparent.

The worst case scenario would be the radar operating in a band while another communication system simultaneously operates in that band. This would degrade the performance of both the radar and the communication system rendering them both potentially useless. Radars need a way to coexist in the EM environment by: 1) operating outside their licensed band while complying with international and domestic regulations (in case their own band becomes occupied); 2) detecting and mitigating the effect of interference from RF sources; and 3) identifying and/or cooperating with RF systems in order to avoid causing interference [1]. There are two main approaches to the decision making that the cognitive radar (which mimics the perception-action cycle of cognition) must use to make informed decisions: 1) the Bayesian approach that builds on prior distributions and knowledge-aided models of the environment obtained from past measurements; or 2) the machine learning approach, which determines the next action based only on the measured data and knowledge of actions previously taken [14]. The approach presented in this paper is the machine learning approach based on the software defined USRP (Universal Software Radio Peripheral) X310.

There are many opinions on the different components needed for a cognitive radar to operate correctly. The components for cognitive radar that will be used in the implementation in this thesis are as follows:

1. Informed decision making via the decision-theoretic approach
2. Passive environmental sensors and radar sensors
3. Learning algorithms to improve performance and adapt to unknown environmental scenarios
4. The knowledge database that contains environmental, clutter, target and other *apriori* information
5. The waveform solution space
6. Receiver-to-transmitter feedback to mitigate clutter/interference and maximize target information [24]

Figure 2.3, taken from [1], shows the general framework being used for the cognitive radar. This figure reveals the actuators (radar transmitters) and the perceptors (radar receivers and environmental sensors) of the cognitive radar system that will be interacting with the environment. This framework also outlines the necessary radar processing and decision making processes. We can see that both the radar sensing data and the spectrum sensing data are required in order to make a decision about which waveform to synthesize. While the goal is to avoid any possible interference and share the spectrum as much as possible, this should not degrade the radar's performance in any way.

The decisions process block is where the radar's performance is evaluated and reinforcement learning is used to find the best possible action and formulate the policy that will assign the optimal action to each observed state. This decision-making process using MDPs and reinforcement learning is further explained in section 2.2.

In addition to this framework, there are certain technological requirements necessary

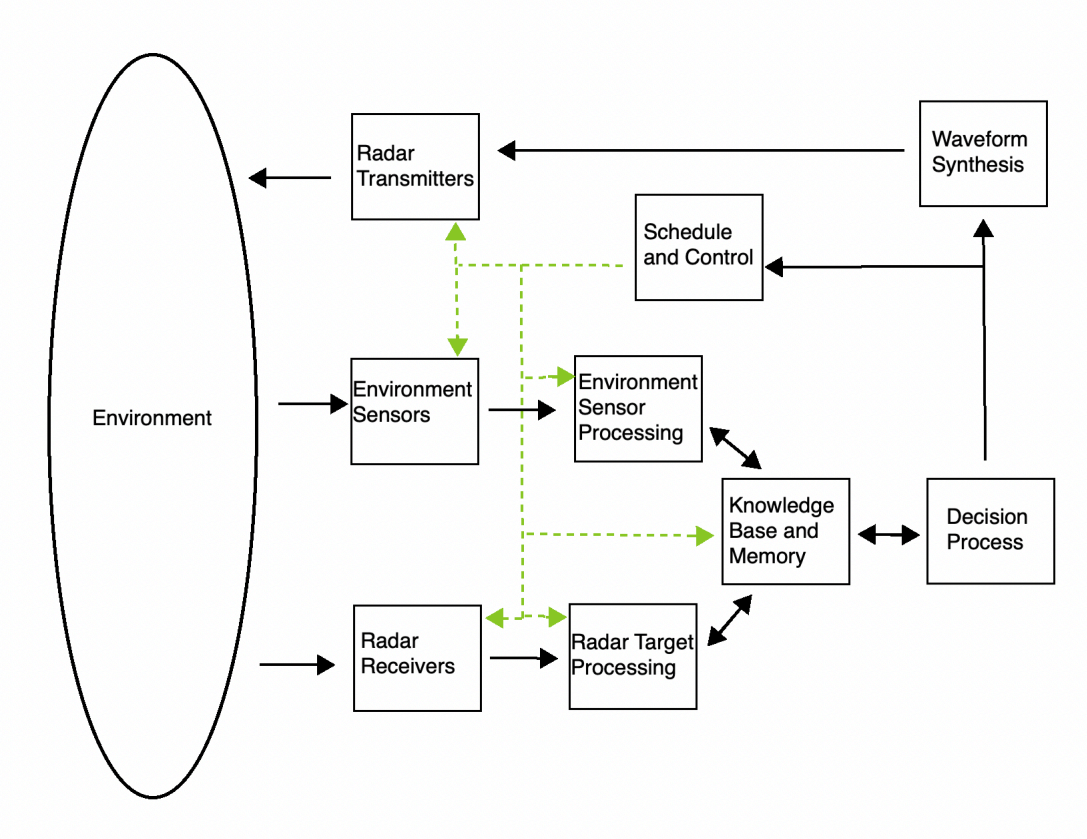

Figure 2.3: Generic framework for a cognitive radar adapted from [1]

for the creation of a cognitive radar. These requirements include:

1. Signal processing algorithms to process target, clutter and environmental information
2. Digital hardware including memory
3. Reconfigurable analog hardware for the radar receiver and transmitter [1]

The USRP X310 software defined radar interfacing with LabVIEW fulfills all of these requirements and appears to be the ideal platform for the cognitive radar to operate. The X310 SDRadar system and how it is controlled and operated is further explained in section 2.4

2.2 Markov Decision Processes and Reinforcement Learning

The need for a cognitive radar that is able to observe and learn from the environment in an effort to mitigate mutual interference with other communication systems has become apparent. In an effort to solve this issue, the radar environment was chosen to be modeled as a Markov Decision Process (MDP) where the reflex agent (the radar) learns the optimal policy through the reinforcement technique of policy iteration. In the following paragraphs the MDP model will be explained along with how reinforcement learning is used to find the optimal policy.

As we will be modelling the radar environment as a Markov Decision Process, it is appropriate to first describe the environment we will be examining in addition to the MDPs. The radar environment consists of a single target, which we assume is a point target, that the radar is tracking. This target has some straight, constant velocity trajectory that is random for each individual training run. A communication system which is capable of operating in one or more frequency bands is assumed to be operating in the environment, providing the radar with some obstacle to avoid. The radar itself is considered stationary, and operates using a linear frequency modulated (LFM) waveform. Any losses due to atmospheric effects or clutter are assumed to be negligible. An example of the radar trajectory is shown in Figure 2.4. The orange circles represent possible target position states, and the arrow represents the trajectory of the target.

A Markov Decision Process consists of states, actions, transitions between states and a reward function [15]. A state is a unique characterization of all that is important in the problem being modeled [15]. For this case, the state includes the target position, X, and target velocity, V, the SINR of the target, and the interference pattern of the communication

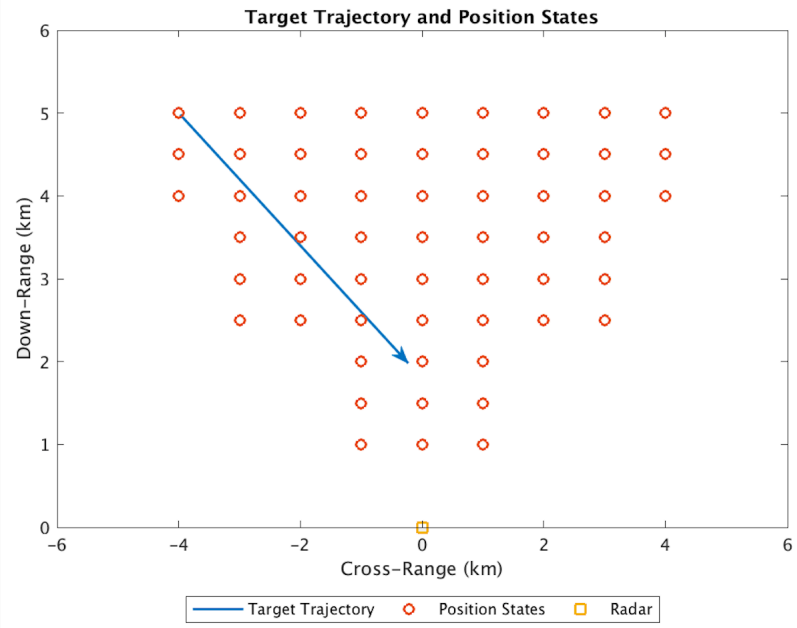

Figure 2.4: A Down-Range example of the target trajectory

system, θ,

$$X = [x_1, x_2, ..., x_p]^T \tag{2.7}$$

$$V = [v_1, v_2, ..., v_v]^T \tag{2.8}$$

$$\theta = [\theta_1, \theta_2, ..., \theta_m]^T \tag{2.9}$$

where p is the number of position states, v is the number of velocity states, m is the number of unique interference states, and T denotes the transpose operation. The interference pattern, θ, is represented as a series of ones and zeros with the length representing the number of possible bands. For example, the interference pattern [0 1 0 1 0] represents 5 possible

bands with detected interference in bands two and four, while the other bands are free of interference. Given an arbitrary number of bands N, the total number of unique interference states is then $M = 2^N$. The total number of states is therefore $N_s = p * v * 2^N$ [16]. Actions can be used to control or change the system state [15]. In this instance, actions consist of choosing the bands in which the radar will be operating. The number of actions has been shown to be $N_a = N * (N + 1)/2$ [16] because we limit actions to transmitting in contiguous bands. This limitation exists because currently there is no radar processing for operating in non-contiguous bands that would result in improved radar performance. Once radar processing is developed that allows for non-contiguous band operation with increased radar performance then the number of actions will increase, also increasing computational complexity.

The transition function is defined as the probability of ending up in state s' after doing action a in state s, denoted as $T(s,a,s')$ [15]. The reward function specifies rewards for being in a state, or taking some action in a state, and is defined by the user [15]. For instance, we define a positive reward for positive SINR and larger bandwidths and a negative reward for negative SINR. An example of the reward structure is shown in Figure 2.5. The transition function and the reward function together define the model of the MDP [15]. By observing the cumulative reward for every transition that occurs in the environment, a policy iteration algorithm can then be used to find the optimal action to take for each state.

An example of a Markov Decision Process is shown in Figure 2.6 taken from [16]. The states are represented by s_1, s_2 and s_3 while the transition probabilities are the values adjacent to their respective arrows. The reward for arriving in a state is also shown beneath the state's label. This figure shows positive rewards for arriving in states s_1 and s_3 with a negative reward for arriving in state s_2. These rewards are user-defined and reflect the goals that the user wishes the system to pursue (in the case of Figure 2.6, the goal is to remain in

SINR	Reward
< 0	-45
0 – 2	+1
2 – 5	+2
5 – 8	+3
8 – 11	+4
11 – 14	+5
14 – 17	+6
17 – 20	+8
> 20	+10

Number of Bands	Reward
1	+0
2	+10
3	+20
4	+30
5	+40

Figure 2.5: An example of the reward structure for an MDP in our cognitive radar problem

state s_1, while arriving in state s_3 is acceptable, and state s_2 should be avoided).

In our system, we must learn T and R, more specifically a transition probability matrix and a reward matrix that are built up over a finite number of training runs. Once all training runs have been completed, these two matrices are used with the Bellman equations to calculate the optimal policy. This policy is a look-up table where the input is the current state, and the output is the optimal action [16]. The optimal policy is defined as π^*

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, E\left\{\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right\} \tag{2.10}$$

where $E()$ is the expectation value of the parameters inside the parenthesis, γ is the discount factor and determines how much importance is placed on future rewards as opposed to immediate rewards, and $R(s_t)$ is the reward of the current state. This policy is found using the Bellman equations by using the method of policy iteration. Before discussing policy

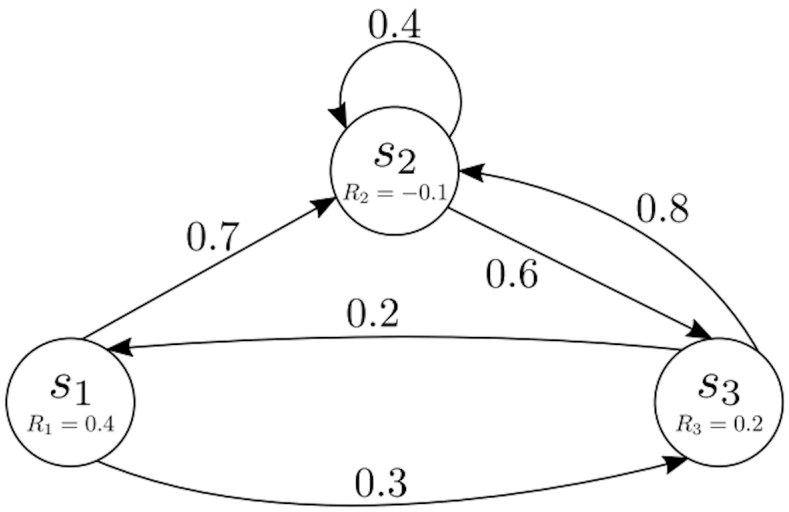

Figure 2.6: An example of a Markov chain

iteration, it is necessary to describe the utility of a state. We can define the utility of a state $U^{\pi}(s)$ with respect to a specific policy π. If we let s_t be the state the agent is in after executing π for t steps (the greater t is the more accurate our evaluation of the utility will be), then we have [13]:

$$U^{\pi}(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s\right\} \tag{2.11}$$

where the utility of a state is simply the expected sum of discounted rewards if the agent executes an optimal policy. Comparing 2.10 to 2.11 we can see that the optimal policy is simply the argument that maximizes the utility of the state, or rather, maximizes the discounted future reward of the agent's action. Now using this knowledge, the policy iteration algorithm can be explained.

The policy iteration algorithm alternates between the following two steps, beginning from some initial policy π_0:

1. Policy evaluation: given a policy π_i, calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed

2. Policy improvement: Calculate a new policy π_{i+1}, using one-step look-ahead based on U_i [13]

The policy evaluation step is essentially just evaluating the total reward for the action currently deemed optimal. The policy improvement step is to try a new action and determine if this action is an improvement upon the current action that the system believes is optimal. These steps usually alternate until a sufficient number of iterations have gone by where the policy improvement step has not found a better action than the one currently deemed optimal. At this point we know that the utility function U_i is a fixed point of the Bellman update and is therefore a solution to the Bellman equations [13]. This solution must therefore be the optimal policy. The specific algorithm can be found in Appendix A.

Due to the nature and sizes of the transition and reward matrices, it is not always possible for a solution (optimal policy) to be calculated. Instead of building up these matrices and using the Bellman equation, we consider using a neural network to reduce computational cost and improve performance by adding memory. The implementation of the Deep-Q Network (DQN) is further discussed in Section 2.5.

2.3 Previous Work

There has been much work done to determine the feasibility of implementing an MDP with reinforcement learning through simulation. Most of the work has been done in MATLAB

and is presented in [6]. The results of this work will be briefly summarized in this section in order to provide background as to what is actually being implemented in a real system.

As described in previous sections, the environment is being modeled as an MDP and the optimal policy that maps the correct action to each observed state is being calculated based upon the user-defined reward structure. The simulation code is split up into two distinct phases: 1) Exploration phase; and 2) Exploitation phase. During the exploration phase, the radar takes random actions (transmitting in a different random set of contiguous bands every time step), observes the states and their transitions and calculates the reward and transition matrices from this information. At the end of the exploration phase the optimal policy is calculated from the reward and transition matrices using the Bellman equations. During the exploitation phase, the radar uses the calculated policy to select an action based upon the state it has observed. This phase could also be called the evaluation phase where the radar's behavior is examined to determine how successful it was in maximizing both SINR and bandwidth.

There were three main interference types that this method of machine learning was tested against: 1) Constant Interferer; 2) Intermittent Interferer; and 3) Triangle Sweep Interferer. There will be two plots and two tables in which the results are presented for the constant interferer. The other interference types results will be presented with only the two plots. The first plot, found in Figure 2.7, shows results for the rewards (blue curve), bandwidth (orange curve), SINR (yellow curve), and target distance (purple curve) as they evolve through time. The second plot shows the state observed (orange curve) and the action taken (blue curve) by the radar represented in binary form as they evolve over time. For example, an interference pattern of $\theta_1 = [1\ 0\ 0\ 0\ 0]$ would be represented by the number 16 and the corresponding action $a_1 = [0\ 1\ 1\ 1\ 1]$ would be represented by the number 15. The first table shows the calculated rewards for every possible action taken when the target is far away from the radar and the second table shows the same except for when the target is

close to the radar.

Figure 2.7, taken from [6], shows the results from testing the MDP on the constant interferer. In this example the interference occupies the first band such that the interference pattern is θ =[1 0 0 0 0] and remains that way for every time step. As indicated by the steadily increasing rewards, the MDP was able to successfully learn the behavior of the interferer and avoid it when necessary. We can see from the second plot that the radar avoids the interference, but at one point it does transition into operating in all bands. This is due to the fact that the target moves closer to the radar, then moves further away. When the target reaches a certain distance from the radar, the radar recognizes that it can operate in the presence of interference and still have an acceptable SINR. This is shown by comparing Table 2.3 with Table 2.4. Table 2.3 shows the rewards when the target is far away from the radar, and Table 2.4 shows the rewards when the target is closer to the radar. It can clearly be seen that for the closer target, the best action is to operate in all bands, and for the further target the best action is to avoid the interference. The radar was able to learn when and how it should avoid interference according to the location of the target and the interference type.

Figure 2.8, taken from [6], shows the results from testing the MDP on the intermittent interferer at 10 percent chance of the interferer occupying band. This means that the interference pattern is θ =[1 0 0 0 0] 10 percent of the time, and θ =[0 0 0 0 0] the remaining 90 percent of the time. We can see that the radar operates in all bands all the time because it has learned that the interferer is not operating often, and when it is operating, it will vacate that band almost immediately. Since the radar cannot predict when the signal will be present, the optimal action is to use all bands all the time.

Figure 2.9, also taken from [6], shows the results from testing the MDP on the triangle sweep interferer. This means that interference follows a specific pattern. The pattern repeats

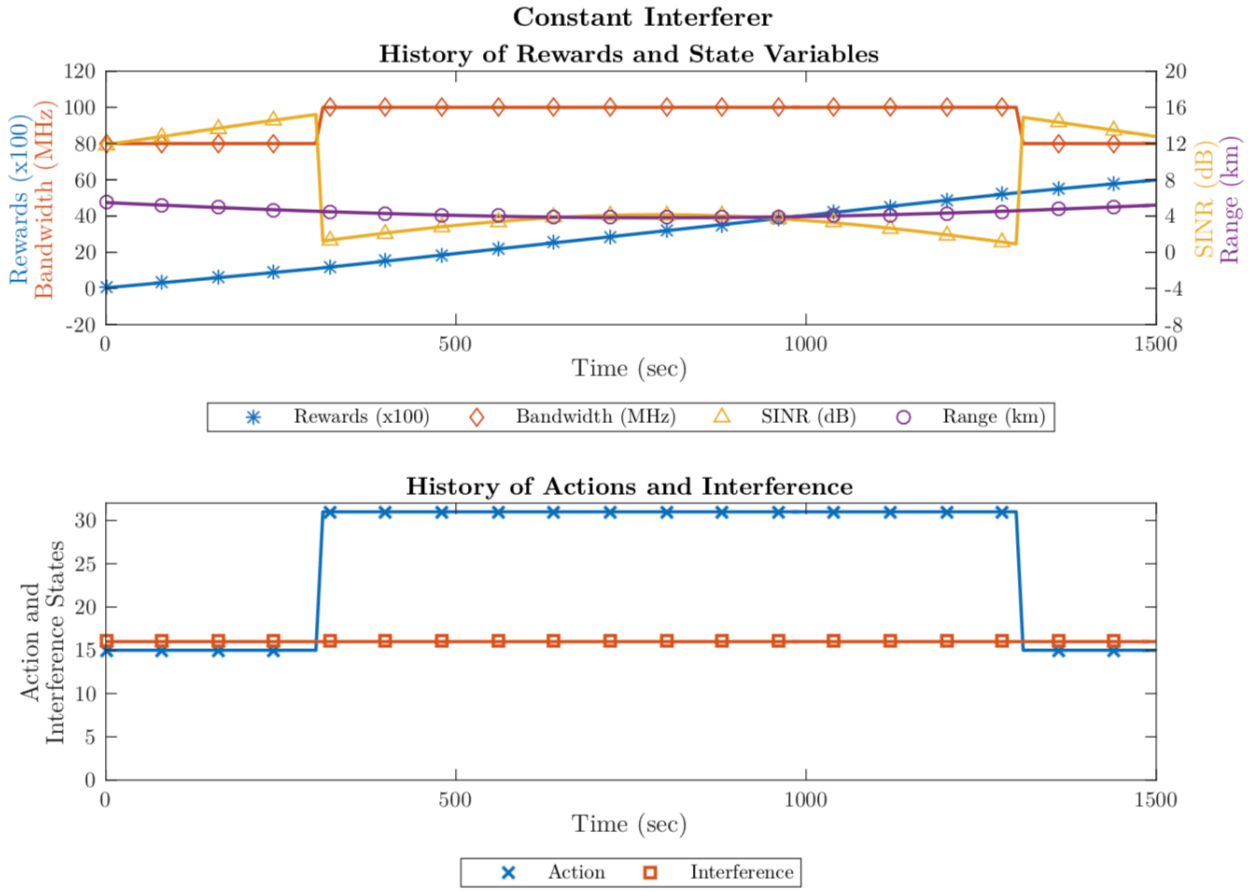

Figure 2.7: Results from testing MDP on Constant Interference taken from [6]. In the top plot, the orange curve represents bandwidth (MHz), the yellow curve represents SINR (dB), the blue curve represents cumuative rewards given to the radar, and range of the target (km). The bottom plot shows the action taken by the radar in blue and the interference pattern in orange.

as follows:

- θ =[1 0 0 0 0]
- θ =[0 1 0 0 0]
- θ =[0 0 1 0 0]
- θ =[0 0 0 1 0]

Table 2.3: Rewards for each action for the constant interferer when the target is far away from the radar

Observed Interference	Range (km)	Action	BW (MHz)	SINR (dB)	Reward
[1 0 0 0 0]	5.5	[0 0 0 0 1]	20	11.8	5
[1 0 0 0 0]	5.5	[0 0 0 1 0]	20	11.8	5
[1 0 0 0 0]	5.5	[0 0 1 0 0]	20	11.8	5
[1 0 0 0 0]	5.5	[0 1 0 0 0]	20	11.8	5
[1 0 0 0 0]	5.5	[1 0 0 0 0]	20	-9.1	-45
[1 0 0 0 0]	5.5	[0 0 0 1 1]	40	11.8	15
[1 0 0 0 0]	5.5	[0 0 1 1 0]	40	11.8	15
[1 0 0 0 0]	5.5	[0 1 1 0 0]	40	11.8	15
[1 0 0 0 0]	5.5	[1 1 0 0 0]	40	-6.1	-35
[1 0 0 0 0]	5.5	[0 0 1 1 1]	60	11.8	25
[1 0 0 0 0]	5.5	[0 1 1 1 0]	60	11.8	25
[1 0 0 0 0]	5.5	[1 1 1 0 0]	60	-4.4	-25
[1 0 0 0 0]	5.5	[0 1 1 1 1]	80	11.8	35
[1 0 0 0 0]	5.5	[1 1 1 1 0]	80	-3.2	-15
[1 0 0 0 0]	5.5	[1 1 1 1 1]	100	-2.3	5

- θ =[0 0 0 0 1]
- θ =[0 0 0 1 0]
- θ =[0 0 1 0 0]
- θ =[0 1 0 0 0]

With only one memory state, the MDP was unable to determine which direction the interferer was sweeping and did not perform very well. There were multiple collisions and missed opportunities. Figure 2.9 shows the results of the MDP with two memory states. By knowing where the interference currently is, and where it was one time step before that, the radar was able to accurately predict and avoid the interference as we can see by the SINR curve that never drops below 0 dB.

Table 2.4: Rewards for each action for the constant interferer when the target is far away from the radar

Observed Interference	Range (km)	Action	BW (MHz)	SINR (dB)	Reward
[1 0 0 0 0]	3.8	[0 0 0 0 1]	20	18.2	8
[1 0 0 0 0]	3.8	[0 0 0 1 0]	20	18.2	8
[1 0 0 0 0]	3.8	[0 0 1 0 0]	20	18.2	8
[1 0 0 0 0]	3.8	[0 1 0 0 0]	20	18.2	8
[1 0 0 0 0]	3.8	[1 0 0 0 0]	20	-2.7	-45
[1 0 0 0 0]	3.8	[0 0 0 1 1]	40	18.2	18
[1 0 0 0 0]	3.8	[0 0 1 1 0]	40	18.2	18
[1 0 0 0 0]	3.8	[0 1 1 0 0]	40	18.2	18
[1 0 0 0 0]	3.8	[1 1 0 0 0]	40	0.3	-35
[1 0 0 0 0]	3.8	[0 0 1 1 1]	60	18.2	28
[1 0 0 0 0]	3.8	[0 1 1 1 0]	60	18.2	28
[1 0 0 0 0]	3.8	[1 1 1 0 0]	60	2.0	22
[1 0 0 0 0]	3.8	[0 1 1 1 1]	80	18.2	38
[1 0 0 0 0]	3.8	[1 1 1 1 0]	80	3.2	32
[1 0 0 0 0]	3.8	[1 1 1 1 1]	100	4.2	42

From these three examples, we have demonstrated that the MDP is capable of learning even some of the more complex interference types (triangle sweep) with increased memory. Since the MDP was found to be a viable method of predicting RFI in the EM spectrum, the next step is to implement this into a real system. The software defined system that the MDP is being implemented on is described in Section 2.4.

2.4 SDR System

The system that the cognitive radar is being developed on is the Ettus X310 software-defined radio. This hardware was able to be converted into a National Instruments (NI) software-defined radar that is able to transmit and receive radar waveforms. The hardware includes

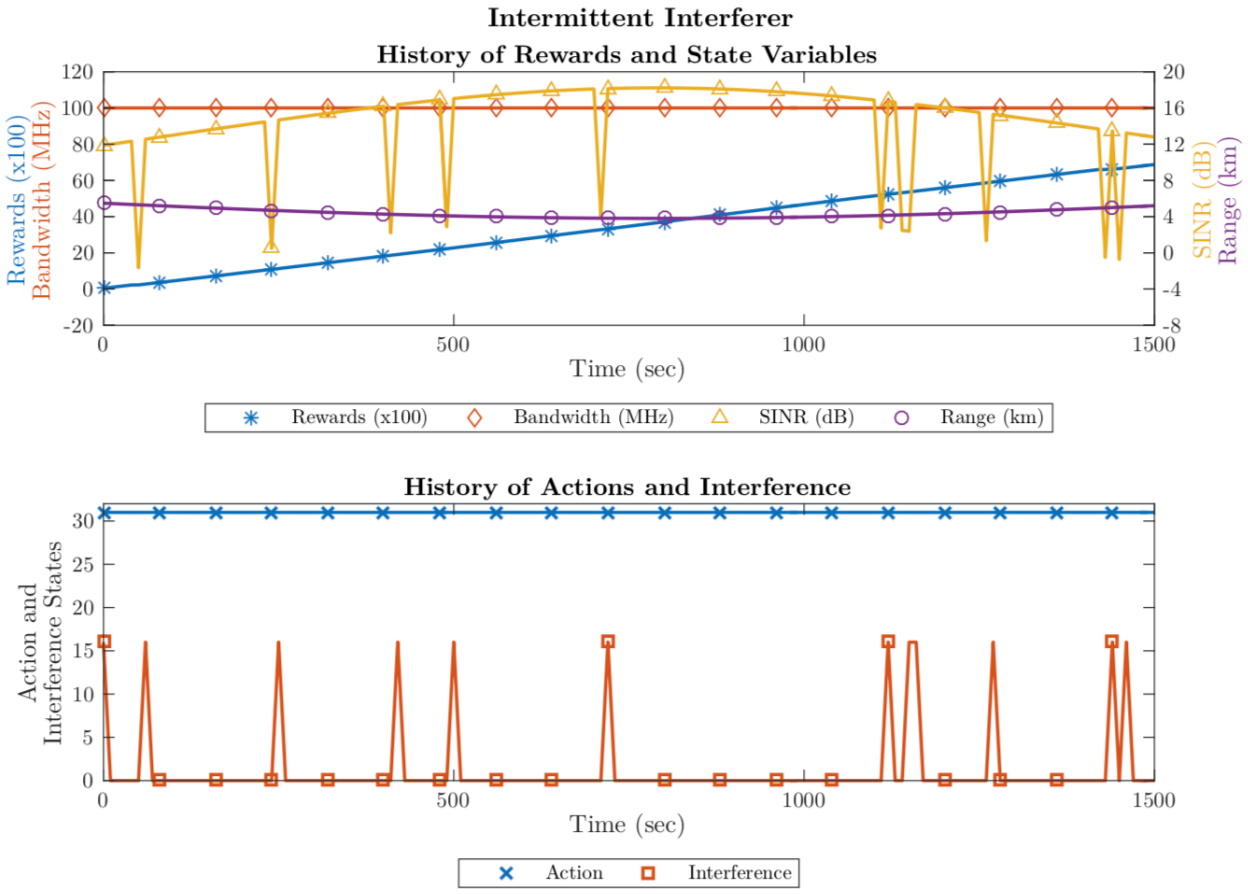

Figure 2.8: Results from testing MDP on Intermittent Interference taken from (cite e3)

two UBX 160 daughterboards that allow for two receive-only ports and two transmit/receive ports. The SDRadar is connected to the host computer with a PCIe 4x card and cable allowing for a high-speed interface of 200 MS/s. The user-programmable XC7K410T FPGA provides high-speed connectivity between all major components within the device including radio frontends, host interfaces and DDR3 memory [17].

The software being used to control the SDRadar is various versions (2017 and 2018) of LabVIEW. For the MDP implementation an interface with MATLAB is also utilized. The LabVIEW code previously created was purely a data collection code that transmitted radar

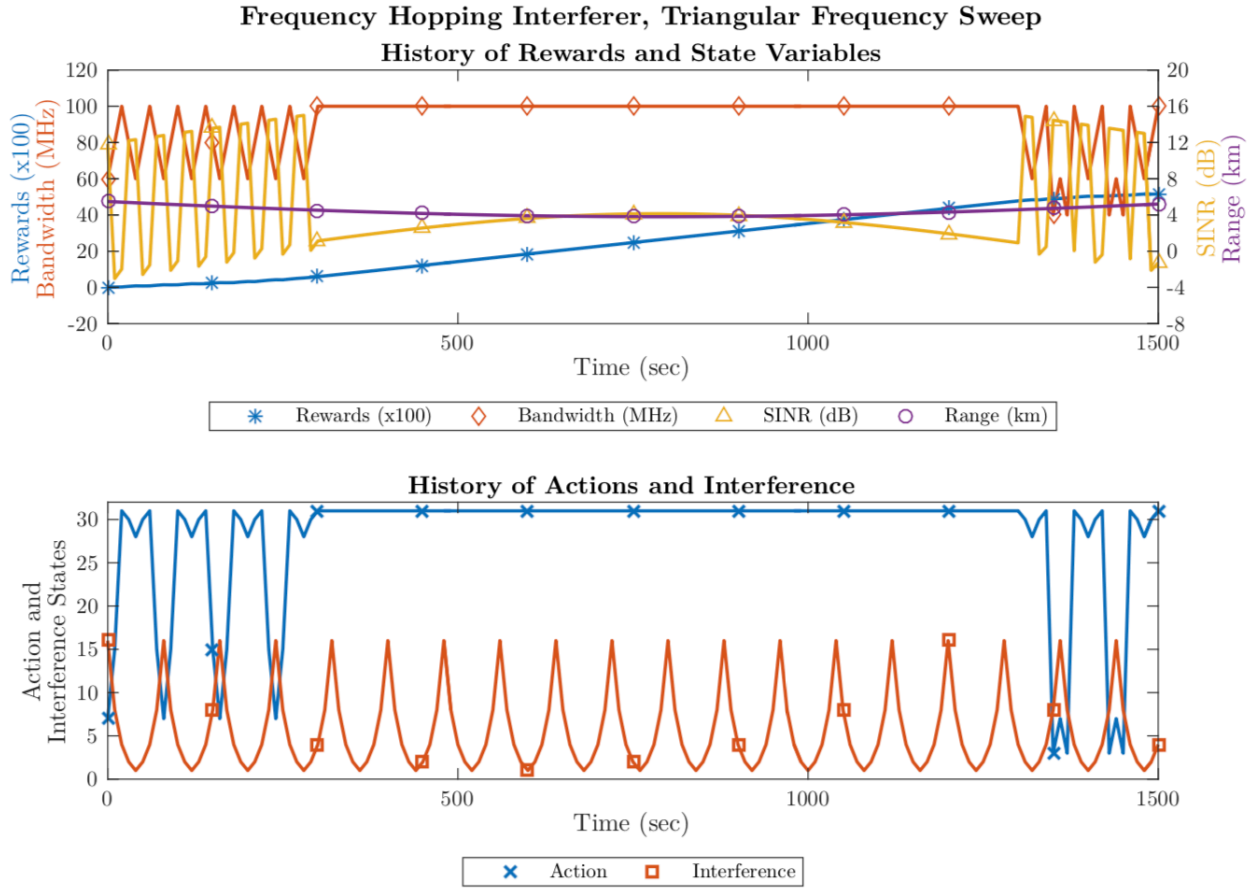

Figure 2.9: Results from testing MDP on Triangle Sweep Interference taken from (cite e3)

waveforms and received both radar waveforms (as the radar data) and frequency spectrum (as the spectrum sensing data).

From Figure 2.11 we can clearly see the allocation of processing that occurs in the FPGA on the SDRadar versus the processing that occurs on the host computer as well as how the data is displayed. This allocation was chosen for its efficiency in order to allow for additional features to be added in the future. My contribution to this system was implementing the cognition in LabVIEW so that the SDRadar is able to sense the spectrum and accurately predict and avoid any potential RFI.

Figure 2.10: The USRP X310 platform that we are using

While Figure 2.11 shows a detailed allocation of the processing, Figure 2.12 shows a broader allocation of resources including the vector signal transceiver (VST). This is a high level view of what is occurring when the radar is being run.

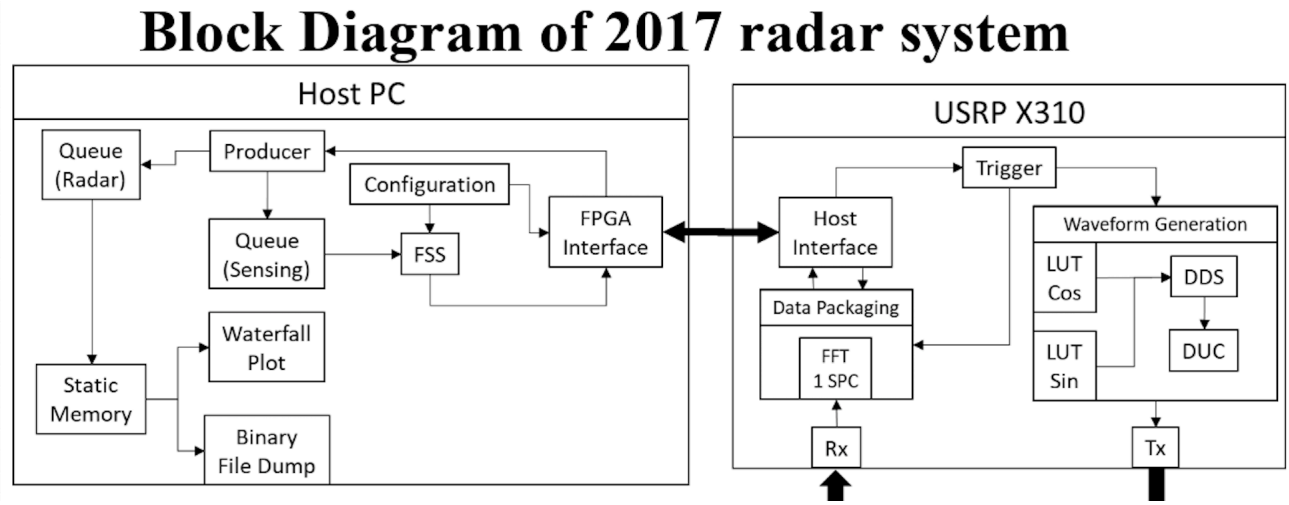

Figure 2.11: The Block Diagram indicating the allocation of resources and processing on the SDR system prior to my involvement

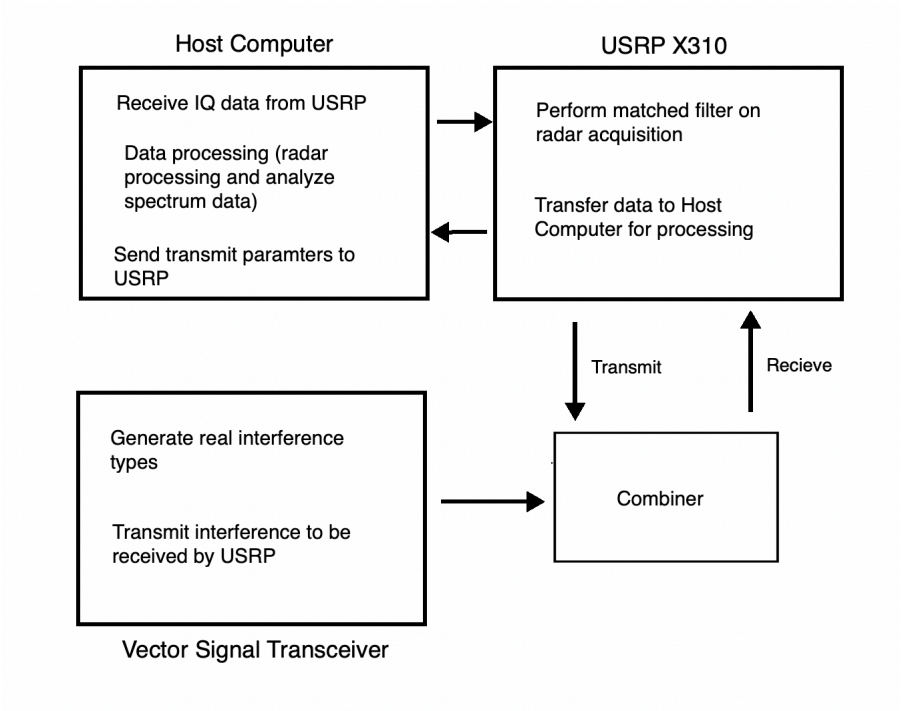

Figure 2.12: The block diagram showing the highest level of what each portion of the system does. The VST generates and transmits the interference, this is combined with the transmitted waveform from the radar, the USRP receives this waveform and performs the matched filter. This data is then sent to the host computer for the radar processing and to analyze the spectrum data.

2.5 The Deep-Q Network

The Deep-Q Network is an extension of the MDP where we replace the estimation of transition and reward matrices, the Bellman equation and policy iteration with a neural network. Neural networks have been successfully applied to radar tracking [18] [19] [20]. This previous work only addresses radar tracking however, and not the radar-communication coexistence. Additionally, neural networks have been shown to be successful in obstacle avoidance in [21], which is analogous to the situation in the EM spectrum where the radar must try and avoid potential Radio Frequency Interference (RFI). Deep-Q Networks (DQN) are an extension of the MDP approach in [16], where here we use neural networks in an attempt to estimate Q-values to find the optimal action instead of using the Bellman equation to find an optimal policy as done in [16]. The two primary uses of neural networks are for classification or function estimation [4]. The latter is what we will be using in order to estimate Q values.

This neural network is trained with a variant of Q-learning using stochastic gradient descent to update the weights [9]. Stochastic gradient descent has been shown to perform exceptionally for large scale problems compared to other optimization algorithms [23]. Stochastic gradient descent (SGD) relies on estimating the gradient and updating the weights, w_t, based upon a single randomly chosen sample z_t [23]. The SGD algorithm is expressed as

$$w_{t+1} = w_t - \gamma_t \nabla_w \ell(z_t, w_t) \tag{2.12}$$

where γ_t is an adequately chosen gain similar to what γ represented in the MDP, sometimes called the learning rate in machine learning, ℓ represents the loss function we wish to minimize, and ∇_w is the gradient with respect to w [23]. Figure 2.13 shows an example of a fully-connected neural network. During training, the input layer is comprised of the previ-

ous state, the action taken, the current state, and the reward that was given. These inputs are passed through fully-connected hidden layers in the neural network which performs the SGD and outputs an estimate of the Q function for each possible action. At each iteration, the weights that are placed upon each of the transitions to the hidden layers are adjusted in order to minimize the loss function and thereby finding the action that maximizes the reward. The loss function is a function comparing the current Q value to previous Q values in an attempt to improve the policy being estimated. Since the optimal action is not known, the reward given to a particular action is used to train the neural network and update the weights to allow the DQN to select what is considered the optimal action (the action with the highest reward for a given state). The neural network being used is built in Python using the TensorFlow library [24] [25].

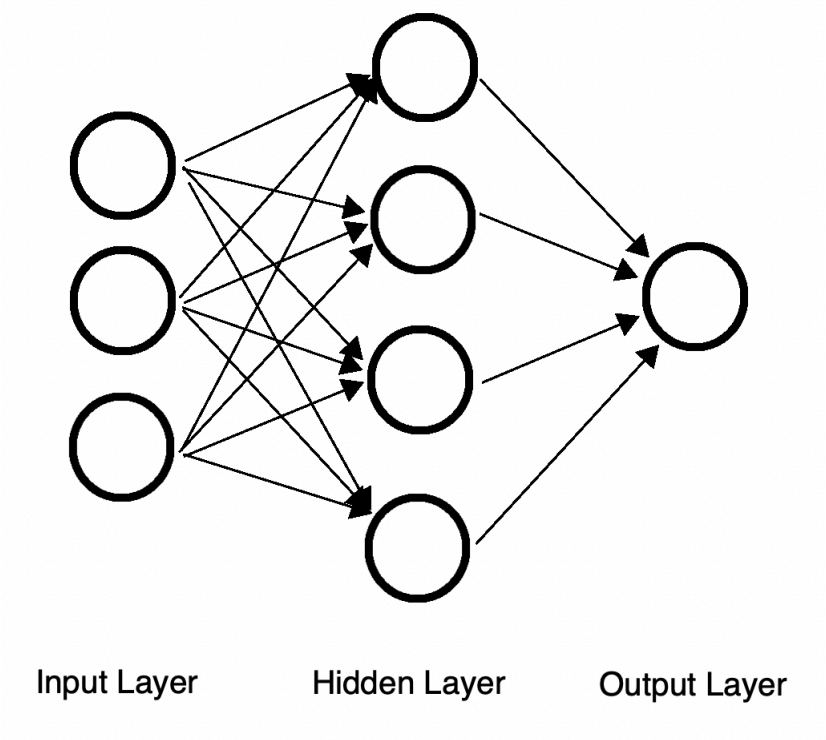

Figure 2.13: An example of a fully connected neural network with an input, hidden and output layer.

Once the neural network is trained, the neural network accepts the current state as an input and estimates the Q values of each potential action. The highest Q value is then chosen and the corresponding action is taken by the radar. Figure 2.14 compares the network architecture while training (left) and during implementation (right). During training the states, rewards and actions are recorded and used to update the weights of the neural network, while during implementation only the state is passed through the neural network and the weights are used to find the optimal action (the weights are still being updated in this scenario as well allowing for adaptation to changing interference patterns).

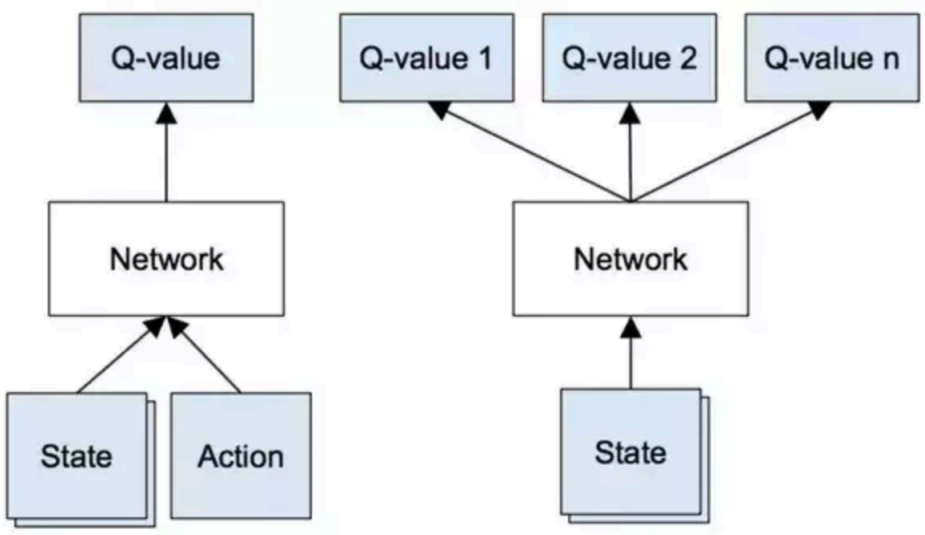

Figure 2.14: Training (left) versus implementing of the neural network (right)

This neural network is advantageous because large matrices are no longer necessary to build up the inputs to the Bellman equation. The only computationally limiting factor is now the size of the state itself or the size of the neural network. There should no longer be a computational complexity issue as long as the state itself remains reasonable.

Chapter 3

Cognitive Radar Implementation

3.1 Contributions to the Project

My contributions to this project included adding the MDP predictive method for avoiding any potential RFI and adding radar processing and display to the existing software defined radar. These contributions can be seen in Figure 3.1 as being highlighted by the blue, yellow and green blocks in the block diagram. While the radar processing was implemented into this system, the main contribution to the system that is being focused on is the implementation of the predictive method.

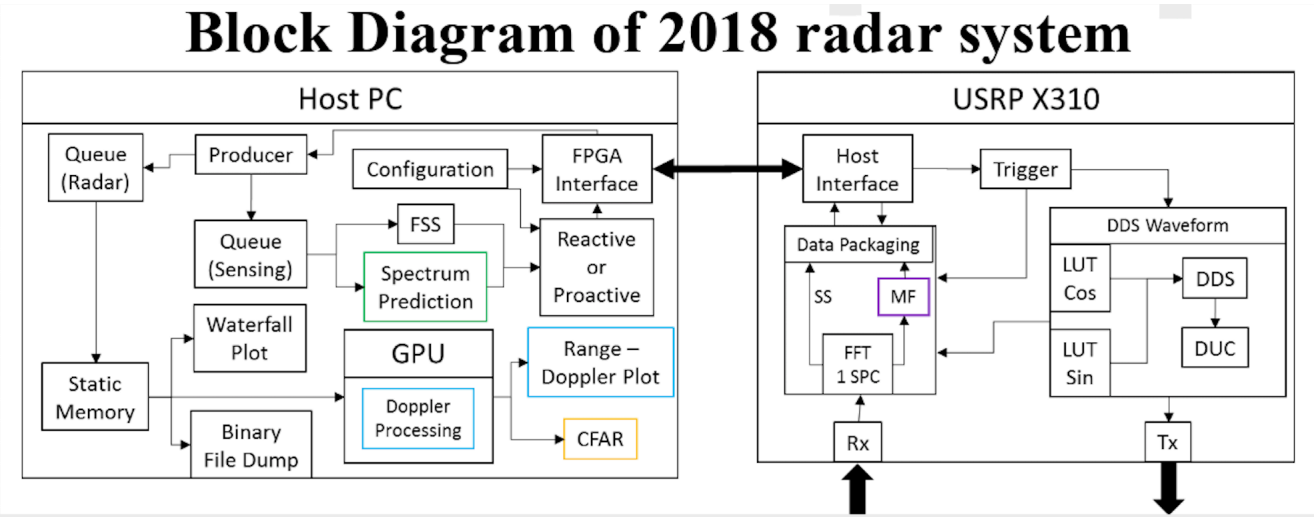

Figure 3.1: The block diagram after I implemented radar processing and the MDP

The previous version of the code (data collection) has two modes: 1) Operating in full bandwidth despite the existence of RFI; and 2) sensing the frequency spectrum and reacting to avoid the interference, which can lead to many missed opportunities and collisions in fast changing environments thereby decreasing the performance of the radar. This reactive code simply observed the recorded spectrum, found the largest band possible and transmitted in that band. The predictive method separates the 90MHz spectrum into five, equally spaced, 18MHz sub bands similar to how it was done in the MDP simulation. Implementing this method also required the transmissions to be separated into a learning phase and an acting phase. During the learning phase, random actions are taken (or simulated), recorded, and sent to the MATLAB portion of the code. Similarly, the interference states are also recorded and sent to the MATLAB interface. Since the MDP simulation was built up in MATLAB, only minimal changes to the current simulation code were made in order to fit it into the overall scheme of the LabVIEW code. The states and actions are now inputs, and the number of training runs is no longer specified by the user. The policy is based on all the actions and states that occurred during the learning phase. The output of the MATLAB code is the calculated policy as well as the probability and reward matrices so that they can be updated as the radar continues to run. The policy is calculated and recalculated every 400 data collects (after 400 states and actions have been saved). This is done so that the code doesn't lag and fall behind on data if it tries to calculate the policy too quickly, and so that it doesn't take an excessive amount of time to calculate at the end thereby missing opportunities to act while it is busy processing.

In Appendix A.2 we see the graphical user interface (GUI) of the system. The total time the program is running is split of into a learning phase and acting phase where the radar learns the optimal policy, then uses this policy. All the other buttons and specified values have purposes unrelated to the machine learning aspect of the code and will not be discussed here. The spectrogram tab allows the user to see the RFI in addition to the radar

transmits. This is the primary way for the user to affirm the proper functionality of the MDP method. Additionally, the MDP tab allows the user to see the policy being calculated every 400 iterations and count up the collisions and missed opportunities that resulted from using the policy to make informed decisions about the best action to take in the next time step. The figure found in Appendix A.2 goes in depth into the code itself and shows the machine learning loop that was added to the code in order to include the predictive method for avoiding RFI. In this figure, there is logic associated with waiting in order to collect 400 states and actions. Then state and action data is sent to the MATLAB block where, during the learning phase, LabVIEW uses a MATLAB command line to perform the Bellman equations and determine an optimal policy. During the acting phase this block changes to counting the missed opportunities and collisions based upon the policy that has been employed. Finally the policy is saved in the computer's memory so that the computer can send the optimal action (based upon the current state) to the FPGA so that the proper waveform can be transmitted during the next time step.

In order to full realize the improvements to the system, a significant amount of radar processing was added. The radar processing that was implemented includes matched filtering the received signal, Doppler processing to determine speed information and various target detection algorithms such as CFAR (Constant False Alarm Rate) detection.

The matched filter and Doppler processing work resulted in a Range-Doppler plot that allows the user to either visually identify a target or to then run a target detection algorithm to determine if a target is actually present or if the signal they are observing is just noise or interference. The matched filter was readily implemented since the information about the transmitted waveform is readily available. This transmitted waveform was time-reversed, Fourier Transformed and multiplied in the frequency domain with the received signal. This provides us with data that shows when the received signal arrived back at the receiver. Since we know the speed of light, this time domain data can readily be changed from time infor-

mation to distance information and a range plot is produced. The Doppler processing done is simply saving a specific number (1000 in this case) of received radar waveform pulses and taking a Fast Fourier Transform (FFT) across those 1000 pulses to determine the frequency shift. This frequency shift corresponds to a Doppler shift which indicates that a target is moving. This provides the velocity information of the target.

A major contribution to this project was implementing both the Doppler processing and the FFT of the matched filter (to get the information back into the time domain) on the GPU of the host. The GPU is well equipped to handle parallel processes, such as taking many simultaneous FFT's. This implementation doubled the speed at which these calculations were performed and the output was a Range-Doppler plot.

In addition to these algorithms, the Constant False Alarm Rate (CFAR) radar detection algorithm was implemented. The CFAR method evaluates a point on the Range-Doppler map by taking the user selected false alarm rate, and selecting a threshold for that point based upon the surrounding data points. If the power value associated with the data point is above this determined threshold, then this indicates a target detection. The probability of detecting a target when there is no target present is termed the Probability of False Alarm and the probability of detecting a target that is present is termed Probability of Detection. These two metrics along with SINR form a tuple that provide information on the probability of detecting a target in the environment.

3.2 Implementation Results

As discussed previously, the MDP tab and spectrogram can be used to determine the performance of the system based upon four metrics: missed opportunities, collisions, SINR and

bandwidth. The results obtained from the LabVIEW implementation are consistent with the results obtained in the MATLAB simulation, in that in most cases the MDP is able to accurately predict the location of the RFI in the next time step and avoid it accordingly (when it is optimal to do so).

The first results that are being presented simply show the MDP's functionality. Figure 3.2 shows the learning phase of the radar. This figure shows frequency on the x-axis, time on the y-axis on the scale of milliseconds and power along the z-axis (i.e. the color). All of the uniform horizontal lines represent the chirp waveform the radar is receiving and the other high power signals represent interference in the environment. During the learning phase, the radar is taking random actions in order find the optimal action for each state that is being observed. The length of time that is spent learning varies based upon the complexity of the interference being observed. Figure 3.2 shows the radar attempting to learn the optimal policy in the 690 MHz band. This band contains low complexity interference (mainly stationary) and the radar only needs about 1-2 seconds to complete the learning process.

Once the learning process is complete and an optimal policy has been calculated, the radar then moves into the acting phase where it uses this policy and the state being measured to find the best action to take for the next time step. Figure 3.3 shows the radar avoiding the RFI that is located in the center band: [0 0 1 0 0], by operating in the two right most bands: [0 0 0 1 1]. These figures show the capabilities of the MDP implemented in this real system for low complexity RFI.

Missed opportunities and collisions are calculated by analyzing the spectrogram plots. The metrics used to directly compare the different modes of the system are SINR and Bandwidth, which is done through the use of Range-Doppler radar plots. These plots have Doppler shift in Hertz on the x-axis, range of the target in kilometers on the y-axis with the power of the signal on the z-axis. Figure 3.4 shows the Range-Doppler plot of the radar

operating in full bandwidth mode in the 690 MHz band shown previously. With the radar and the communication system interfering with each other, the noise floor is around -70 dBm with a target showing up at 0 Hz Doppler shift and about 250 meters. The SINR of this received signal averaged over all range-doppler bins is about 31 dB. Figure 3.6 shows the Range-Doppler plot for the MDP mode where the radar is now avoiding the interference present in the 690 MHz band. We can see the noise floor drops away and the target becomes much clearer. The SINR in this plot increases by 10 dB so the average SINR for this mode is around 41 dB. Figures 3.5 and 3.7 show the 2-D range and Doppler plots for each mode (full bandwidth and MDP approaches). These plots were generated by plotting the range at 0 Hz Doppler frequency and plotting Doppler frequency at 205 meters range (the target is located at 205 meters and 0 Hz). These plots, in addition to the Range-Doppler maps, show the effect of avoiding interference on target detection. A target is determined to exist if there is a high power signal being received that is greater than the surrounding noise. In Figure 3.5 it is unclear if a target is present in the range plot, however once we start avoiding interference in Figure 3.7 then we can clearly see a target present at roughly 200 meters.

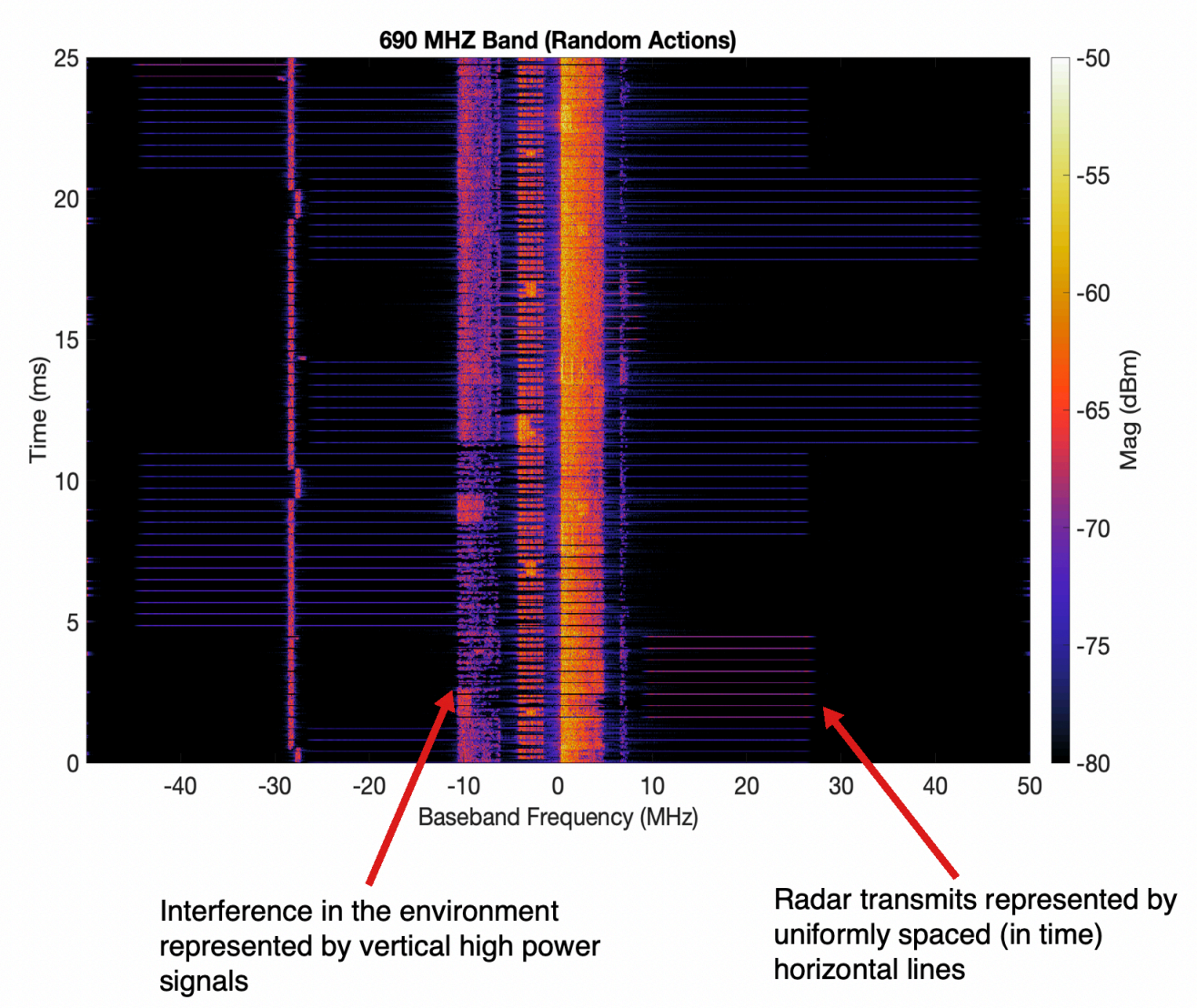

Figure 3.2: Spectrogram of the radar taking random actions in baseband where 0 MHz represents 690 MHz. This is the learning phase where random actions are taken in order to build up statistics on which actions perform with high rewards in certain states. Any vertical lines resulting from interference are simply the side-lobes of the said interference that is being generated.

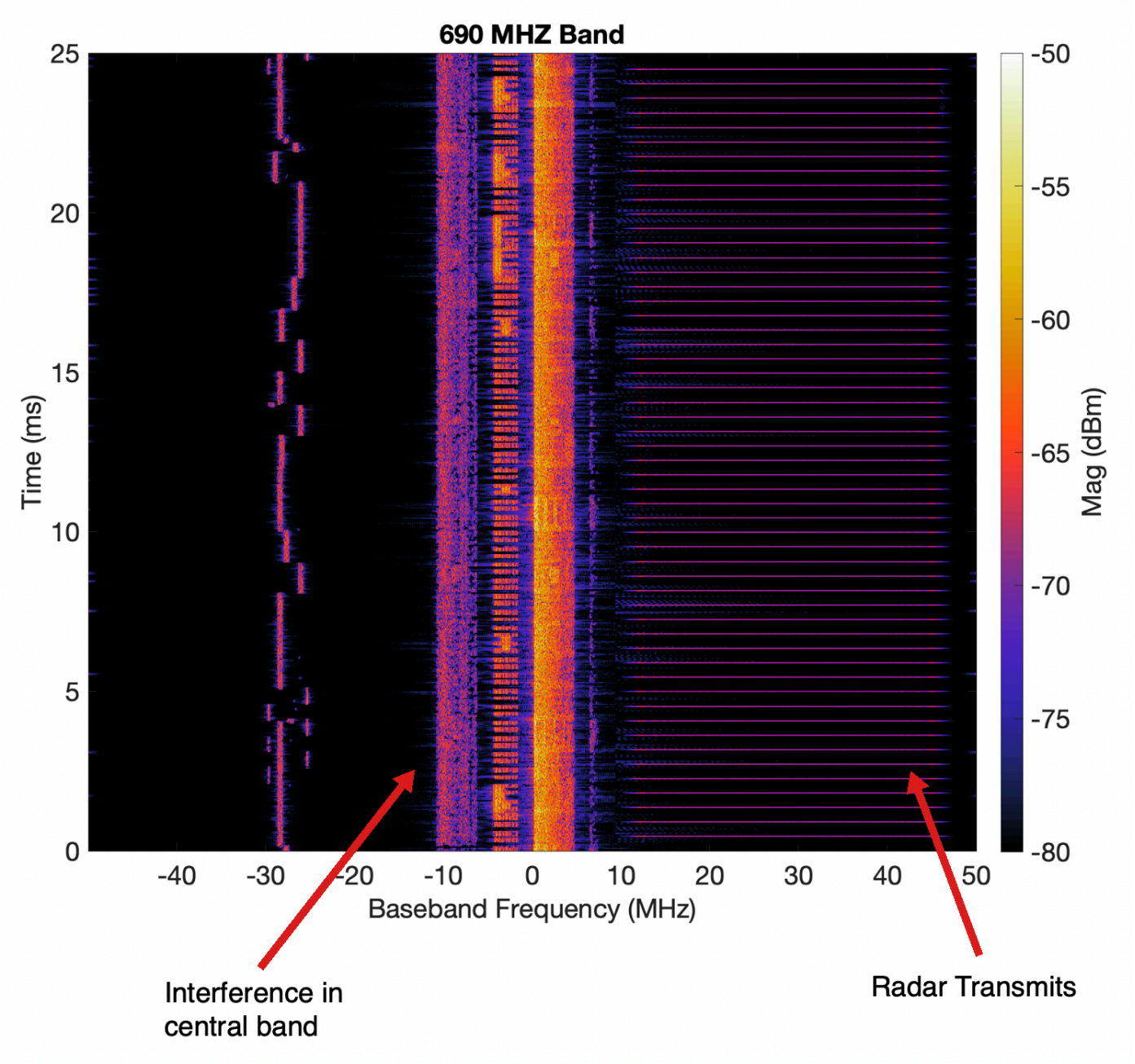

Figure 3.3: Spectrogram of the radar taking actions based upon the policy. This is the "acting" phase where the policy has been calculated so the radar acts based on which state is being observed.

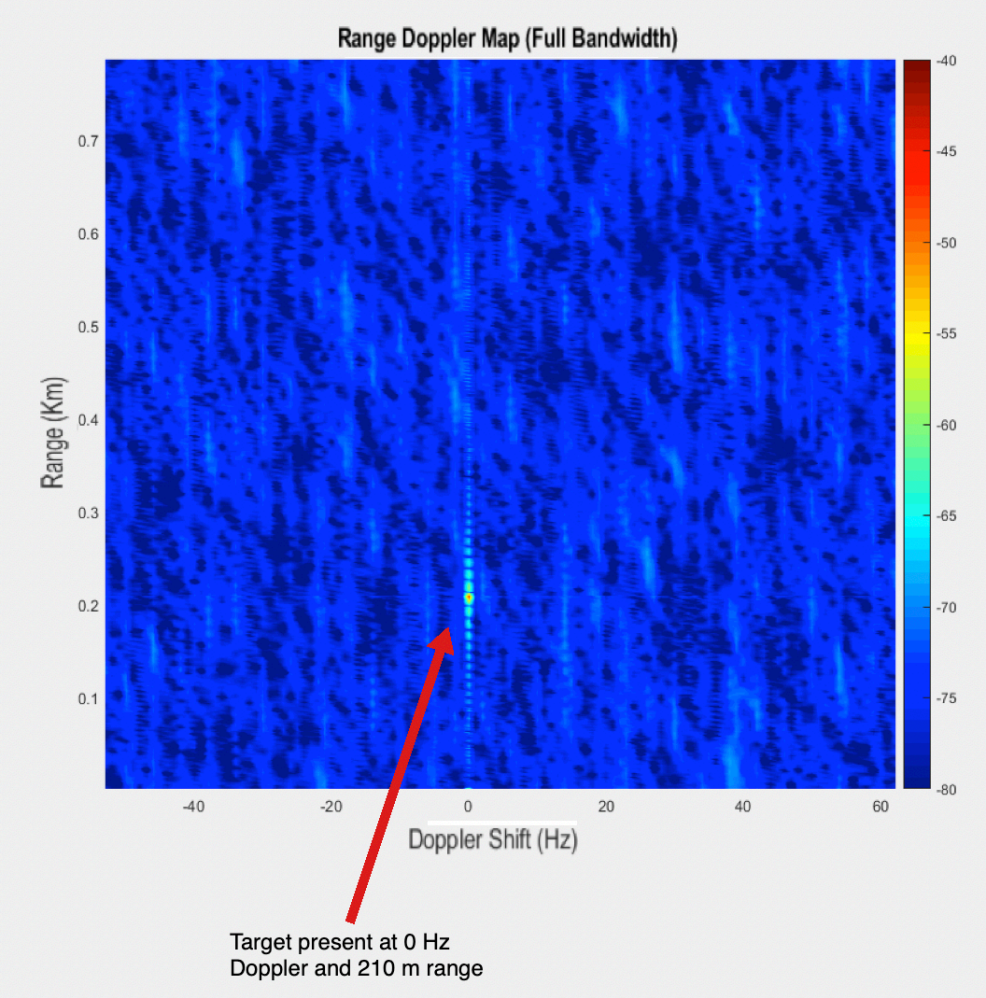

Figure 3.4: The Range-Doppler plot for the mode where the radar is operating in full bandwidth despite the existence of interference. We can see the target present in the environment

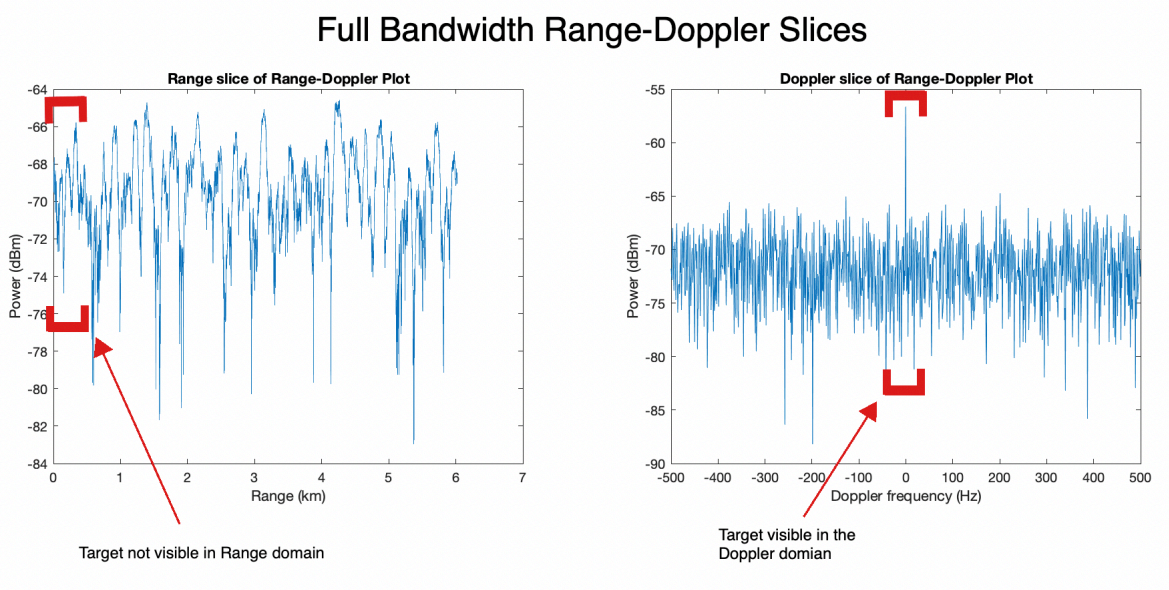

Figure 3.5: The 2-D range and Doppler plots for the mode where the radar is operating in full bandwidth despite the existence of interference. We can see the target present in the only in the Doppler plot. A target is difficult to discern in the range plot.

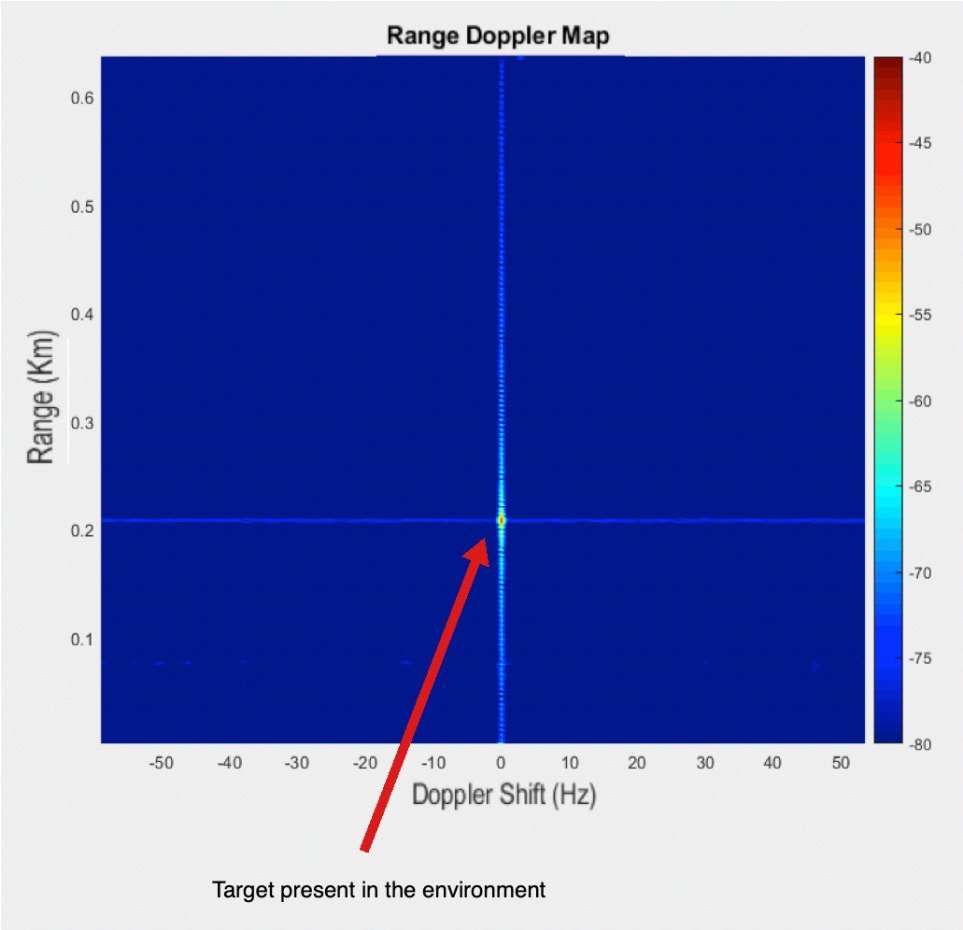

Figure 3.6: The Range-Doppler plot for the MDP mode where the radar is avoiding the RFI. The noise floor is lower than the full bandwidth mode and the total SINR is higher resulting in a higher probability of detection.

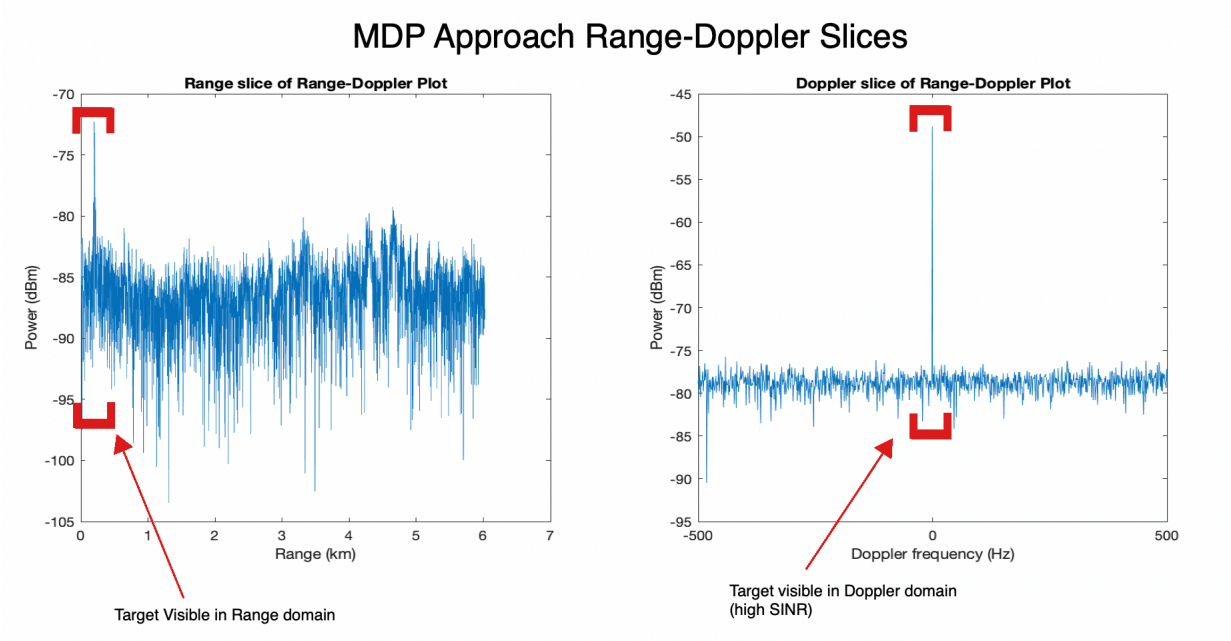

Figure 3.7: The 2-D range and Doppler plots for the MDP mode where the radar is avoiding the RFI. Comparing to Figure 3.5, we can see that the target is much easier to identify in both range and Doppler plots. The difference between the returning signal power and the average noise floor is much greater than the full bandwidth approach.

From Figure 3.3 we see how effective the MDP is in identifying and avoiding the RFI present in the environment and from Figures 3.4 and 3.6 we can see the improvements to the radar processing when going from the full bandwidth mode to the MDP mode. The 690 MHz interference band is relatively low complexity interference; a more illuminating interference type to compare methods is the moderately complex Swept Tone Interferer. This interference type sweeps across the 100 MHz frequency band we are observing in five steps, each step lasting for approximately 16 ms. This interference type will be evaluated by observing the missed opportunity, collisions and SINR metrics as well as probability of detection (necessary to radar) for each of the three avoidance cases (Full Bandwidth, Reactive and MDP). Table 3.1 reveals this comparison directly.

Table 3.1: Comparing avoidance methods in terms of defined metrics for the swept tone case.

	Full Bandwidth	Reactive	MDP
Collision Rate	1.0	0.098	0.0
Missed Opp. Rate	0.0	0.097	0.44
Total Error Rate	1.0	0.195	0.44
SINR (dB)	33.2	34.8	39.8
Prob. of Detection	0.0	0.0	1.0

From Table 3.1 we can see how the collision rate decreases going from Full Bandwidth to Reactive and them to the MDP approach. The missed opportunity rate however, increases in that same order. This results in the Reactive mode having the lowest total ”error” rate (sum of collision rate and missed opportunity rate), however, it still interferes with other communication systems resulting in a significant loss of SINR (5dB). This loss in SINR is the difference between target detection and missing the target. While the MDP approach did not always take advantage of all the bands it could occupy, it was able to accurately predict where the interference would be and preemptively avoid it. This resulted in the MDP approach always being able to detect the target while the other methods were not

able to detect a target due to the mutual interference. Figure 3.8 shows the Range-Doppler plot for the Full Bandwidth method where the interference is sweeping across the 100 MHz band and the radar is not taking any action to avoid it. The target, which is located at 0 Hz and approximately 250 m, is very hard to see in the plot and in fact the radar is not able to detect the target at all. The Range-Doppler plot for the Reactive method is similar to the Full Bandwidth case and not shown. Figure 3.10 shows the Range-Doppler plot for the MDP method where the radar has spent some time learning the optimal policy and is now acting upon that policy to predict and avoid the interference whenever possible. While there still exist some low power noise in this Range-Doppler plot, the location of the target is much clearer to the user and the radar is able to detect this target's range and Doppler shift accurately. Table 3.1 revealed a 6.6 dB increase from the Full Bandwidth mode to the MDP mode which can be clearly seen when comparing Figures 3.8 and 3.10.

Figures 3.9 and 3.11 are images of the range and Doppler slices of the Range-Doppler plot where the target is located (the same as what the plots done for the 690 MHz case). It is more clear from these plots that the interference causes the target to be hidden behind high power interference signals, however once we start avoiding the interference the target becomes clear in the environment.

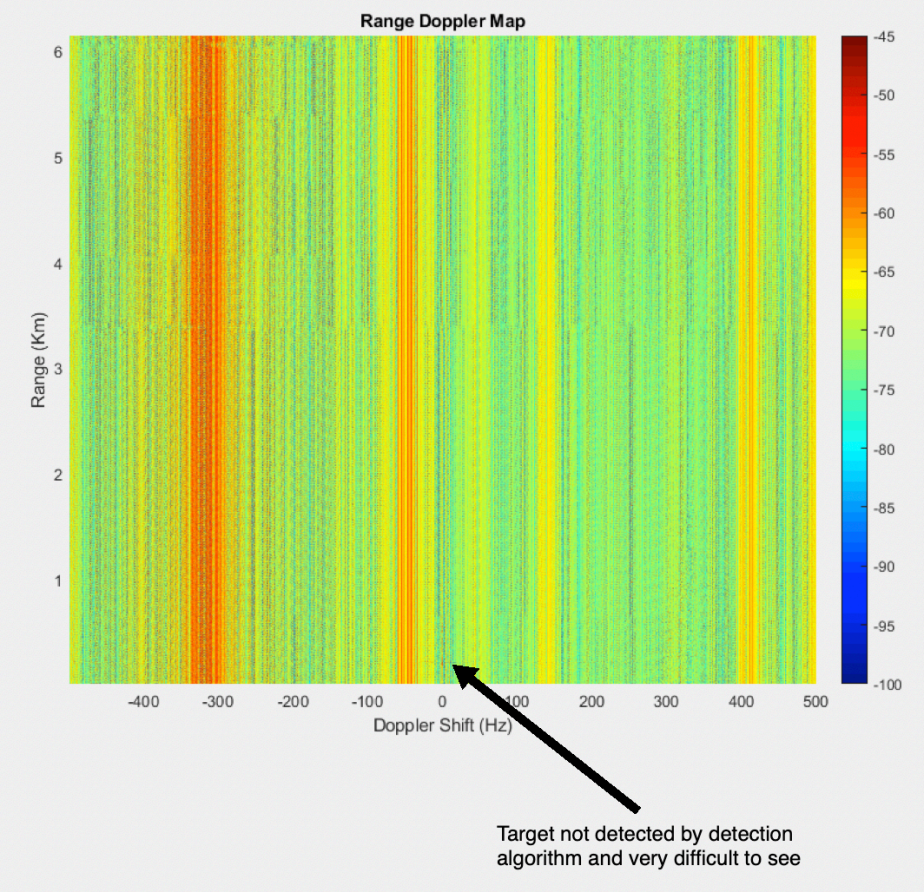

Figure 3.8: The Range-Doppler plot for the Full Bandwidth Mode in the Sweep interferer case. It is very difficult to see any target present and there is large interference from the swept tone.

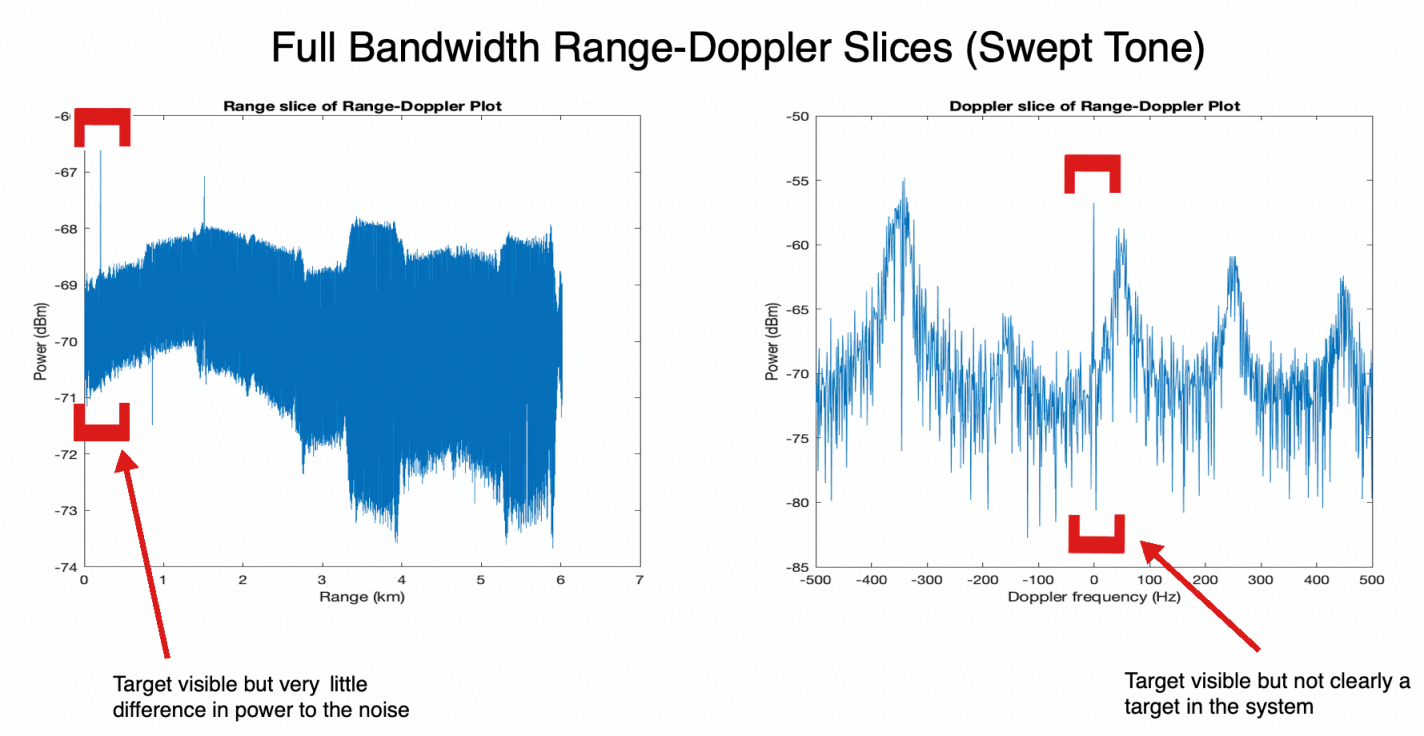

Figure 3.9: The range and Doppler plots for the Full Bandwidth Mode in the Sweep interferer case. It is very difficult to see any target present in either plot in the area where the target should be located. The difference between the target power and average noise power is very low.

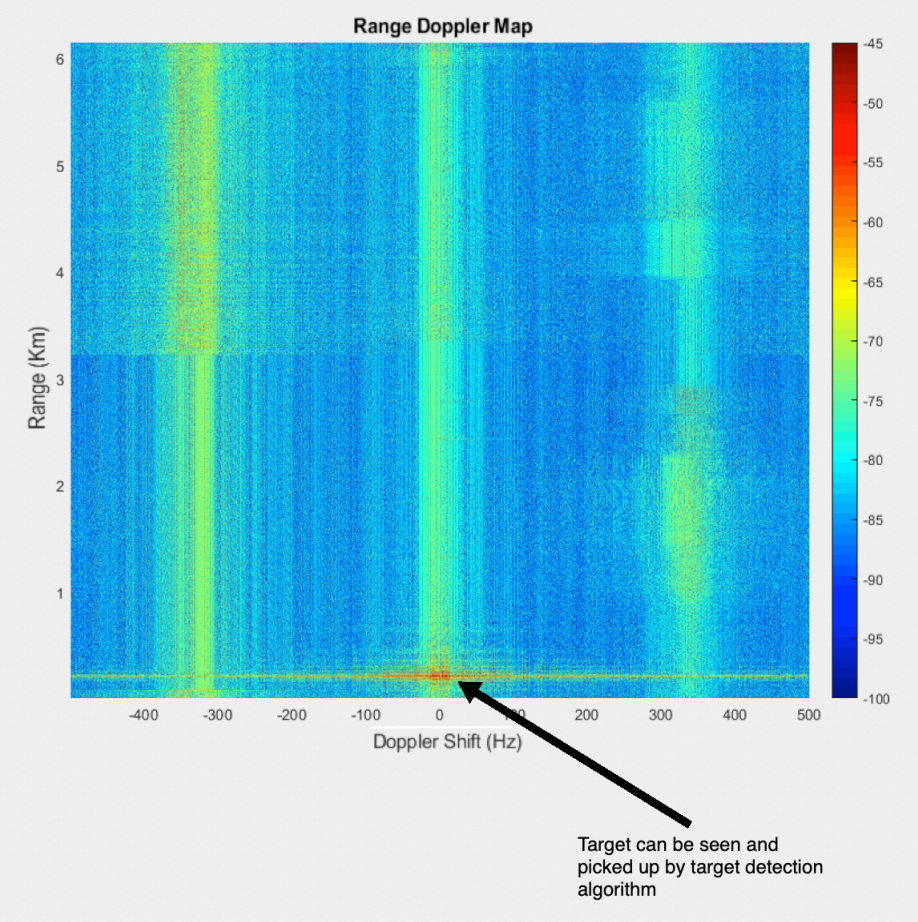

Figure 3.10: The Range-Doppler plot for the MDP mode where the radar is avoiding the RFI in the sweep interferer case. We can now see the target and the interference has reduced in power greatly. It hasn't gone away completely since the matched filter only reduces the effect of the interference and doesn't completely eliminate it.

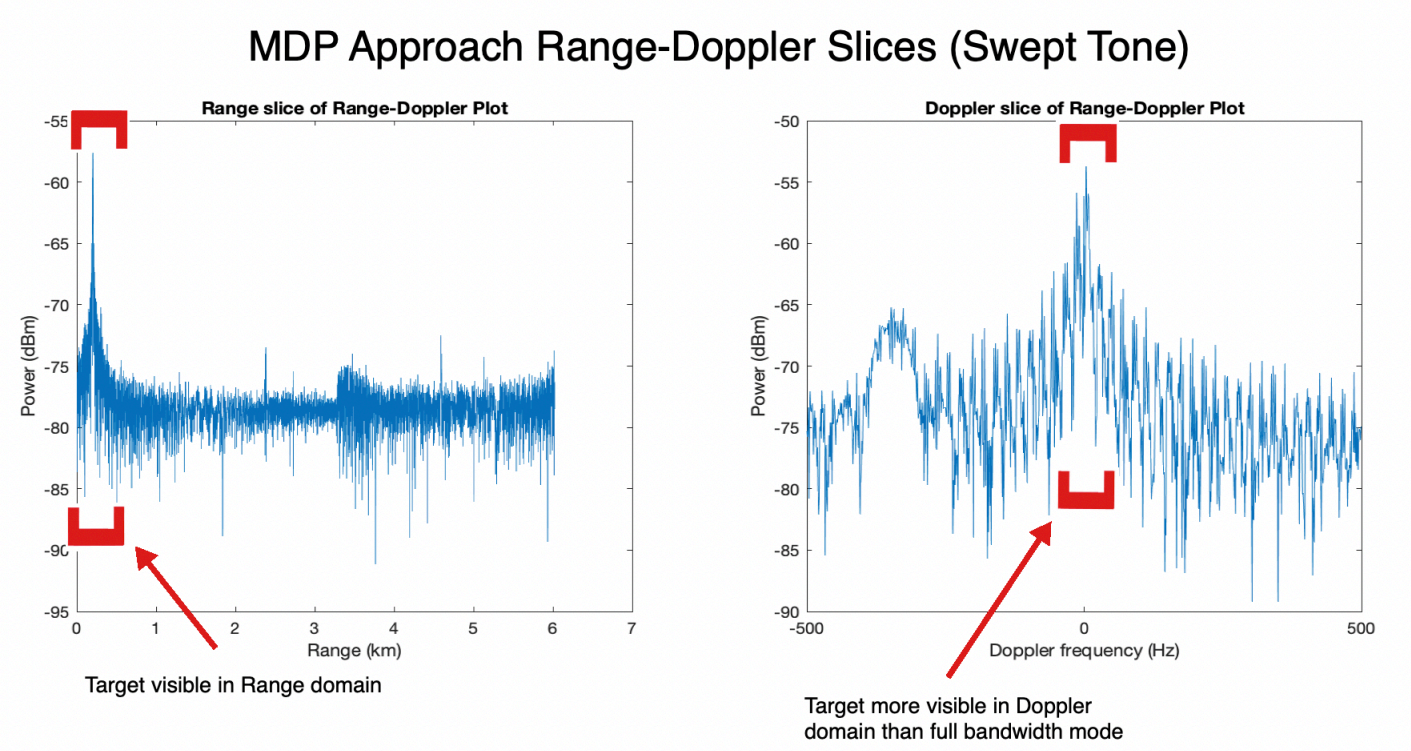

Figure 3.11: The range and Doppler plots for the MDP mode where the radar is avoiding the RFI in the sweep interferer case. We can now see that the interference has reduced in power greatly, revealing targets in range and in doppler. There is still interference present due to the side-lobes of the interference interfering with the signal, but it is much more clear that a target is present in the system. The difference between the target power and the average noise power is much greater than the full bandwidth approach.

The MDP method was tested on real world recorded interference ranging from low complexity to high complexity. Portions of these interference types are displayed in Figure 3.12. The interference taken from the 690 MHz and 1750 MHz bands are considered low complexity interference because the 690 MHz band interference is mostly stationary while the 1750 MHz band interference is infrequent and narrow band. The Swept Tone interferer is considered medium complexity because it is continuously changing bands however it is very deterministic. The 2450 MHz band is considered a high complexity interferer because it is very dynamic and the optimal action to take in this environment is not immediately known.

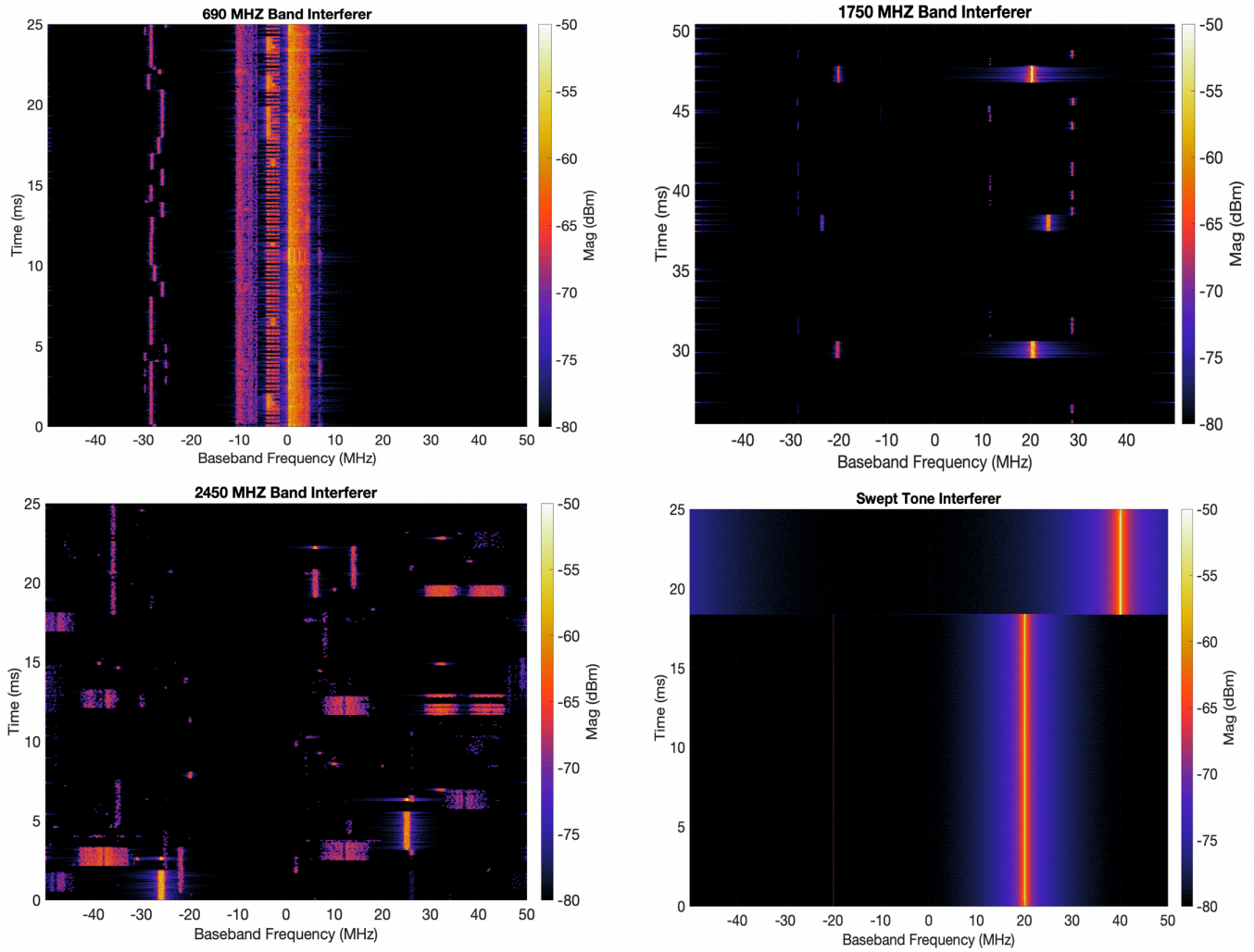

Figure 3.12: The spectrogram plots of just the various interference types. Interference measured from the 690 MHZ band is in the top left plot, 1750 MHz band in the top right plot, 2450 MHz band in the bottom left plot and the generated sweep interferer in the bottom right plot.

Table 3.2 shows the SINR (in dB) values of the two cases previously explored as well as two new interference cases that are considered high complexity (very dynamic and changing constantly on the millisecond level or very congested with little room to operate). These two new interference cases are measured interference from the 1750 MHz Band and the 2450 MHz Band respectively. These measurements were taken with the Software Defined Radar for the purposes of displaying this data as Receiver Operating Characteristic (ROC) curves.

Table 3.2: Comparing SINR (in dB) across multiple interference types and across all methods of avoidance

	Full Bandwidth	Reactive	MDP	Complexity
690 MHz	11.6	16.7	17.2	Low
1750 MHz	13.5	16.9	17.3	Low
2450 MHz	13.4	13.7	16.1	High
Swept Tone	13.2	14.8	19.8	Medium

We can see that even for the higher complexity interference types (swept tone and 2450 MHz), the MDP achieves greater SINR which corresponds to a greater probability of detection and a lower probability of false alarm. A ROC curve shows the probability of false alarm vs probability of detection based upon the measured SINR. These curves allow for the determination of the probability of detection (P_d) based upon the SINR and probability of false alarm (P_{fa}). Maintaining a high P_d and a low P_{fa} is the desired behavior of the radar. In Figure 3.13 we can see that the SINR curves shift to the left as we go from the Full Bandwidth mode to the Reactive mode, and then again when going to the MDP mode. For the purposes of showing these results, the noise was assumed to be zero mean Gaussian distributed noise. This plot indicates an increase in performance since the shift left is a shift closer to the optimal behavior of a P_d equal to 1 and a P_{fa} equal to 0. From this figure it can clearly be seen that the Reactive mode is an improvement upon simply acting in Full Bandwidth and the MDP mode is an improvement upon the Reactive mode all in terms of radar metrics. In the 1750 MHz and 2450 MHz cases this improvement is marginal, however this improvement is more pronounced in the other interference types. It should be noted that these ROC curves were built up in MATLAB using measured SINR values from the physical system (again assuming Gaussian noise). The SDR system is currently not set up to generate probability of detection values.

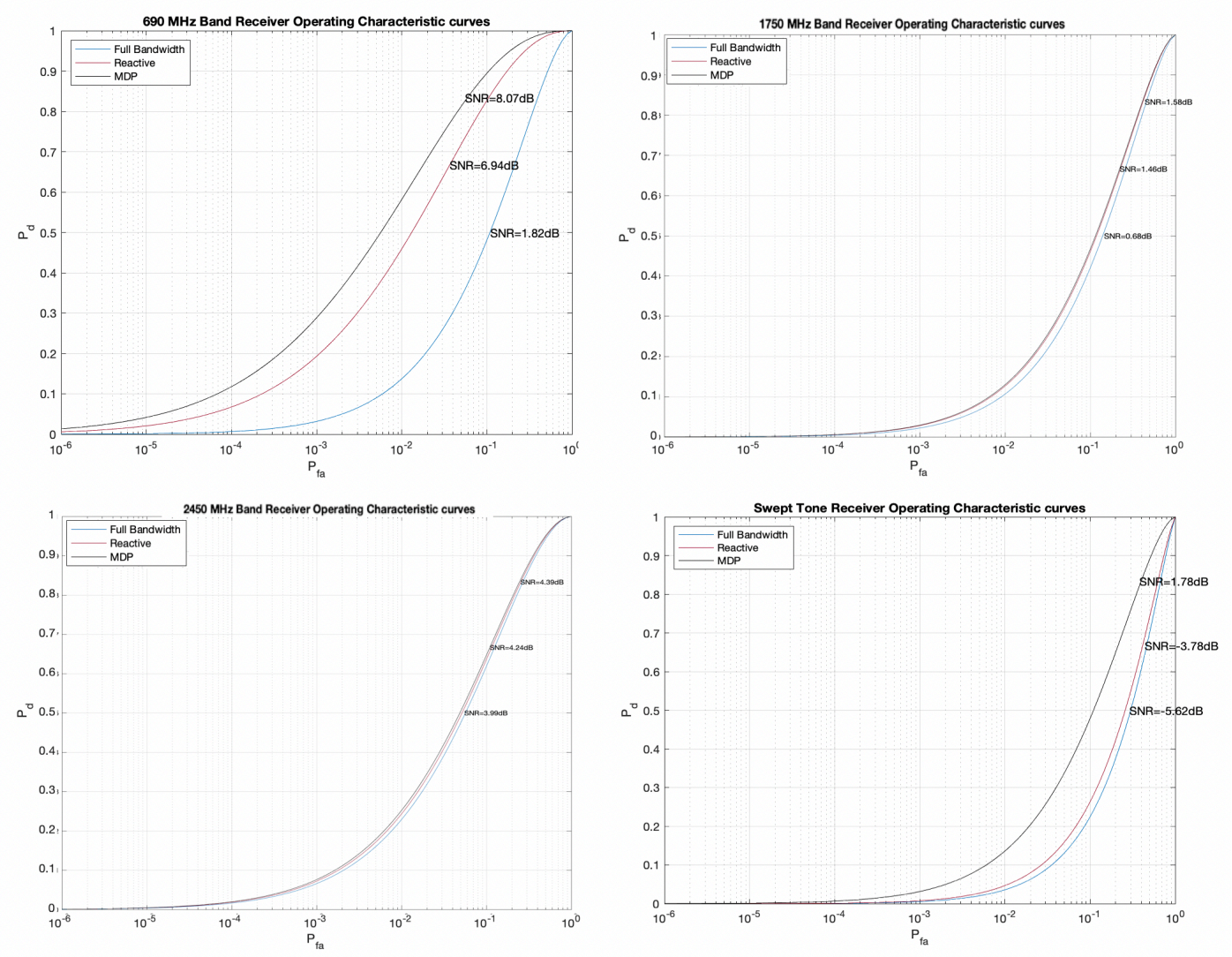

Figure 3.13: The ROC curves for each of the 4 cases mentioned in Table 3.2. The top left shows the 690 MHz band, top right shows 1750 MHz band, bottom left shows 2450 MHz band and bottom right shows the swept tone interferer. The black curves represents the MDP approach, the red curve represents the reactive (5 ms) approach, and the blue curve represents the full bandwidth approach.

While the MDP method shows improvement over current methods in both the 690 MHz case and the Swept Tone interferer as well as the more complex interference cases, there exists several limitations in the physical implementation that prevents maximum performance. The strengths and weaknesses of the MDP method are discussed in section 3.3. A possible solution to the weaknesses of the MDP are also discussed in section 3.3 which will be a part of the future work on this project.

3.3 Discussion

There are several strengths, but also some limitations to this method. One strength is that the MDP approach is shown to be able to accurately predict and avoid RFI, which increases the average SINR of the received signal. With a target present, this means that the MDP allows for target detection in cases where the reactive method or constant bandwidth approach do not. This is shown in Figure 3.14 where the difference between the SINR and bandwidths of the various methods (full bandwidth, reactive and MDP) are clearly shown for the 690 MHz case. This increase in SINR allows for increased P_d and decreased P_{fa} which shows improvement in radar detection by the MDP. It should also be noted that the MDP has the least amount of bandwidth which corresponds to the least accurate range resolution. The full bandwidth approach results in 1.5 meters of range resolution, the reactive approach results in 3.4 meters of range resolution and the MDP approach results in 4.3 meters of range resolution. This is due to the current reward structure which places higher priority on SINR in this environment than it does upon the bandwidth. Full bandwidth achieves the best range resolution however results in the lowest bandwidth where the MDP results in the worst range resolution but the greatest SINR. There are different reward structures that allow for more bandwidth but will reduce the SINR, however it is significant to note that

the maximum SINR the MDP can produce is greater than the other measured SINR values of the other methods.

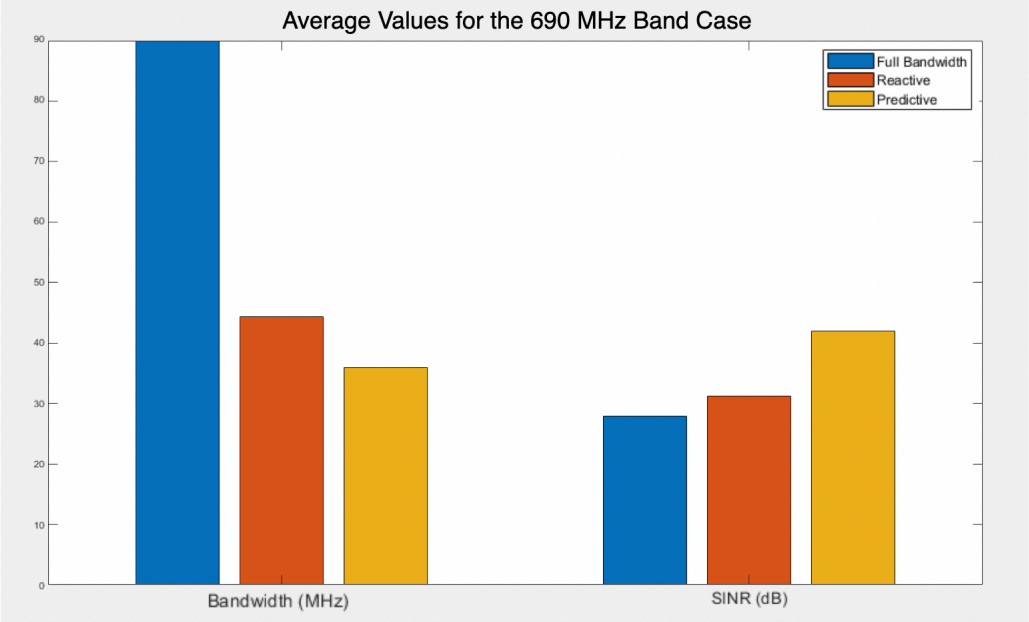

Figure 3.14: Comparing the average SINR and bandwidths for the three current adaptive methods: Full Bandwidth, Reactive and MDP

Another strength is that this implementation includes online learning. This means that the radar is learning continuously instead of having a block of time at the beginning for offline learning, and then simply acting on what it has learned. In the simulation we only had offline learning, but with the real system implementation we have the option for both offline and online learning. This allows the radar to ”reset” its optimal policy if the interference changes drastically. A setting could be created that will reset the policy if the collision rate becomes too high, as this would indicate that the current policy is no longer valid and that learning should begin anew.

The MDP also reduces the collision rate in the more dynamic spectrum where reacting to interference is too slow to be of any help. While it does not always reduce the missed opportunity rate and the Reactive mode has a better missed opportunity rate, the MDP

mode still has the option of changing its priority to maximizing bandwidth over maximizing SINR. This option would however also increase the collision rate of this method if the number of memory states is not also increased. Both metrics could be improved with increased memory, this is shown in Figures 3.15 and 3.16. These plots show the results of testing the MDP on the same real interference type with varying amounts of memory (this was done in simulation so that memory could be implemented). This reveals the necessity of adding memory to increase performance.

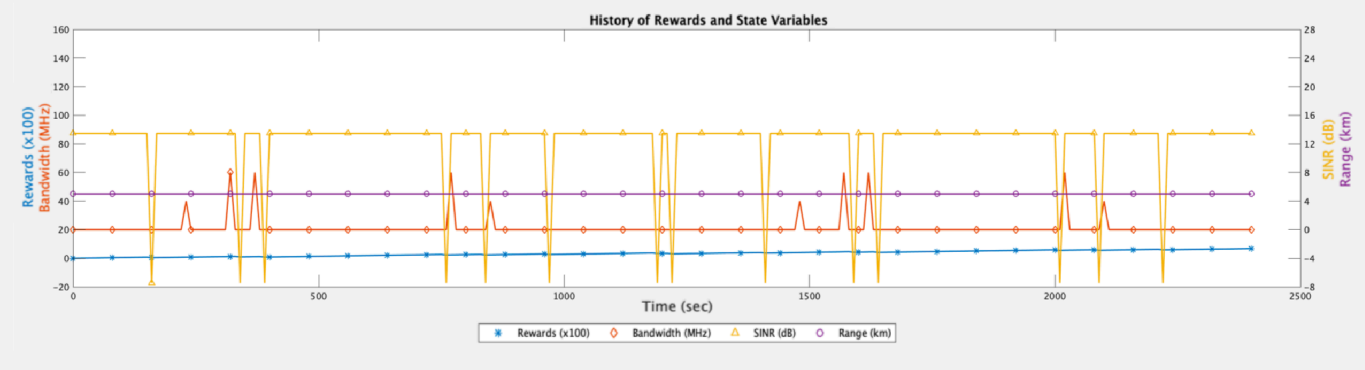

Figure 3.15: The SINR, bandwidth and rewards of the radar tested with one memory state on the 2450 MHz band. There are several drops in SINR and the bandwidth remains at 20 MHz for the majority of the time.

The nature of the MDP and its reward structure is another advantage for the system. With the reward structure, it is simple for the user to tailor the system to how they wish the radar to behave. With minor changes to the rewards structure, the user can change the radar to avoid RFI at all costs, or to be very aggressive in the bands it chooses to occupy (these being just two examples of behavior). This variability in behavior as defined by the user is a major strength in implementing MDP method. The reward structure also allows for additional metrics to be considered if the user chooses to value those metrics as well. For example, if the user does not wish the radar to change bands as often, they could implement a structure that would increase the reward given if the previous action was the same as the

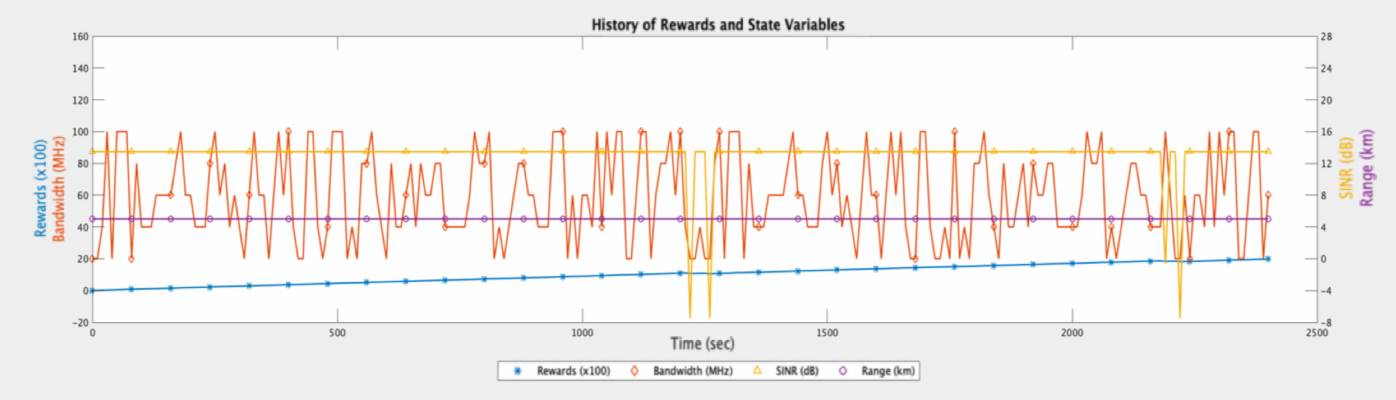

Figure 3.16: The SINR, bandwidth and rewards of the radar tested with three memory states. This is a huge improvement over one memory state as there are much fewer drops in SINR (collisions) and the radar takes more advantage of open bands (reducing missed opportunities).

current action.

A significant limitation to this method is the limited memory capabilities. As previously discussed, adding memory into the MDP causes the state space to grow exponentially. In the simulation, the MDP could handle three or four memory states when operating with 5 sub bands. In the LabVIEW implementation, the MDP can only handle one memory state and a maximum of five sub bands. This is not ideal, because the more memory states in the MDP the more accurate the prediction for the next state will be, which will allow for a decreased collision and missed opportunity rate (as shown in Figures 3.15 and 3.16). Additionally, it was the goal of the project to eventually increase the number of sub bands up to 10, but the current implementation is not able to operate with 10 sub bands. Another significant weakness is with the MDP structure itself. The MDP is not able to act on states that it has not seen before, it will simply choose the default action which, in this case, ends up being in the single right-most sub band. This is obviously not ideal when the goal is to maximize bandwidth while minimizing interference. These weaknesses have led us to explore a machine learning method that would solve these issues; that method being the implementation of a Deep-Q Network described in Section 2.5.

Chapter 4

Deep-Q Network (DQN) Application

For a description of the DQN please refer back to Chapter 2.

4.1 DQN Results

As we saw in Chapter 3, increasing memory will increase the performance of our radar, however the implementation of the MDP can only handle one memory state. The DQN was suggested as an alternative to the MDP since it does not need the transition and reward matrices and is much less computationally complex. This leads to the DQN being able to record many memory states with little impact on the complexity of the system.

The first objective is to ensure that the DQN could perform as well as the MDP approach given in [16]. They need to perform the same in terms of finding the optimal action to take when presented with interference from some communication system to ensure the DQN does not perform worse than the MDP and is also able to find the optimal policy. The second objective is to show that the DQN outperforms the MDP by being able to operate in a larger state space and operate with states not previously seen in training. We will start with five bands (each with 20 MHz width) and will represent this using the following notation: [1 0 0 0 0], where a 1 means that particular band is occupied and a 0 means that band is open. There will be three types of plots in this section that represent the performance of the system. The first plot (see Figure 4.2), located in the upper left of the figure, contains two

curves and represents the policy found by the MDP; the purple curve represents the SINR, the orange curve represents the bandwidth in megahertz over which the radar is operating. The second plot, located in the bottom left of the figure, represents the optimal policy found by the DQN and contains the same curves as the previous plot. Both plots progress with some arbitrary representation of time. The third plot, located on the right side of the figure, compares the missed opportunities and collisions experienced by both solvers. We define a collision to be when the radar and the communication system are operating in the same sub band, creating interference. A missed opportunity is defined as the contiguous bands that the radar is not, but should be, operating in. For example, a communication system operating in the left most band, [1 0 0 0 0], and the radar operating in the right most band [0 0 0 0 1], would be considered to have three missed opportunities since the radar could operate in the three middle sub bands as well. These four metrics are what we will be using to determine the performance of the cognitive radar approach. The fewer collisions and missed opportunities the better, and we wish to optimize SINR and bandwidth so that the radar operates in as many bands as possible without interfering significantly.

4.1.1 Baseline Results

Figure 4.1 verifies that the DQN is able to perform as well as the MDP. The interference in this case is a triangle sweep interference, where the communication systems starts in the first band, [1 0 0 0 0], moves to the second, [0 1 0 0 0], until it reaches the right-most band, then it works its way back to the initial state. This pattern simply repeats. We can see from Figure 4.1 that both solvers are able to maintain high bandwidth while observing no loss in SINR. Both solvers accurately predict where the interference will be and avoids the said interference by choosing the optimal action. The missed opportunities and collisions plot was

not included in this example, because both methods resulted in zero missed opportunities and zero collisions.

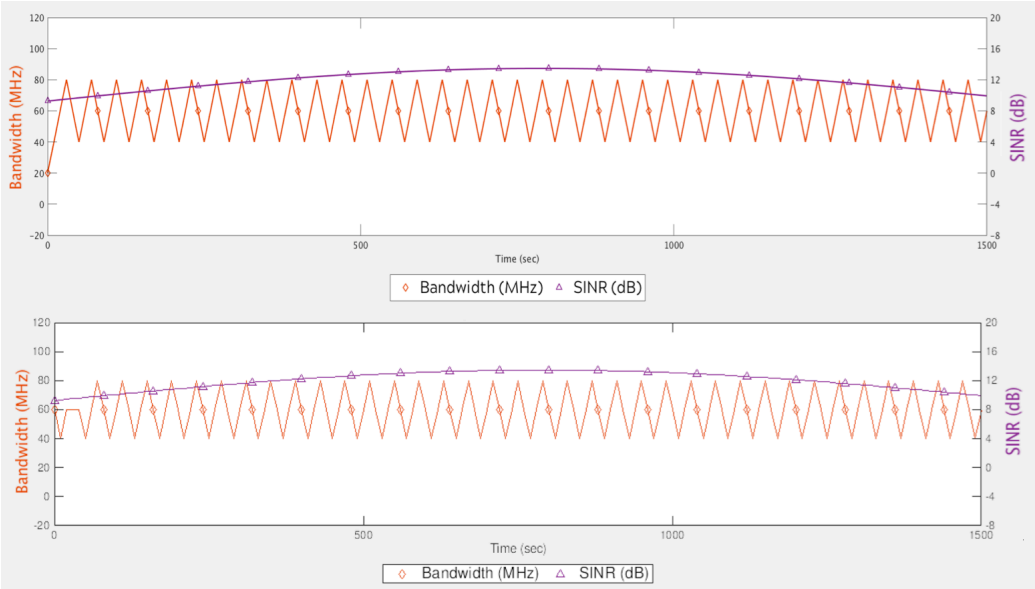

Figure 4.1: Comparison between DQN (bottom plot) and MDP (top plot) showing equal capabilities. Both approaches have the same results in that they are able to predict the next interference state and avoid it, maintaining high SINR. Their bandwidths are also exactly the same indicating both approaches chose to take the same (optimal) action.

4.1.2 Impact of "New" Interference

There are several limitations when experiencing interference that is different from training. The MDP requires many training runs to record all possible states and all possible actions for each state in order to accurately find an optimal policy. If it does not see all states or all actions with each state it cannot find an optimal policy. The DQN will also not be able to find an optimal policy right away, however the DQN employs online learning so this limitation is much less severe than the MDP as it will eventually learn the optimal policy.

Figure 4.2 shows that the DQN outperforms the MDP when each solver is introduced

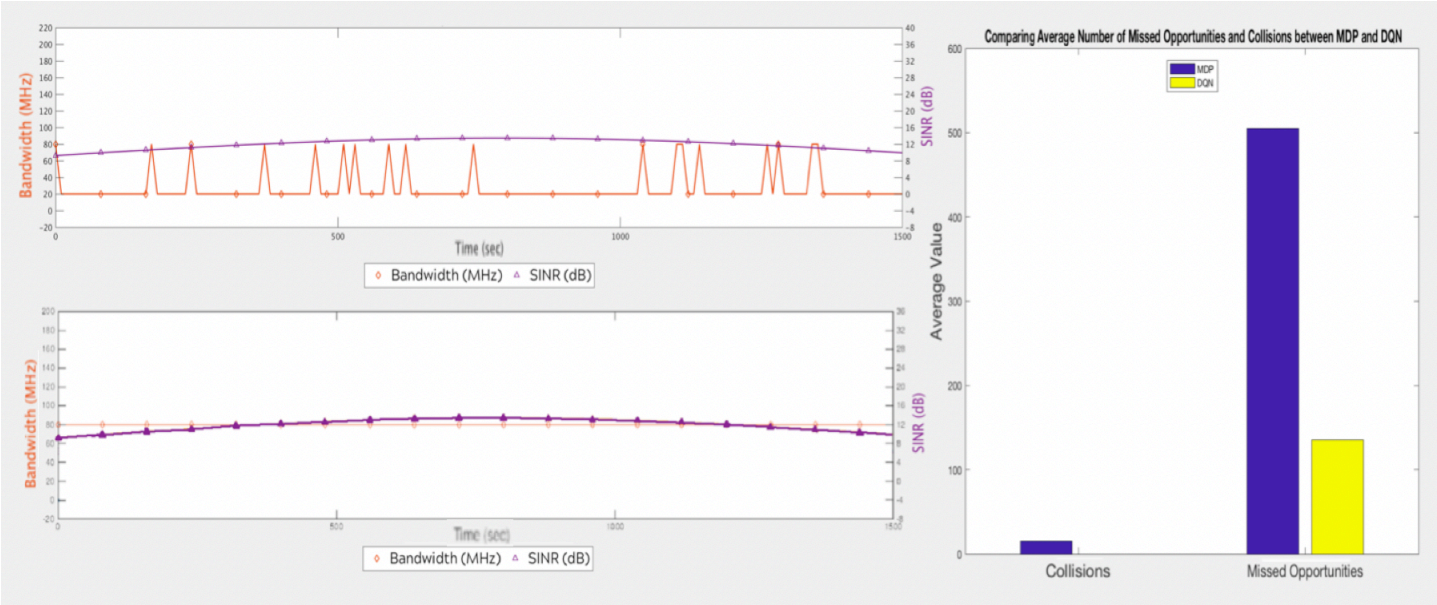

Figure 4.2: Comparing DQN (bottom plot) and MDP (top plot) solutions to new states appearing. The MDP maintains high SINR but its bandwidth remains at 20 MHz the majority of the time. This means that the radar is only operating in one band resulting in many missed opportunities. The DQN is able to reduce the amount of missed opportunities because it is not restricted as the MDP is in its policy.

to a state it has not been trained on. The interference being trained on is constant interference where the communication system is only operating in the first sub band, [1 0 0 0 0]. The interference the policy is being evaluated on is intermittent interference, where the communication system only operates in the first band 10 percent of the time. We see that the MDP is not able to choose an optimal action because it has not seen the state [0 0 0 0 0] before, so the radar chooses to operate in a single band (the default policy) even though it interferes with the communication system. The DQN on the other hand, maintains its bandwidth because there is no degradation in SINR, and it has learned that operating in the other four bands produces the highest reward for all of the interference types it has seen.

4.1.3 Complex Interference

Figure 4.3 shows that the DQN outperforms the MDP when the computational complexity increases, since the MDP is unable to implement memory due to its limitations in state space [1]. When the state space is increased, the computational complexity growth is exponential for MDPs and linear for DQNs. For these plots, the radar is now capable of operating in 10 bands, and the solvers are trained on both constant and intermittent interference, and evaluated on intermittent interference. We can see that the MDP is only able to react to interference appearing, resulting in loss of SINR and drops in bandwidth as well as missed opportunities and collisions. This is due to the fact that with only one memory state, the MDP is unable to determine if the interference is constant or intermittent and reacts to any interference that appears. The DQN on the other hand, operates in all 10 bands. This is because with two memory states, it recognizes the interference type is intermittent, and the communication system will vacate the band almost immediately after entering the band. This also produces SINR drops but maintains high bandwidth, resulting in the same number of collisions as the MDP but no missed opportunities.

Despite this improvement we are able to see from the previous results, the DQN is still not able to perform as well as it should. In Figure 4.4 we see the DQN results when trained on Constant, Intermittent and Triangle Sweep interferers and being evaluated on the Triangle Sweep interferer with two memory states. We can clearly see from the SINR curve that we are interfering with the communication system very frequently. This can also be shown in the bottom plot. Since there are two memory states, the DQN should be able to identify that this is not Constant or Intermittent interference and in fact it does since it is able to avoid the interference occasionally. However, the actions the DQN chooses to take are

[1] With 10 bands, the matrix that needs to be built up for the MDP approach grows so big that the computer is unable to hold that much data at once, and is unable to complete the simulation

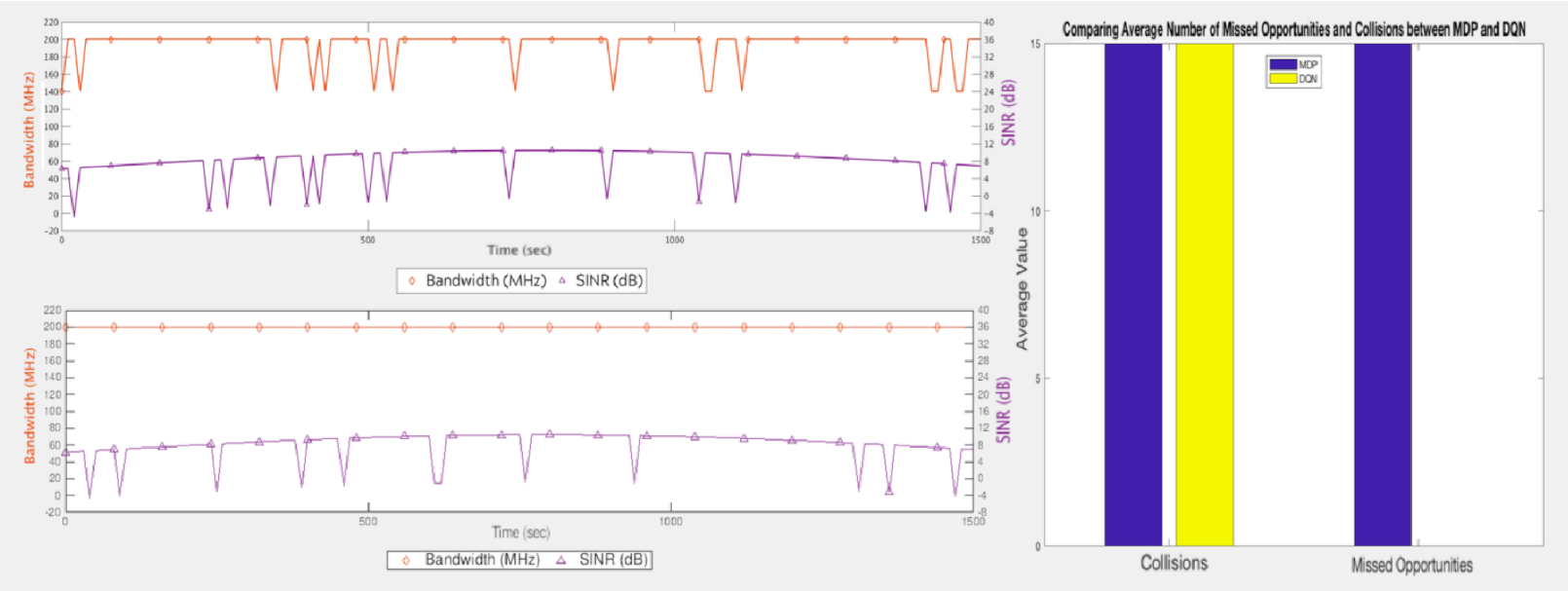

Figure 4.3: Comparing DQN (bottom plot) and MDP (top plot) solutions when memory is needed for optimal solution. The MDP produces both collisions and missed opportunities since it is not able to differentiate between constant interference and intermittent interference with only one memory state. The DQN is able to act with two memory states so it is able to differentiate between the two interference types which results in the same number of missed opportunites but no collisions.

oversimplified and based upon the learning that was done in the Constant and Intermittent interferer cases which are by no means optimal for this case. The shows a need for a Deep Recurrent Neural Network, which adds the previous output of the NN as an input in the next time slot, effectively adding memory to the system and potentially solving this issue.

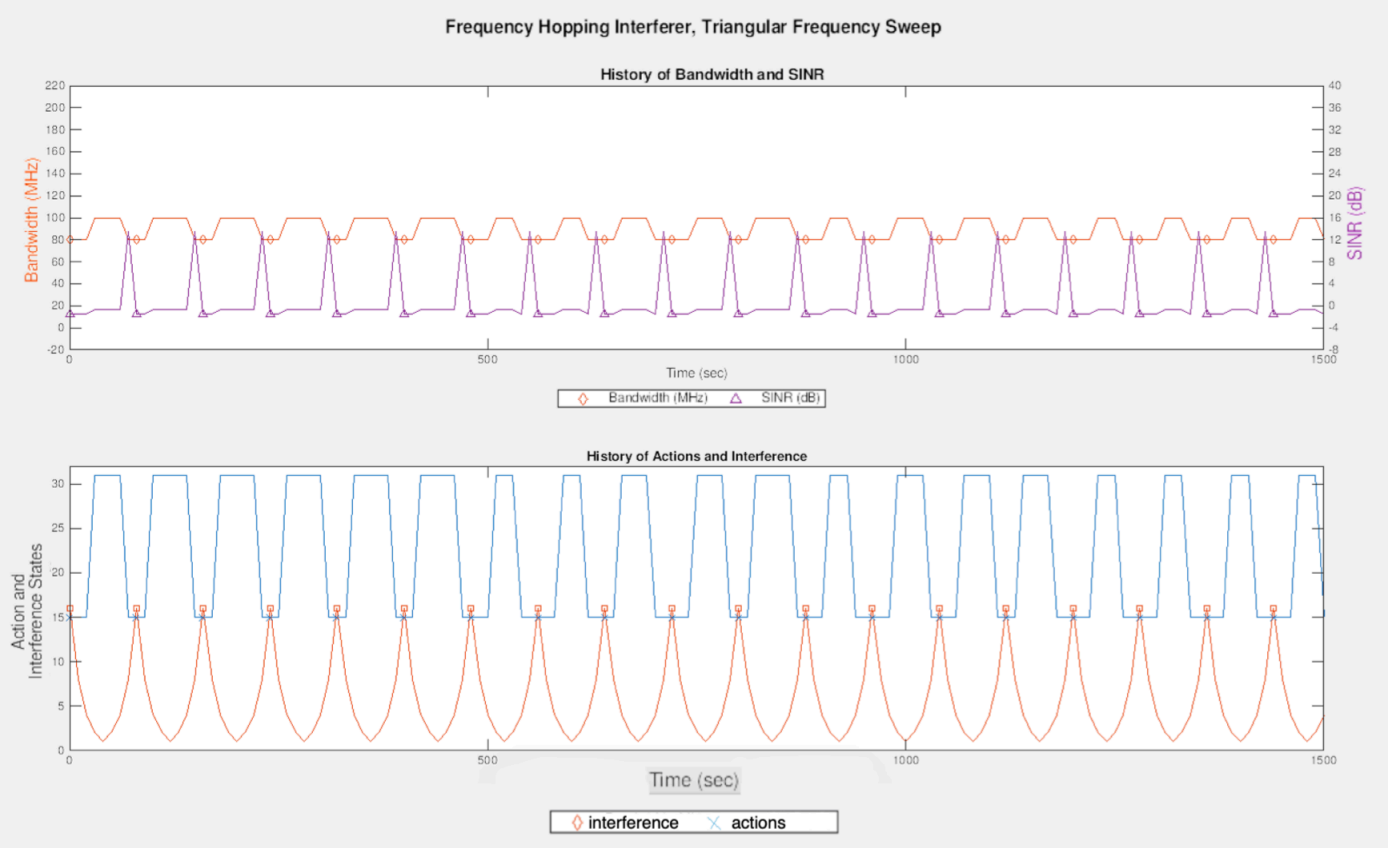

Figure 4.4: DQN results when trained on Constant, Intermittent and Triangle Sweep interferer and evaluated on the Triangle Sweep interferer. The top plot shows the SINR and bandwidth. The bandwidth remains relatively high but the SINR drops below 0 very frequently indicating the DQN is not able to predict the difference between these interference types. The bottom plot shows the interference in orange and the actions taken in blue. We can see that the radar only takes two actions indicating that it may be over-simplifying the actions.

4.1.4 DQN Performance based on Collisions

The DQN has been shown to be an improvement upon the MDP in two scenarios so far. The DQN allows the radar to make intelligent decisions when encountering an interference state it has never seen before where the MDP would merely take the default action. Additionally, the DQN is able to handle more memory states than the MDP due to the fact that the DQN does not require the large matrices necessary for the policy iteration.

One benefit to the DQN is that is has a faster convergence rate for simpler interference types than the MDP does. Where the MDP can take up to 1000 training runs to determine an optimal policy, the DQN will learn the optimal policy within 100 training runs for the constant and saw-tooth sweep interferers. Another benefit to the DQN is that it implements online learning and adapts to a change in interference relatively quickly. The MDP does not adapt to changing interference patterns well due to the fact that it acts on the total statistics that it has been trained on and not necessarily the most recent and relevant interference. Both of these benefits to the DQN can be seen in observing the collision rate during training of both the DQN and the MDP.

Figure 4.5 reveals how fast the DQN converges to an optimal policy, which we are defining here as completely avoiding the interference and producing no collisions. This is intuitive in the sense that the MDP requires many training runs to build up enough statistics and test all possible actions to accurately model the environment. The DQN only needs a few training runs (for this case) to identify the best action to take. In Figure 4.5 the red dotted line located at the 200th iteration indicates the time at which the constant interference switches from [1 1 0 0 0] to constant interference located at [0 0 1 1 1]. The x-axis indicates the training run number divided by 5 (a total of 2000 training runs). The y-axis indicates the total number of collisions the occurred during that training run. It can clearly be seen that the DQN immediately identifies the correct action to take in iteration 1 and

produces no collisions for the rest of the training. Once the interference changes bands, it again only takes 1 iteration for the DQN to learn the optimal action to take and avoid any collisions. The MDP on the right shows that at certain points the MDP is able to produce no collisions, however it takes much longer to converge to an optimal policy than the DQN does. Once the interference changes, the MDP takes even longer to converge to an optimal policy, or at least a policy that can consistently produce no collisions. This is due to the fact that the MDP must rebuild its statistics and relearn the environment.

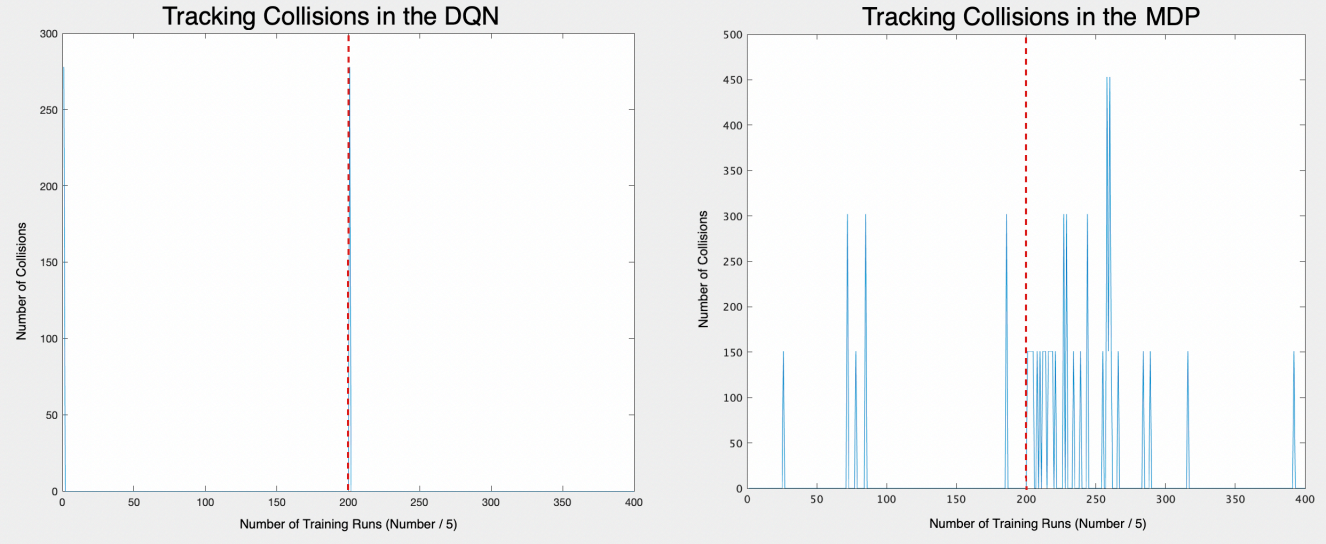

Figure 4.5: DQN vs MDP when the interference changes halfway through training for constant interference

From Figure 4.5 it was clear how the DQN is an improvement over the MDP in terms of colliding with other potential communication systems. The DQN is much faster at identifying the optimal policy to avoid the interference. It is also very adaptable and able to find the optimal policy once the interference has changed much faster than the MDP can. These interference types were very simple however, and it would be beneficial to see how the DQN and MDP compare for a more complex interference type. Figure 4.6 shows similar results for a more complex interference type. The interference in this figure is initially a saw-tooth

sweep pattern sweeping up through the sub bands:

- θ =[1 0 0 0 0]
- θ =[0 1 0 0 0]
- θ =[0 0 1 0 0]
- θ =[0 0 0 1 0]
- θ =[0 0 0 0 1]

This pattern repeats through all the training runs. The interference pattern then changes halfway through training to a saw-tooth sweep that is sweeping down:

- θ =[0 0 0 0 1]
- θ =[0 0 0 1 0]
- θ =[0 0 1 0 0]
- θ =[0 1 0 0 0]
- θ =[1 0 0 0 0]

From Figure 4.6 we can see that the DQN is able to learn the optimal policy in less than 20 iterations and does not have any collisions for the remaining training period. Once the interference pattern switches it takes less than 30 iterations for the DQN to find the optimal policy and produce 0 collisions. The MDP on the other hand does not seem converge in the 200 iterations during which it was trained. There are several instances where there

are 0 collisions however it never consistently remains at 0 collisions. Once the interference changes, the MDP never converges in the 200 iterations and in fact rarely gets a training run with 0 collisions. Comparatively, the DQN performs much better than the MDP. In fact, in the set training time it was able to find an optimal policy and produce no collisions for the majority of the training time.

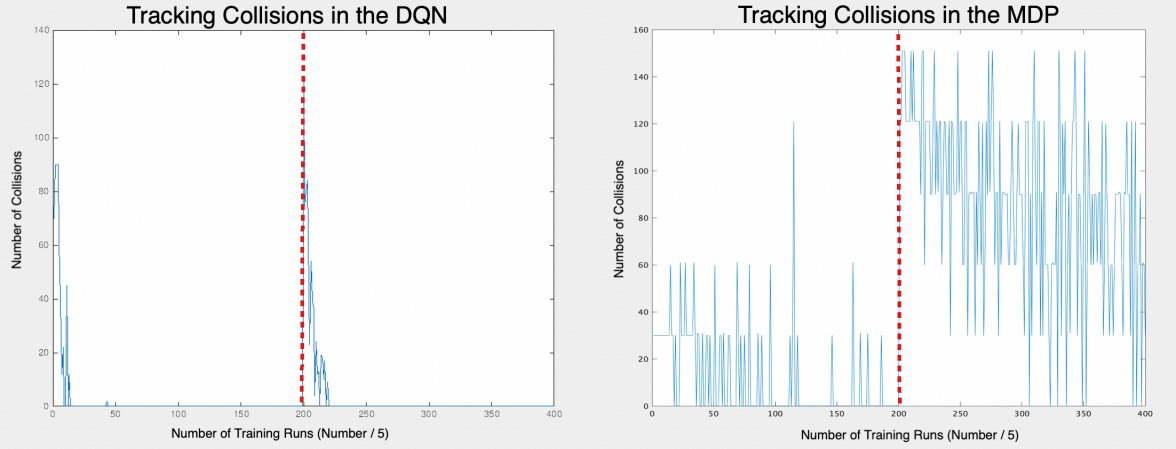

Figure 4.6: DQN vs MDP when the interference changes halfway through training for saw-tooth interference

The MDP is not able to quickly and efficiently adapt to changing RFI the way the DQN is able to. This is due to the fact that the MDP needs to build up new statistics on the new RFI in order to be able to accurately model the environment. Since the MDP also has the statistics on the old RFI, there needs to be some manual input to reset the training or it will take more time to find the optimal policy since the two RFI may be very different and require more training time so that the old statistics become irrelevant. In either case the DQN performs better and does not require any input to reset training. The DQN already implements online training and updates the optimal action per state accordingly.

The DQN inherently implements online learning due to how the neural network works. The weights of the neural network are updated every few iterations with stochastic gradient

descent by back-propagation. This is an improvement over the MDP which needs to run the full policy iteration algorithm each time it needs to update the policy (which favors offline learning more than online learning).

4.2 Discussion

The DQN has been shown to be an improvement upon the MDP method in simulation, however this may not hold true for the implementation. A strength of the DQN is that it requires fewer computer resources so it should be able to handle more memory states than the MDP. However, the implementation stresses timing over complexity, since the goal is to make the system able to react as fast as possible. The MDP is able to save the states and actions while running the learning algorithm on 400 saved states/actions at a time without any time lag in the system. It is unclear how a neural network will affect the timing of the system, both during training, and during estimation of Q values. Hence it may result in the DQN not being a viable application at all if it isn't able to mesh well with the LabVIEW code. The obvious solution is to partition out a time to do offline training, and in that way it eliminates the need for the DQN to be able to learn in line with the rest of the SDR operation. While this method would work, the most optimal implementation would be getting online training to work so that the DQN can update it's Q-values in real time. This would emulate the results shown in simulation where the DQN is able to quickly adjust to changing interference types.

Additionally, the DQN is more difficult to predict since we are not able to see how it is being trained. We are simply inputting the states, actions and rewards and viewing the output of the neural network in order to determine its effectiveness. The MDP can be more easily evaluated since we can view the matrices as they are being built up and determine for

ourselves what the best action the MDP should be deciding to take. Therefore it is much easier to identify an error in the MDP rather than in the DQN, which is better for consumer troubleshooting and bug fixing. An example of this is shown in Figure 4.7. This figure tracks collisions that occur when the DQN is trained on a triangle sweep interference pattern. We see that the policy converges right around the 1000th iteration and produces consistently low collisions per training run after this point. The MDP is able to consistently produce 0 collisions once it has been trained and we can see that the DQN never reaches this point. The average collision per training run for the DQN once the policy has converged is 0.3401, where for the MDP it is 0. It is unclear why the DQN has a collision every three training runs when the other training runs produce no collisions. Perhaps altering the number of layers or the number of nodes per layer in the neural network will be the solution to this.

In more complex interference examples, the DQN has also been shown to over-simplify the result. This is actually quite common in Deep-Q Networks and implementing a Double DQN has been shown to improve performance and decrease this over-simplification [26]. Another possible solution to improve performance is to add memory into the neural network where the neural network receives the output of the previous iteration as an input. This is called a Deep Recurrent Q Network, where information is integrated through time and performance has been shown to increase [27].

Despite these limitations on the DQN, it has been shown that the DQN has a clear advantage over the MDP in terms of amount of memory able to be utilized, decisions based on unknown states, convergence of collision rates for learning interference behaviors and convergence of collision rates when interference is changed halfway through learning. LabVIEW has a similar interface with Python as it does with MATLAB so implementation of the DQN onto the SDR platform should be relatively easy.

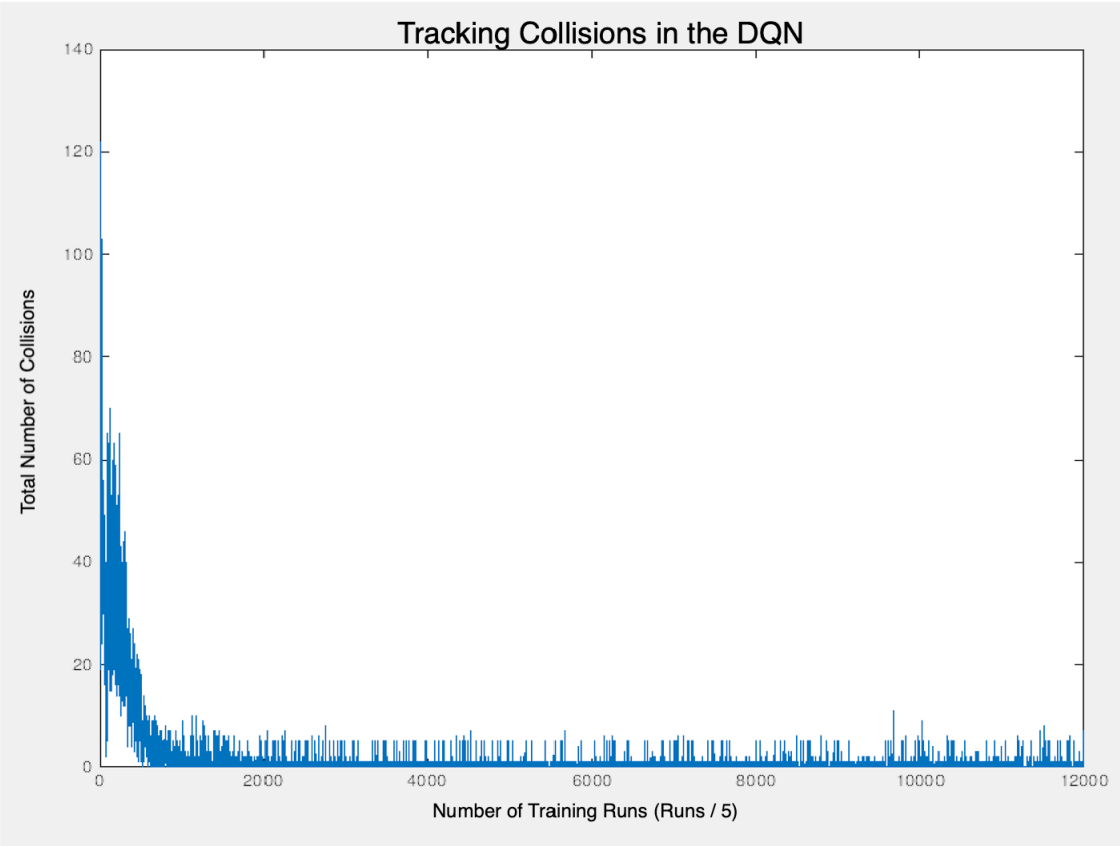

Figure 4.7: Tracking collisions for the DQN method when trained on a triangle sweep pattern.

Chapter 5

Conclusion

The application of Deep-Q Networks to cognitive radar is introduced in this work, substituting a neural network to the previously developed MDP model. The DQN is able to perform as well as the MDP, with the results being exactly the same for the majority of the training runs as we can see from Figure 4.1, and similar results were obtained for other interference types. The DQN behaved in a similar manner to the MDP when trained and evaluated on constant and intermittent interference types respectively. The DQN is also able to outperform the MDP in terms of extrapolation to new states not experienced before by the cognitive radar, as shown in Figure 4.2. The DQN is also able to find an optimal policy due to increased number of memory states where the MDP was not, as seen in Figure 4.3. The DQN has also been shown to have improved performance in terms of the speed at which it learns, and the speed at which it learns new interference types after being initially exposed to a different interference type. Overall the DQN has shown to be an improvement upon the MDP in simulation and will continue to be explored.

The MDP implementation on the Software Defined Radar has demonstrated improvement over the current interference avoidance algorithms. These algorithms being the Full Bandwidth mode where the radar simply operates in full bandwidth, and the Sense and React mode where the radar is able to sense the environment and avoid the interference in the next time step. The metrics used to compare these methods were SINR, Bandwidth, collisions rate, missed opportunity rate and probability of detection. The MDP implementation

was able to outperform the other methods with increased SINR, decreased collisions rate and increased probability of detection. While the MDP did not always show improvements for missed opportunity rate or Bandwidth, the behavior of the radar is easily controlled via the reward structure. This means that if the user decides to value a decreased missed opportunity rate and a higher Bandwidth more than the other metrics, then they can alter the reward structure to make the radar behave with those metrics being its primary goal to improve.

Future work will include implementing the DQN into the SDR system and observing its performance to measure against the MDP's performance. While the DQN has shown improvements in simulation, this may change drastically when implementing into a LabVIEW/Python interface. Additionally, the SDR code will be expanded to be able to use many different avoidance/predictive algorithms to directly compare to the MDP and the DQN. There will also be other metrics that can be used for evaluation such as ROC curves of the system as well as variance of the range value away from the true range of the target.

Additional future work will be to explore variations on the DQN method. Some of these methods are currently being explored such as the Deep Recurrent-Q Network which includes the previous output of the neural network in the input for the next iteration. In this way we add memory to the DQN and will hopefully see improvements in prediction. Another method currently being explored is the Double DQN. This method uses two separate neural networks, one for learning and the other for acting. In this way, we are able to decouple the selection and the evaluation, which will prevent over-estimation of Q values, a property found in the general DQN structure. The DQN will also be tested upon real RFI like the MDP has been, and its performance will be evaluated in comparison to the MDP. In addition to the different tests and types of DQN that will be done in the future work, the neural network parameters (number of layers and number of nodes per layer) should also be altered in order to see if there is a change in performance.

Bibliography

[1] A. Martone, "Cognitive Radar Demystified," *URSI Bulletin*, no. 350, pp. 10-22, 2014.

[2] A. Martone, K. Ranney, K. Sherbondy, K. Gallagher and S. Blunt, "Spectrum Allocation for Noncooperative Radar Coexistence," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, no. 1, pp. 90-105, 2018.

[3] S. Haykin, "Cognitive Radar, A way of the future," *IEEE Signal Processing Magazine*, vol. 23, pp. 30-40, Jan 2006.

[4] D. Hammerstrom, "Neural networks at work," *IEEE Spectrum Conference*, pp. 26-32, June 1993.

[5] S. Haykin, Cognitive Dynamic Systems: Perception-action Cycle, Radar, and Radio, Cambridge University Press, 2012.

[6] E. Selvi, Cognitive Radar Applied To Target Tracking Using Markov Decision Processes, Blacksburg, 2017.

[7] A. Kolobov, "Synthesis Lectures on Artificial Intelligence and Machine Learning," in Planning with Markov Decision Processes: An AI Perspective, 2012, pp. 1-210.

[8] M. Levorato, S. Firouzabadi and A. Goldsmith, "A Learning Framework for Cognitive Interference Networks with Partial and Noisy Observations," IEEE Transactions on Wireless Communications, vol. 11, no. 9, pp. 3101-3111, 2012.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *NIPS Deep Learning Workshop 2013*, 2013.

[10] T. Agarwal, "RADAR- Basics, Types Applications," Elprocus, [Online]. Available: https://www.elprocus.com/radar-basics-types-and-applications/. [Accessed 3 January 2018].

[11] M. Richards, J. Scheer and W. Holm, Principles of Modern Radar, Edison: Scitech Publishing, 2010.

[12] V. Cutsurdis, "Cognitive Models of the Perception-Action Cycle: A View from the Brain," in International Joint Conference on Neural Networks, 2013.

[13] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Upper Saddle River: Pearson Education, 2010.

[14] M. Greco, F. Gini, P. Stinco and K. Bell, Cognitive Radar: A Reality?, 2018.

[15] M. V. Otterlo and M. Wiering, "Reinforcement Learning and Markov Decision Process," in *The Logic of Adaptive Behavior*, IOS Press, 2008.

[16] E. Selvi, M. Buehrer, A. Martone and K. Sherbondy, "On The Use of Markov Decision Processes in Cognitive Radar: An Application to Target Tracking", in *2018 IEEE Radar Conference*, April 2018.

[17] "X300/X310," Ettus Research, 24 July 2018. [Online]. Available: https://kb.ettus.com/X300/X310.

[18] V. Schmidlin, G. Favier and R. Féraudt, "Multitarget Radar Tracking with Neural Networks," *IFAC Symposium on System Identification*, vol. 27, no. 8, pp. 621-626, 1994.

[19] C. Y. Kong, C. M. Hadzer and M. Y. Mashor, "Radar Tracking System Using Neural Networks *unpublished*," Perak.

[20] D.-J. Juang, K.-C. Hu, M.-L. Lee, C.-Y. Wang and Y.-N. Chung, "Extended neural network of radar tracking problems," in *TENCON 2007 - 2007 IEEE Region 10 Conference*, Taipei, 2007.

[21] J. Djugash and B. Hammer, "Neural Networks for Obstacle Avoidance *unpublished*," Pittsburgh.

[22] K. Arulkumaran, M. Deisenroth, M. Brundage and A. Nharath, "A Brief Survey of Deep Reinforcement Learning," *IEEE Signal Processing Magazine*, 28 September 2017.

[23] Bottou L. (2010) Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier Y., Saporta G. *Proceedings of COMPSTAT'2010*. Physica-Verlag HD

[24] M. Abadi, et al, "Tensorflow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/. [Accessed 2018].

[25] M. Zhou, "Reinforcement-learning-with-tensorflow," in *GitHub repository*, 2016. Available: https://github.com/MorvanZhou/ Reinforcement-learning-with-tensorflow, commit: 81fea33905c7f81719ec031eab51c68225eb7cce.

[26] H. van Hasselt, A. Guez and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Thirtieth AAAI Conference on Artificial Intelligence*.

[27] M. Hausknecht and P. Stone, "Deep Recurrent Q-Learning for Partially Observable MDPs," in *AAAI 2015 Fall Symposium*.

Appendices

Appendix A

Appendix

A.1 Policy Iteration Algorithm

This is the code used in conjunction with the Bellman equation and transition and reward matrices in order to find an optimal policy. While we were able to show that this process produced reasonable results in simulation, it's processing time was not able to keep up with the SDR data collection time, resulting in the transition to the DQN.

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      if $\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']$ **then do**
        $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

Figure A.1: The algorithm for policy iteration that was used in the MDP code

A.2 Graphical User Interface of the Program

This image shows the GUI of the program that allows the user to select various aspects that they may wish to change. These aspects include total run time of the radar, MDP parameters such as number of memory states, transmit power, how many pulses in a CPI, how long a PRI is etc. This program was built to be used by many different users in various environments so there is a significant level of adaptability built into the system.

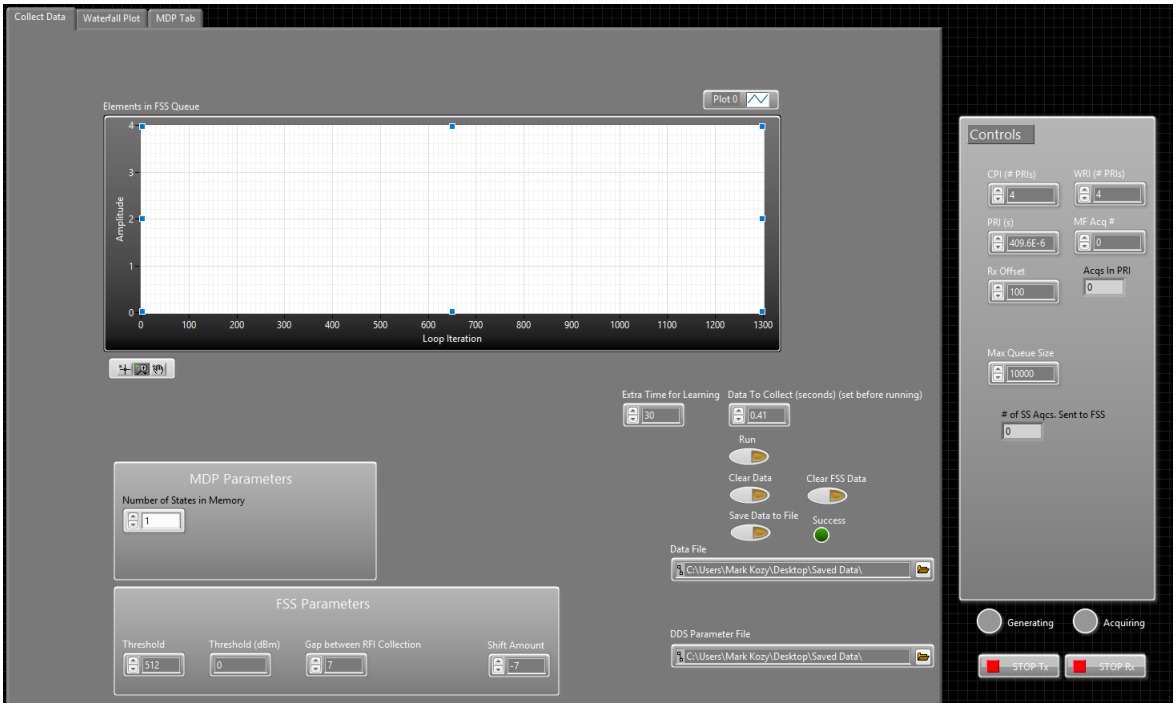

Figure A.2: The graphical user interface of the software defined radar

A.3 Example LabView code of the Machine Learning Loop

This is an example of the LabView code added to the previous version in order to add predictive capabilities to the system. This is shown so that users can get a feel for how this technique was implemented and either improve upon this method or even add their own predictive algorithm in its place. The DQN will also be implemented in a similar fashion in future work.

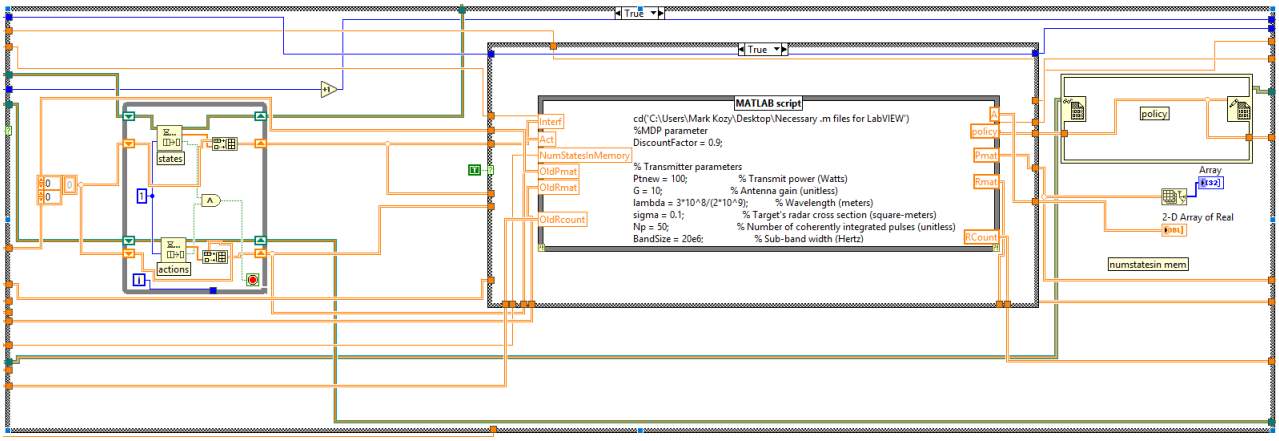

Figure A.3: The code that added the Machined Learning (MDP) to the simple data collection code