ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Triennale in Ingegneria Elettronica

TITOLO DELLA TESI

MACHINE LEARNING ALGORITHMS FOR DRONES COOPERATIVE NAVIGATION

Tesi in
Comunicazioni Digitali e Internet

Relatore:
Chiar.mo Prof. Ing.
DAVIDE DARDARI

Presentata da:
ANDREA PAGLIALUNGA

Correlatori:
Dott. Ing.
ANNA GUERRA

SESSIONE III
ANNO ACCADEMICO 2020-2021

Keywords

Autonomous Navigation and Indoor Mapping

Reinforcement Learning

Markov Decision Process

Q-learning

UAV - Unmanned Aerial Vehicles

5G	fifth generation
AWGN	additive white Gaussian noise
CR	cognitive radar
EIRP	effective radiated isotropic power
EKF	Extended Kalman-Filter
GPS	Global Positioning System
HPBW	half power beamwidth
KF	Kalman-Filter
LOS	line of sight
MDP	markov decision process
mmW	millimeter-wave
MSE	mean square error
PDF	probability density function
RCS	radar cross section
RL	reinforcement learning
RTT	round-trip time
RV	random variable
RX	receiver
SLAM	simultaneous localization and mapping
SNR	signal-to-noise ratio
THz	terahertz
TOA	time-of-arrival
TX	transmission

UAV	unmanned aerial vehicle
UAVs	unmanned aerial vehicles

Contents

Sommario

In questa tesi vengono analizzati argomenti legati alla capacità di un drone di poter navigare in maniera autonoma e rilevare un target in un ambiente sconosciuto mentre se ne stima una mappa di occupazione. Queste operazioni devono essere eseguite nel minor tempo possibile e in modo da poter minimizzare l'errore nella detection del target e nella ricostruzione della mappa. In seguito, l'analisi è stata estesa alla presenza di più droni e ad uno scenario in cui sono presenti più target.

Si è quindi studiato un algoritmo per permettere la mappatura di ambienti interni e l'identificazione di dispositivi di interesse. In particolare, si è utilizzato un approccio basato sull'apprendimento per rinforzo (reinforcement learning), specificatamente un algoritmo di Q-learning.

Questa tesi ha l'obiettivo, inizialmente, di descrivere i droni e la sensoristica a bordo necessaria per le operazioni di detection e mapping. Successivamente il problema di navigazione è stato formulato seguendo la filosofia dei processi di decisione Markoviana. Una possibile soluzione a questo tipo di problema è utilizzare algoritmi di apprendimento rinforzato (reinforcement learning) che sono stati descritti nel dettaglio e applicati al caso in esame.

Tramite simulazioni, si è verificata la capacità dell'algoritmo di risolvere il problema in esame, inoltre, si è potuto prendere visione dell'accuratezza nella stima dell'ambiente circostante e della capacità nell'identificazione delle traiettorie migliori dal punto di vista della detection dei target.

Abstract

This thesis analyzes topics related to the capacity of a drone to be able to navigate autonomously, and detect a target in an unknown environment while estimating an occupation map. These operations must be completed in the shortest time as possible and in such a way as to minimize the error in the detection of the target and in the reconstruction of the map. Subsequently, the analysis has been extended to the presence of multiple drones and a scenario in which there are multiple targets.

An algorithm is therefore studied to allow the mapping of an indoor environments and the identification of devices of interest. We use an approach based on a specific reinforcement learning algorithm, called Q-learning.

This thesis aims, initially, to describe the drones and the sensors on board necessary for detection and mapping operations. Subsequently, the navigation problem is formulated following the philosophy of the Markovian decision-making processes. A possible solution to this type of problem is to use reinforcement learning algorithms that are described in detail and applied to the case in question.

Through simulations, the ability of the algorithm to solve the problem under analysis is verified, and in addition, it is possible to view the accuracy in estimating the surrounding environment and the ability to identify the best trajectories, from the point of view of target detection.

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicle

An unmanned aerial vehicle (UAV), known as a drone, is an aircraft without any human pilot, crew or passengers on board, in fact, aerial vehicles are not guided but have the capability to guide themselves autonomously.

Most of the applications of UAVs require a network of UAVs that can achieve determinate tasks. This is accomplished using cooperative UAVs. The capacity of creating various shapes during navigation can help maximizing an information measure by having more informative measurements in each time instant.

Drones have many advantages over terrestrial sensors. Thanks to the continuous progress in the use of UAVs, they are becoming a very important and useful tool for detection and mapping. UAVs can obtain much more information in less time than terrestrial sensors. The technological systems installed on drones are capable of realizing an highly accurate mapping.

UAVs have a very high precision and are easy and fast to deploy. UAVs have also important disadvantages such as weather dependence, in fact, they are vulnerable to weather conditions and another concern is safety and privacy.

One of the main tasks that an UAV can accomplish is the localization of targets while mapping an unknown environment. This is helpful in many fields, in fact, UAVs can be used for remote surveillance, logistics, emergency situations and for many other functionalities. For example, UAVs are widely used by the police and firefighters to investigate site of deadly explosions, to inspect burned-out buildings, to penetrate dangerous environment, to detect people that need to be rescued. Usually UAVs are used in post-disaster situations because in these situations is common that the terrestrial localization systems have collapsed. Using UAVs, we can obtain a different point of view of the environment and also have a biggest range of visibility that in dangerous situations is impossible to have from the human operators. UAVs can be quickly deployed over disaster zones, operators are using them to produce maps, track the victims, and to understand the seriousness of the situation.

To accomplish what we have described above, we have to take into consideration the problem of localization and the problem of the radar sensor.

1.1.1 Localization Technologies

Starting from the invention of the telegraph to the next 6G cellular systems, the technological development has always brought positive feedback in our society. Thanks to all these improvements we brought a change in how we see the world and in how we want it to be in the future. The world we want to live in is a world that ensures security and connections and thanks to the great minds that lived before us and with us, we have devices that assure that. In fact, through these devices we can communicate with everyone all around the world and help our lives to be more secure, starting from the simplest thing such as reminding us of thing we need to do, to the most important for us, such as monitoring our home when we are out, or monitor our bank account. These devices offer services that depend on the needs of the user. Many services require the use of the position both in outdoor and indoor environment.

In outdoor environments, the users positions are usually retrieved thanks to GPS, while, in indoors, it is difficult to access the GPS signals and, thus, ad-hoc positioning systems should be used to infer users' position. Because of that, the need to dive into this interesting topic, that is indoor localization, has arisen.

We have the possibility to use different technologies for indoor localization. For localization, we could use communication technologies such as WI-

FI. Using the WI-FI received signal strength we can estimate the smartphone user's location and movement [1].

An important indoor localization technology for accurate indoor mapping is the ultra-wide bandwidth (UWB) technology.

The UWB technology permits to obtain a sub-meter localization accuracy. This technology is used because of its many advantages [2] [3]: provides high localization precision based on time-of-arrival estimation of the received signal, presents a good resistance to jamming, has an easier target information recovery from reflected signals, is capable of working in harsh communication channels with low SNRs and is a cheaper and simpler solution compared to other technologies.

Another technology used in indoor localization is the received signal strength (RSS) based fingerprinting method. The RSS fingerprinting based localization systems are implemented to determine user location by measuring electromagnetic radiations sent from different access points (APs) [4].

1.1.2 Enabling Radar Technologies for Target Detection and Indoor Mapping

Radar are detection systems that transmit electromagnetic wave signals that objects reflect. By capturing the reflected signal, a radar system can determine various parameters such distance, velocity and angle of a target.

Low complexity radars can be installed in UAVs to increase accuracy.

This section illustrates the possible technologies that can be mounted on UAVs and that can help the agent/UAV to accomplish the two tasks of target detection and mapping. We review some radar technologies and then we will focus on THz radars as they can achieve higher accuracies in map reconstruction. Section 1.2.1 describes the idea of Cognitive Radar, section 1.2.2 illustrates the functioning of Frequency Modulated Continuous Wave radar, while section 1.2.3 is dedicated to Millimeter wave radar and THz radar.

Cognitive radar

Radars have established themselves as an indispensable tool for detection and tracking of targets using radio waves. A cognitive radar is an intelligent, dynamic system that is aware of its outside world, exploits prior knowledge, and learns through interaction with the environment. A cognitive system can produce more information and it can be faster than a human operator could [5].

A cognitive system uses the signal it receives to collect information about the environment, combines these observations with prior knowledge and then learn from it. A cognitive radar can adapt the radar sensors, in response to a change of the environment, to meet the needs of the mission, taking into consideration the goal of the operation.

Three concepts are basic to the constitution of cognitive radar: 1) intelligent signal processing (learning through interactions of the radar with the surrounding environment); 2) feedback from the receiver to the transmitter; and 3) preservation of the information content of radar returns, which is realized by the Bayesian approach to target detection through tracking [6].

A cognitive radar uses certain components to get information about the environment. The components are: radar transmitter, radar receiver and environmental sensors. The radar transmitter produces pulses of energy that are radiated into space by the antenna to interrogate the environment, the sensors perceive the echoes of the sent signals, so they give more information about the surroundings and the radar receiver elaborates the data acquired by the sensors.

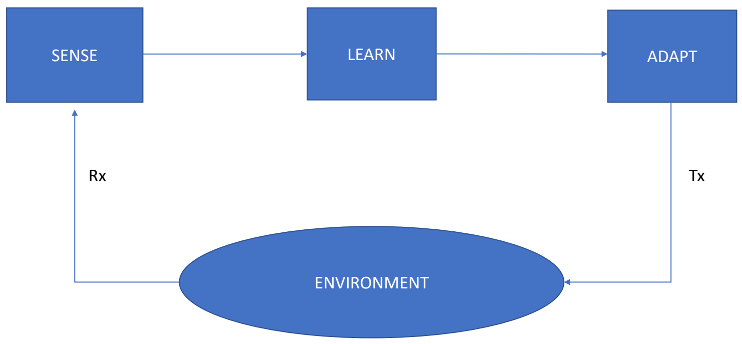

Figure 1.1 represent the functioning of a cognitive radar. Cognitive radars are systems that sense the environment, learn from it relevant information about the target and the background, then adapt the radar sensor to optimally satisfy the needs of their mission according to a desired goal.

The learning procedure best suited for cognitive radar is reinforcement learning, because the CR needs a well-known environmental model due to the fact that the cognitive radar moves in a fast-changing environment.

In a Cognitive radar network, the system is formed by several radars working together in a cooperative manner with the goal of improving what the radar components can achieve individually.

There are two types of cognitive radar networks:

1) Distributed cognition: all the radars are cognitive. This is the best option, because the delay that is found in centralized cognition is avoided.

2) Centralized cognition: only the central base station is cognition based.

Along with artificial intelligence and machine learning cognitive radar is a very active area of research. One example of study in the field of cognitive radar is the investigation of ways to maximize the radar performances in congested radio frequency environment. Wider bandwidth gives us better range resolution and more spectrum to operate in, so that the environment is better pictured and also, if the radar finds an interference can switch quickly to another portion of the spectrum. This is called RF interference avoidance.

Frequency Modulated Continuous Wave Radar

A continuous wave is an electromagnetic wave of constant amplitude and frequency that is considered to be of infinite duration. Frequency modulation is one of the transmission techniques used to transmit information using the carrier's frequency variations. Continuous wave radar devices without frequency modulation cannot determine target range because they lack the timing mark that is necessary to allow the system to time accurately the transmission and reception and to convert this into range [7] [8].

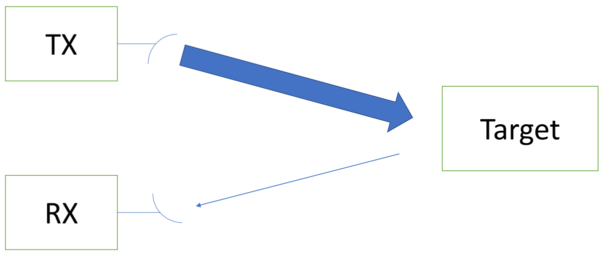

Figure 1.1: Transmission and reception of a target source

The ability to measure both speed and range of the target is an important theme. Continuous wave radars have some limitations: they can measure only the speed of the target and they are not able to measure the distance between radar and target. So, to avoid these problems, FMCW radars are introduced. FMCW radar is a special type of radar that transmits continuous wave signal whose transmitting frequency is modulated by a specific signal, so it changes its operating frequency during measurements. FMCW radars interrogate the environment with a signal linearly modulated in frequency.

The FMCW radar can measure the speed of the target as well as the distance of the target from the radar. The distance measurement is found by comparing the frequency of the received signal to the transmitted signal. The distance is proportional to the frequency difference between the two.

An FMCW radar consists essentially of the transceiver and a control unit with a microprocessor. The transceiver is a compact module, and usually includes the patch antenna implemented as separate transmit and receive antenna. The high frequency is generated by a voltage controlled oscillator which directly feeds the transmitting antenna. If using a single antenna there is the need to separate the transmitting and receiving signal, and this can be done using a duplexer. The duplexer is an electronic switch. Having two separate antennas, one for transmission and one for reception, is much cheaper. On a common substrate, the transmitting antenna array and the receiving antenna array, are placed directly above each other with polarization direction rotated by 180° against each other.

In general, we have that frequency modulation can be summarized and represented as in Figure 1.2:

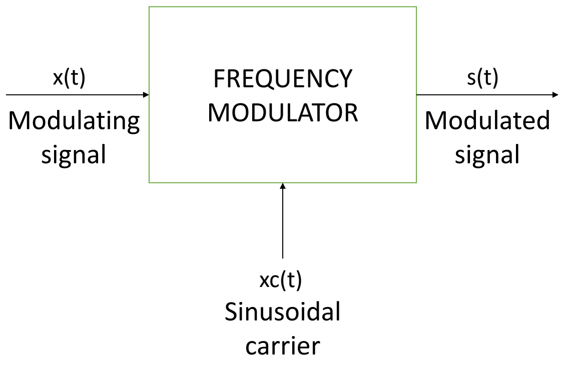

Figure 1.2: Frequency modulation

The information to be transmitted is $x(t)$ and the sinusoidal carrier is $x_c(t) = V_0 cos(2\pi f_0 t + \phi_0)$, where f_0 is the carrier's frequency, V_0 is the carrier's amplitude and $(2\pi f_0 t + \phi_0)$ is the carrier's phase.

The modulator combines the carrier wave with the modulating signal to get the modulated signal. So the modulator perturbs the carrier's parameters, such as frequency and amplitude, based on the temporal evolution of the modulating signal $x(t)$.

An important parameter in frequency modulation is k_f that is the modulation sensitivity, that is fundamental in the modulation law $\Delta f(t) = k_f x(t)$, that identifies the fact that the instantaneous frequency deviation is proportional to the modulating signal.

The signal's swing increases when $x(t)$ is greater than zero while when $x(t)$ is less than zero the signal's swing decreases. So, in FMCW radar, a signal is transmitted, which increases or decreases in frequency periodically.

We can represent the modulated signal as:

$$s(t) = V_0 cos(\phi(t))$$

where:

$$\phi(t) = \alpha(t) + 2\pi f_0 t + \phi_0$$

$$\alpha(t) = 2\pi \int_0^t \Delta f(\tau) d\tau$$

and

$$\Delta f(t) = k_f x(t)$$

Substituting the equation representing $\alpha(t)$ in the equation representing $\phi(t)$, we obtain:

$$\phi(t) = 2\pi \int_0^t \Delta f(\tau) d\tau + 2\pi f_0 t + \phi_0$$

so the resulting modulated signal is:

$$s(t) = V_0 cos \left(2\pi f_0 t + \phi_0 + 2\pi k_f \int_{-\infty}^t x(\tau) d\tau \right)$$

There are several possible modulation patterns which can be utilized. One of the most used is the sawtooth modulation. If $x(t)$ is represented by a sawtooth signal, we have that a delay will shift the echo signal in time. This results in a frequency difference between the original signal and the delayed echo signal, that is a measure of the distance of the reflecting object. So, the received waveform is simply a delayed copy of the transmitted waveform.

The distance R to the reflecting object can be determined by the following relations:

$$R = \frac{c_0 \Delta t}{2} = \frac{c_0 \Delta f}{2S}$$

where c_0 is the speed of light, Δt is the delay time, Δf is the measured frequency difference, R is the distance between antenna and the reflecting object, S is the frequency shift per unit of time and f_D is the Doppler frequency caused by the speed.

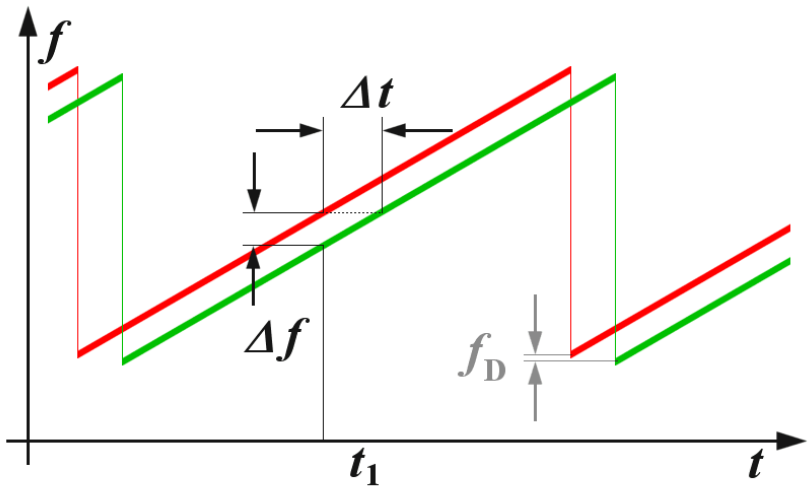

Figure 1.3: FMCW system with sawtooth chirp modulation

Millimeter wave radar and TeraHertz radars

Radar systems are used to provide an accurate image of the environment and their performance depends on the frequency at which they operate.

The millimeter waves (mmW) band of frequencies extends from 30 GHz to 300 GHz. Millimeter Wave radars transmit signals with a wavelength that is in the millimeter range. Thanks to the short wavelength, the size of system components can be miniaturized. The terahertz (THz) band extends from 300 GHz to 30 THz [9]. Terahertz waves, which have higher frequencies and shorter wavelengths than millimeter waves, allow the construction of radar systems with a smaller footprint and higher resolution. THz waves allows us to access environment that are visually inaccessible to us or with camera systems, because some optical opaque materials are more transparent to terahertz frequencies. Terahertz radiation can penetrate fabrics and plastic materials, so this type of radiation can be used also for security purpose [10]. THz radiation and millimeter waves are used in many other fields such as physics, archaeology, medicine and chemistry.

It is useful to describe also microwave radars. Microwave is a form of electromagnetic radiation with frequencies between 300 MHz and 30 GHz respectively. Microwave radar technology is used to image the terrain, ocean, space and to find the presence and to identify object, in fact the short wavelength of microwaves causes large reflections from objects the size of motor vehicles, ships and aircraft.

1.2 Case of Study

The main purpose of this thesis is to illustrate how, using machine learning algorithms, we can solve a UAV navigation problem to improve the mapping of an indoor environment and the detection of a target of interest. In fact, it will be described an algorithm that allows an unmanned aerial vehicle to navigate an unknown indoor environment and minimize the error in detecting a target.

The problem is formulated as a Markov decision process (MDP) where UAVs run a reinforcement learning algorithm (RL), specifically a Q-learning algorithm, and are equipped with a THz radar that can scan the environment.

A visual example of the problem that we are going to face and that is going to be studied in the next chapter is:

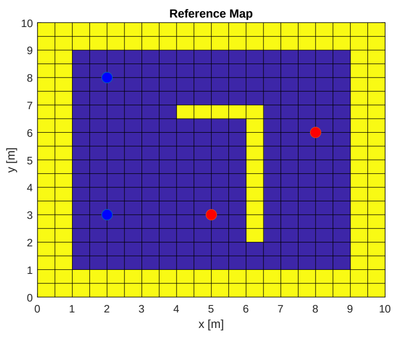

Figure 1.4: Representation of the environment studied

Chapter 2

Estimation Techniques by UAV Networks

The problem investigated is a scenario where a static target is present in the environment and a UAV has to detect the target during its navigation. The map of the environment is estimated using energy measurements collected by the radar. The problem is formulated as a Markov decision process where the UAV is equipped with a THz radar capable of scanning the environment with high precision. This problem can be solved using RL.

In this thesis, we consider that UAVs are designed to accomplish two main tasks that are: (1) tracking/detecting a target (2) mapping an unknown indoor environment. If a network of UAVs is considered, information sharing between UAVs is possible. In these networks, data are exchanged with neighbors in order to improve the environmental awareness.

When in flight, a UAV measures its position thanks to GNSS/INS, sends it to its neighbours and receives the positions of the other UAVs via multi-hops. A network of UAVs interrogates the environment via radar signals and by the output they can detect the presence of a target and understand important parameters of the target such as position-related and velocity information. Then, acquisitions are exchanged with the neighbours. Two important parameters coming from radar measurements are ranging and bearing, where ranging represents the target's distance from the radar, while bearing represents the horizontal angle.

In our case of study, a static target is present in the environment. The goal of the algorithm, is to estimate the target presence/absence in the environment, thanks to the measurement acquired from a single drone in navigation. At each time instant, the first step is to obtain state-related information from radar measurements and the second step is to infer the best action to be taken in order to achieve the final goal. The agent can be depicted as a block where two estimation processes take place. The first is the state estimator that can be implemented using a Bayesian filtering approach and provides an estimate of the state s_k, the second step is the policy estimator which infers the best action to be taken to maximize the expected return.

In section 2.1 will be described the Bayes estimation and as a subsection, the Kalman filter, in section 2.2 will be described how to implement target detection and mapping of an environment and in section 2.3 will be described our case of study.

2.1 Bayesian Estimation

In Bayesian estimation, θ is a random variable that has an unknown distribution. Bayesian estimation uses both data and prior information. After acquiring the data in the present time, we combine them with the prior probability density function to create the posterior distribution. Before defining the posterior distribution we need to define the likelihood function.

Let θ be an unknown parameter and $x_1, ..., x_n$ be a random sample from the probability density function $f(x|\theta)$ where $x_1, ..., x_n$ are independent and identically distributed. The likelihood function is a function of the parameter given the data that we have observed:

$$L(x_1, ..., x_n|\theta) \tag{2.1}$$

We hold constant the data from x_1 to x_n and we try different values of θ to see which likelihood we find. The higher the likelihood, the more likely that parameter value is, given the data that we have observed. So, the likelihood function is the joint probability density function (PDF) of our random samples conditioned on parameter theta:

$$L(x_1, ..., x_n|\theta) = f(x_1, ...x_n|\theta) \tag{2.2}$$

Since the samples are independent then we can split the PDF into the product of marginals, i.e., univariate, that means the function depends on only one random variable:

$$L(x_1, ..., x_n|\theta) = f(x_1|\theta)...f(x_n|\theta) \tag{2.3}$$

We can rewrite the equation above, and we obtain:

$$L(x_1, ..., x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta) \tag{2.4}$$

where $f(x_i|\theta)$ represent the PDF given θ using different values of x.

To find the most likely value of θ, that is called the maximum likelihood estimator because it maximizes the likelihood function, we take derivatives.

A very important likelihood function is the log-likelihood function:

$$l(x_1, ..., x_n|\theta) = logL(x_1, ..., x_n|\theta) \tag{2.5}$$

We use the log-likelihood function because the logarithm is a monotone function, so, if we want to maximize the log-likelihood is the same thing as maximizing the likelihood function.

So, to find the posterior distribution of theta, i.e., the PDF of parameter theta given the measurements $x_1, ..., x_n$, we need the prior PDF and the likelihood function $L(x_1, ..., x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$.

The posterior distribution can be written as:

$$g(\theta|x_1, ..., x_n) \propto p(\theta)L(x|\theta) = p(\theta)\prod_{i=1}^{n} f(x_i|\theta) \tag{2.6}$$

The posterior distribution is proportional to the prior distribution multiplied by the likelihood function:

$$g(\theta|x_1, ..., x_n) \propto p(\theta)\prod_{i=1}^{n} f(x_i|\theta) \tag{2.7}$$

Bayesian estimation is all about combining the prior information with the obtained data to learn something about the unknown parameter.

A Bayes estimator is an estimator that minimizes the posterior expected value of a loss function, equivalently, it maximizes the posterior expectation of a utility function.

Let $\hat{\theta}$ be an estimator for θ based on some statistic, then the loss function associated with $\hat{\theta}$ is $L(\hat{\theta}, \theta)$ where $L(\hat{\theta}, \theta) >= 0$ and $L(\theta, \theta) = 0$.

Two common loss functions are:

$$L(\hat{\theta}, \theta) = (\hat{\theta} - \theta)^2 \tag{2.8}$$

and

$$L(\hat{\theta}, \theta) = |\hat{\theta} - \theta| \tag{2.9}$$

If the loss function associated with $\hat{\theta}$ is $|\hat{\theta} - \theta|$ then the Bayes estimator is the median of the posterior distribution (commonly called maximum a posteriori (MAP) estimator) while if the loss function associated with $\hat{\theta}$ is $(\hat{\theta} - \theta)^2$ then the Bayes estimator is the posterior mean (commonly called minimum mean square error estimator - MMSE).

In the case where the purpose is to estimate a sequence of parameters representing, for example, the time evolution of a single parameter (typically referred to as "state"), iterative approaches such as Bayesian filters can be adopted. The Bayes filter is a technique for recursive state estimation. It is used to find the current state of a system given current and past observations. We take our belief at time instant k (i.e., the posterior distribution of the state at time k), and we advance it to the next state $k + 1$, so we go from time step k to time step $k + 1$ using the most recent observations, as well as the knowledge acquired from the previous ones.

There are different realizations of the Bayes filter, for example the Kalman filter and the extended Kalman filter [11]. These are realizations of a recursive Bayes filter, so they follow the same equations and structure to perform the state estimation.

2.1.1 The Kalman Filter

The Kalman filter (KF) is a particular implementation of a Bayesian filter.

The Kalman filter is the optimal state estimator given some knowledge about the types of disturbances and measurement noise. We assume that both the disturbance and the noise are a Gaussian white noise process. In fact, the Kalman filter assumes that all the PDFs involved in the filter are Gaussian and all models are linear. If we are in a non-linear world, we cannot use the Kalman filter, but we have to use the extended Kalman filter (EKF). The extended Kalman filter is a variant of the Kalman filter that deals with non linearities. It performs a local linearization and turns the non-linear model into a linear model.

We adopt an extended Kalman filter to compute the Gaussian belief of the state. This algorithm is a two-step process: the first step consist in determining a prediction about the system's state, so it takes into account the steering information in order to predict the next point in time; instead the second step is the correction step and uses the noise measurements and the sensor observation to refine the system status estimate.

At each time instant, we perform a prediction step and a correction step and then the new observation comes in. So, we have a recursive procedure which always updates the previous belief and turns it into the next belief.

2.2 Target Detection and Mapping

The UAVs are equipped with a receiver able to receive and process the signal coming from an active target that transmits at frequency f_t, and with a radar capable of interrogating the environment and operating at a frequency f_r. At each time instant k, the UAV performs a scan of the environment to acquire as much information as possible.

Considering a maximum time to complete the mission, we want to minimize the uncertainty in estimating the map of an unknown environment. We consider a grid representation of the environment where there are N_{cell} and these N_{cell} can have two states: occupied or free. In addiction, we consider that the environment does not change with time.

The state $\mathbf{s}_k$ describes the system and it consists of: the UAV position,

$s_{U,k}$, a parameter t indicating the presence or not of a target and the state of the cell, $\mathbf{m}$. So the state can be defined as:

$$s_k = [\mathbf{s}_{\mathrm{U},k}, t, m_1, ..., m_{N_{\mathrm{cell}}}] = [\mathbf{s}_{\mathrm{U},k}, t, \mathbf{m}] \tag{2.10}$$

where $\mathtt{t}$ indicates the presence or not of the target, $\mathbf{m}$ is the true map and m_i represents the state of the ith cell of the map.

2.2.1 Mapping using radar measurements

The mapping is performed by using energy measurements acquired by the radar during the interrogation of the environment. The signal received by the radar is given by:

$$r(t, \theta_b) = \sum_{n=0}^{N_p-1} x(t - nT_f, \theta_b) + n(t) \tag{2.11}$$

where $x(t, \theta_b)$ is the signal acquired when pointing at direction θ_b, $n(t)$ is the addictive white Gaussian noise (AWGN) and N_p are the transmitted pulses sent from the pulse-based radar. A pulse-based radar is a radar system that determines obstacle's parameters using pulse-timing techniques [12].

The received signal is passed through a filter to eliminate the out of band noise, so is generated a filtered version of the signal called $y(t, \theta_b)$.

The final energy value is given by:

$$e_{bs} = \sum_{n=0}^{N_p-1} \int_{(s-1)T_{ED}}^{sT_{ED}} y^2(t + nT_f, \theta_b) dt \tag{2.12}$$

where T_f is a time frame and is divided into N_{bins}. The temporal bin index is indicated as $s = 1, ..., N_{bins}$.

The observation vector can be written as:

$$e_k = [e_{(11,k)}, ..., e_{(N_{tot}N_{bins},k)}]^T \sim \mathcal{N}(\mu_{e_k}, \Sigma_{e_k}) \tag{2.13}$$

The equation represents the energy measurements observed at time instant k with b being the steering index and N_{tot} the number of steering directions and s being the bin index. .

The mean vector is μ_{e_k} and the covariance matrix is Σ_{e_k}. We can calculate the variance and the expectation of $e_{bs,k}$ [12]:

$$\mathbb{E}[e_{bs,k}] = N_p \int_{(s-1)T_{ED}}^{sT_{ED}} \tilde{x}^2(t,\theta_b)dt + \sigma^2 N_p T_{ED} = E_{bs,k} + E_n \tag{2.14}$$

$$var(e_{bs,k}) = \sigma^2_{bs,k} = N_0(2E_{bs,k} + E_n) \tag{2.15}$$

where $E_{bs,k}(m)$ is equal to:

$$E_{bs,k}(\mathbf{m}) = \sum_{l\in I(s)} \int_W \frac{L_0(f)\rho_l}{d^4_{ik}} G^2(\tilde{\theta}_l, f)df \tag{2.16}$$

where $\tilde{x}(t)$ is the filtered version of $x(t)$, $\tilde{\theta}_l = \theta_l - \theta_b$ is the difference between the arrival and steering angles, $I(s)$ is the set of cells located at the same discrete distance s from the radar, $L_0(f)$ is the path-loss at the reference distance of 1 meter, $G(\theta)$ is the gain and ρ_l is the radar cross section of the lth cell, that is the equivalent area seen by the radar.

To infer the map of the environment, the UAV needs to search for the maximum posterior probability:

$$\hat{\mathbf{m}}_k = arg\max_{\mathbf{m}} b_k(\mathbf{m}) = arg\max_{\mathbf{m}} p(\mathbf{m}|e_{1:k})) \tag{2.17}$$

where $b_k(\mathbf{m}) = p(\mathbf{m}|e_{1:k})$ is the posterior probability distribution of the map given the observations collected until the instant k. The mapping problem is described as a maximum posterior estimation problem. We operate cell-by-cell as:

$$\hat{m}_{i,k} = arg\max_{m_i} b_k(m_i) = arg\max_{m_i} p(m_i|e_{1:k})) \tag{2.18}$$

$$= arg\max_{m_i} \frac{p(e_k|m_i)b_{k-1}(m_i)}{p(e_k|e_{k-1})} \tag{2.19}$$

m_i can have two possible values: $m_{i,1} = (m_i = 1)$ indicate the event for the ith cell of being occupied and $m_{i,0} = (m_i = 0)$ indicate the event for the ith cell of being free.

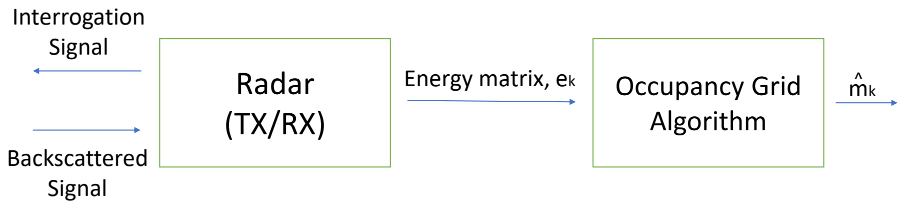

Figure 2.1: Block representation of the mapping process

2.2.2 Target detection

The target detection problem is analyzed taking into account a fixed maximum time to complete the mission. We want to minimize the error in detecting a target so we need to minimize the mis-detection and false alarm.

After filtering, the received signal $r(t)$, that is a signal referring to a receiver that has the objective to sense signals emitted by a target, is sampled at the Nyquist rate, obtaining the vector $\mathbf{y} = [\mathbf{y[1]},\ldots, \mathbf{y[N]}]$ where $N = 2TW$ is the number of samples, T is the observation time window and $\hat{t}_k$ denotes the target's presence estimate.

The normalized energy test statistic can be written as:

$$\frac{2}{N_{0_v}} \int_0^T [r(t)]^2 dt = \frac{1}{\sigma_v^2} \sum_{n=0}^{N-1} |y[n]|^2 \tag{2.20}$$

where $\sigma_v^2 = N_{0_v} W$ represent the noise power, N_{0_v} is the one-sided noise power spectral density of the receiver of the detector module and W is the bandwidth.

The related discrete time detection problem can be written as:

$$H_0 : y[n] = v[n] \tag{2.21}$$

$$H_1 : y[n] = x[n] + v[n] \tag{2.22}$$

where H_0 represents the absence of the target and H_1 represents the presence of the target while $x[n]$ and $v[n]$ are the nth samples of the signal and noise component with $v[n] \sim \mathbb{N}(0, \sigma_v^2)$.

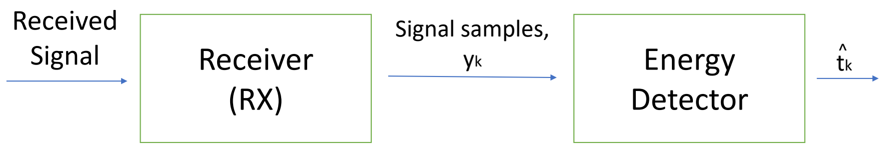

Figure 2.2: Block representation of the detection process

Chapter 3

Reinforcement Learning

Reinforcement Learning (RL) is an aspect of Machine Learning where an agent learns how to behave in an environment, by performing certain actions and observing the rewards that gets from those actions. The agent is the learner in RL. The actions are the choices made by the agent, the states are the positions of the agent and the basis for making the choices and the rewards are the basis for evaluating the choices.

Reinforcement learning is about learning from interaction how to behave to achieve a goal. The reinforcement learning agent and its environment interact over a sequence of discrete time steps. In reinforcement learning, the goal of the agent is defined by the reward, provided by the environment to the agent. The agent's goal is to maximize the total amount of reward it receives, this means maximizing cumulative reward. In reinforcement learning we have to explore and to exploit. We need to explore to find more information about the environment, but we need also to exploit the already known information to maximize the expected reward.

There are many approaches to Reinforcement Learning. The most important are:

- Policy based approach.
- Value based approach.

In policy-based reinforcement learning, we have a policy which we need to optimize. A policy can be deterministic or stochastic. A deterministic policy is a policy that at a given state s will always return the same action a, while a stochastic policy gives a distribution of probability over different actions.

In value based reinforcement learning the goal of the agent is to optimize the value function $V(s)$ which is defined as a function that tells the maximum expected future reward of the agent at each state.

In our case of study we used a policy based approach, in fact, the aim is learning a control policy for maximizing a numerical reward signal. According to this principle, an agent, determines which actions brings to the largest expected return while taking into account the immediate reward and future rewards. In this way, the agent is capable to learn a policy while achieving its goal.

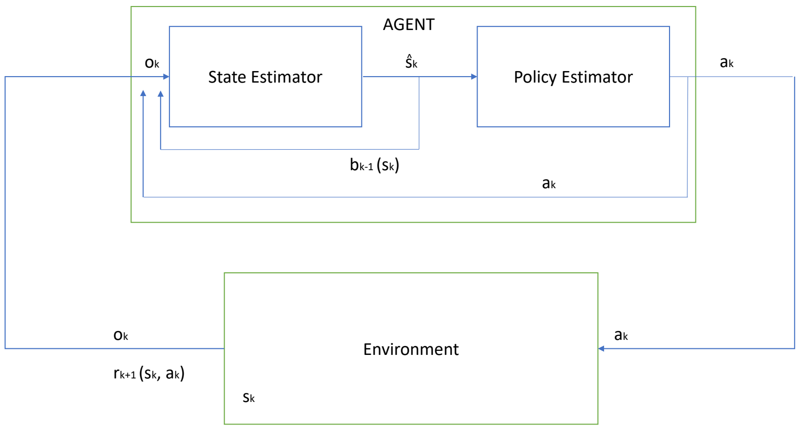

Figure 3.1: Representation of the interactions between agent and environment

In particular, the current observations are indicated with e_k (o_k in figure 3.1), $r_{k+1}(s_k, a_k)$ identifies the reward at time instant $k+1$ and $b_{k-1}(s_k)$ is the previous posterior distribution.

The agent is seen as a block where two processes take place: the state estimator (implemented by using a Bayesian filtering approach) that provides an estimate $\hat{s}_k$ of the state s_k; the policy estimator which infers the best action a_k to be taken to maximize the expected return.

In the following sections, three important reinforcement learning algorithm will be described, that are: Temporal-difference Learning, SARSA and Q-learning algorithm.

3.1 Temporal-difference Learning

Temporal-difference (TD) learning is a combination of Monte Carlo methods and dynamic programming [13].

Dynamic programming wants to simplify a complicated problem by breaking it down into simpler sub-problems in a recursive manner. Monte Carlo methods require only experience (states, actions and rewards). In Monte Carlo methods the agent learns about the states and reward when it interacts with the environment so any prior knowledge of the environment's dynamic is not required. Thanks to this method, the agent can find an estimate of a state-value function or a action-value function. The Monte Carlo update can be expressed as:

$$V(S_k) = V(S_k) + \alpha[G_k - V(S_k)]$$

In the prediction step our goal is to learn a value function that estimates the total sum of the reward (returns) starting from a certain state:

$$v_\pi(s) = E_\pi[G_k|S_k = s]$$

where $G_k = r_{k+1} + \gamma r_{k+1}$.

Temporal difference learning combines experience with the Bellman equation. The Bellman equation defines that we can estimate the value of the current state based on the values of the states that we reach after choosing certain actions and moving into those states.

Temporal-difference methods use experience to predict what will come next, in fact, given some experience in using a certain policy π, TD methods update their estimate of the value function. At time $k+1$ temporal-difference methods make an update using the observed reward R_{k+1} and the estimate $V(S_{k+1})$.

The algorithm starts by initializing a table $V(s)$, then we evaluate the policy π and obtain a reward r and thanks to this we can update the value function for the old state. This is represented as:

$$V(S_k) = V(S_k) + \alpha[R_{k+1} + \gamma V(S_{k+1}) - V(S_k)]$$

The value $R_{k+1} + \gamma V(s_{k+1})$ represent the target for the TD.

The TD error is given by:

$$R_{k+1} + \gamma V(S_{k+1}) - V(S_k)$$

3.2 On-policy/Off-policy

The goal of reinforcement learning is to learn an optimal policy. We can have off-policy and on-policy reinforcement learning.

When a reinforcement learning agent is on a mission, it takes actions, learn which actions are good or bad and updates the Q-values. The policy that determines the action taken by the agent is called behaviour policy. This policy describes how the agent will behave in a certain state. The policy that allows the agent to learn from its actions and from the rewards it receives, is called the target policy.

An off-policy learner learns independently of the agent's actions. More in depth, this means that the target policy and the behaviour policy are different. Instead, in an on-policy learner the target policy and the behaviour policy are the same.

3.3 SARSA algorithm

SARSA is an on-policy iterative algorithm that balances exploration and exploitation with temporal difference learning. It is used in the reinforcement learning area of machine learning. This algorithm, whose acronym stands for "state-action-reward-state-action", can be used for learning a Markov Decision Process policy.

We want to evaluate the Q-function $q(s, a)$, because we are doing model-free reinforcement learning, where we do not use the transition probability distribution, so we do not know how states transition into next states. The main function for updating the Q-value depends on the current state of the agent S_k, the action the agent chooses A_k, the reward R_k the agent gets for choosing this action, the next state after the action S_{k+1} and the next action that the agent choose A_{k+1}. The update rule is given by:

$$Q(S_k, A_k) = Q(S_k, A_k) + \alpha[R_{k+1} + \gamma Q(S_{k+1}, A_{k+1}) - Q(S_k, A_k)]$$

A SARSA agent interacts with the environment and updates the policy based on actions taken. The Q-value for a state-action is updated by an error, adjusting by the learning rate α. Q-values represent the possible reward received in the next time step for taking action a in state s, plus the discounted future reward received from the next state-action observation.

A learning rate α of 0 will make the agent not learn anything while a factor of 1 would make the agent consider only the most recent information. The discount factor γ determines the importance of future rewards.

An extension of SARSA is the expected SARSA. The expected SARSA turns the SARSA algorithm from an on-policy to an off-policy algorithm by weighting the Q-value of each next state-action pair by the probability of taking that action giving the next state. It can be represented as:

$$Q(S_k, A_k) = Q(S_k, A_k) + \alpha[R_{k+1} + \gamma \sum_a \pi(a|S_{k+1}) Q(S_{k+1}, a) - Q(S_k, A_k)]$$

3.4 Q-learning

A very important reinforcement learning algorithm is Q-learning. Q-learning is a value based and off-policy learning algorithm. Value-based algorithms updates the value function based on an equation (Bellman Equation). The Q in Q-learning stands for quality or in other words, how useful a given action is in gaining future reward.

In chapter 4 we will discuss in depth the Q-learning algorithm and its fundamental aspects.

Chapter 4

Markov Decision Process

A Markov decision process consists in learning from interaction to achieve a goal and it is a representation of sequential decision making, where actions influence immediate and future reward.

The decision maker is called the agent and it interacts with the environment. The agent selects an action while the environment responds to the action. There are three important signals that are passed between the agent and the environment: the first signal represents the actions, indicated with A_k, that are the choices made by the agent; the second signal represents the states, indicated with S_k, that define the information through which the agent made the choices; the last signal represents the reward, indicated with R_k, that is used to define the agent's goal.

The action can be deterministic or stochastic. Deterministic actions mean that for each state and action a new state is specified, while stochastic actions mean that for each state and action a probability distribution over next states is specified. In our case of study, the actions are chosen thanks to a policy that is a probability distribution of selecting actions given the state the agent is in.

The agent and the environment interact at discrete time steps, $k = 0, 1, 2, 3, ...$ At each time step k, the agent receives a representation of the environment and chooses an action. At the next time step, the agent receives a reward R_{k+1} and it will be in a next state S_{k+1}, in general different from the previous one.

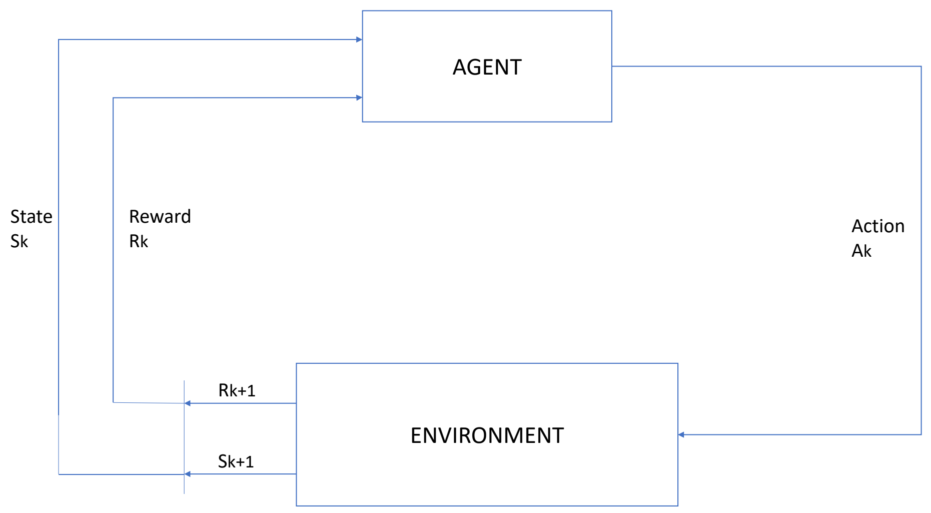

Figure 4.1: Functioning of a Markov Decision Process

A Markov process is a sequence of states with the Markov property. The Markov property means that the transition from state s_k to s_{k+1} is entirely independent of the past. This can be expressed as:

$$p[s_{k+1}|s_k] = p[s_{k+1}|s_1, ..., s_k]. \tag{4.1}$$

So, a MDP is defined by a tuple containing the state space $\mathcal{S}$, the action space $\mathcal{A}$, the reward space $\mathcal{R}$, and the probability of transitioning from one state s_k, at time instant k, to the state s_{k+1} at time $k+1$, defined as:

$$p(R_{k+1} = r_{k+1}, S_{k+1} = s_{k+1}|S_k = s_k, A_k = a_k). \tag{4.2}$$

The probability completely characterizes the environment's dynamics. It describes the possibility that the values s and r will occur at time instant $k+1$, knowing the value of the precedent action a and state s.

The actions are decided by the agent following a specific policy given by:

$$\pi(a_k|s_k) = p(A_k = a_k|S_k = s_k). \tag{4.3}$$

A policy function is a mapping from state space to the action space. The objective in a Markov decision process is finding a good policy, that means finding a function that defines the action that the agent will choose when finds itself in state s.

The optimal policy is chosen by selecting actions that maximize the Q-function (state-action value function), defined as:

$$Q_\pi(s_k, a_k) = E_\pi \left\{ \sum_{l=0}^{\infty} \gamma^l r_{k+l+1} | S_k = s_k, A_k = a_k \right\} \tag{4.4}$$

where γ is the discount rate or discount factor that has a value between 0 and 1.

The problem of finding an optimal policy can be stated as:

$$\pi^*(a_k, s_k) = \text{argmax}_{a_k} Q_\pi(s_k, a_k). \tag{4.5}$$

In the following, we will detail the definitions and derivations leading to (4.4) and (4.5).

4.1 Rewards and returns

Rewards are numerical values that the agent receives from the environment when performing an action. Rewards can be positive or negative based on the actions chosen by the agent. They allow the agent to learn from the environment and choose the best action to achieve the goal. The agent wants to maximize not the instantaneous reward, but the cumulative reward. So, if the agent needs to accomplish a certain action, we have to provide rewards so that in maximizing them the agent will also complete its mission. The sum of rewards the agent receives is called *return*.

$$G_k = r_{k+1} + r_{k+2} + r_{k+3} + ... + r_T \tag{4.6}$$

where T is the final time step.

4.2 Episodic and continuous tasks

Episodic tasks are the tasks that have a finite duration, so there is an end. They end in a special state called the terminal state, followed by a reset to an initial state. An example could be the game "Pacman". We start the game and play till we win or we lose and then the game will restart. This is called an episode and every episode is independent. While continuous tasks are tasks that do not have an end, that go on without limit. In this case the terminal state is infinite and not finite as in episodic tasks. In continuous tasks we could have infinite return, so to help us in this situation we define the discount factor.

4.3 Discount Factor

The Discount Factor helps us avoid infinity as a reward in continuous tasks. It has a value between 0 and 1 and it is usually called γ. If the value is 0, it means that the immediate reward has more importance so in this case we only consider immediate reward, while, if the value is 1, it means that we consider only future rewards. The equation for the discounted return G at time step t is:

$$G_k = r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + ... = \sum_{l=0}^{T} \gamma^l r_{k+l+1} \tag{4.7}$$

We can rewrite the equation above in a recursive way:

$$\begin{aligned} G_k &= r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + ... + \gamma^T r_T && (4.8) \\ &= r_{k+1} + \gamma(r_{k+2} + \gamma r_{k+3} + \gamma^2 r_{k+4} + ... + \gamma^{T-2} r_T) && (4.9) \\ &= r_{k+1} + \gamma(G_{k+1}) && (4.10) \end{aligned}$$

4.4 Policies and value functions

Value functions are functions that estimate how good it is for the agent to be in a given state. The definition of value function is linked to the definition of policy. The value function of a state s under a policy π, denoted as $v_\pi(s)$ is expressed as:

$$v_\pi(s) = E_\pi \left[\sum_{l=0}^{\infty} \gamma^l r_{k+l+1} | S_k = s \right] \tag{4.11}$$

A policy π is a probability distribution of selecting actions given the state we are in, so it defines what action to perform when we are in a certain state. A policy π is better than or equal to a policy π' if the expected return of π is greater or equal to the expected return of π' for all states. There is always an optimal policy. There could be more than one optimal policy, but all the optimal policies have the same state-value function, denoted as v_* and defined as:

$$v_*(s) = \max{}_{\pi} v_\pi(s) \tag{4.12}$$

4.5 Bellman Equation

The Bellman Equation helps us finding optimal policies and value functions. The agent's goal is to find a sequence of actions that will maximize the return, that is the sum of rewards during an episode.

The v-function or the state-value function, measures the goodness of each state. The value of $v_\pi(s)$ is:

$$v_\pi(s) = E_\pi[G_k|S_k = s] = E_\pi\left[\sum_{l=0}^{T} \gamma^l r_{k+l+1}|S_k = s\right] \tag{4.13}$$

that describes the expected value of the total return G, from state s at time instant k.

Now we can define the state-action value function, also called Q-function, as already mentioned. It defines the value of taking action a in state s following a policy π. The state-action value function, called $Q_\pi(s, a)$, is equal to:

$$Q_\pi(s,a) = E_\pi[G_k|S_k = s, A_k = a] = E_\pi\left[\sum_{l=0}^{T} \gamma^l r_{k+l+1}|S_k = s, A_k = a\right] \tag{4.14}$$

Knowing that $\pi(a|s)$ is the probability that a policy π selects an action a given a state s, we know that:

$$\sum_a \pi(a|s) = 1 \tag{4.15}$$

then we can write now the state-value function as:

$$v_\pi(s) = \sum_a \pi(a|s) Q_\pi(s|a) \tag{4.16}$$

The Bellman equation simplifies the computation of the value function decomposing it into different parts.

Defining p as the probability of action a, in state s, arriving at state s' with reward r, the Bellman equation for the state-value function is defined as [14]:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p^a_{ss'}(r(s,a) + \gamma v_\pi(s')) \tag{4.17}$$

where $p_{ss'}$ is the state transition probability.

The Bellman equation for the action-value function is:

$$Q_\pi(s,a) = r(s,a) + \gamma \max_{a'} Q_\pi(s',a') \tag{4.18}$$

that can be written as:

$$Q_\pi(s,a) = \sum_{s'} p^a_{ss'}(r(s,a) + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s',a')) \tag{4.19}$$

with an alternative form given by:

$$Q_\pi(s,a) = \sum_{s'} p^a_{ss'}(r(s,a) + \gamma v_\pi(s')) \tag{4.20}$$

The optimal state-value function can be expressed as:

$$v_*(s) = \max_\pi v_\pi(s) \tag{4.21}$$

while optimal state-action value function can be written as:

$$Q_*(s,a) = \max_\pi Q_\pi(s,a) \tag{4.22}$$

4.6 Q-learning algorithm

Q-Learning is a value-based reinforcement learning algorithm which is used to find the optimal action-selection policy using a Q-function. The Q-function we refer to, is the function described in the previous section.

In Q-Learning we define a Q-Table that is a lookup table where we calculate the maximum expected future rewards for action at each state. In the Q-table, the columns are the actions and the rows are the states. This process is an iterative process because we need to improve the Q-Table at each iteration. The Q-function uses the Bellman equation and has two inputs: the state s and the action a. The Bellman Equation divides the value function into sub-problems, so it simplifies the computation of the value function. The Q-learning equation is given by:

$$Q(S_k, A_k) = Q(S_k, A_k) + \alpha[R_{k+1} + \gamma \max_a Q(S_{k+1}, a) - Q(S_k, A_k)]$$

When we start, all the values in the Q-Table are zeros. There is an iterative process of updating the values. As we start to explore the environment, we obtain observations, so the Q-function gives us better approximations by continuously updating the Q-values in the table.

The Q-learning algorithm process has 5 steps:
1) Initialize Q-Table: initialize the values of the Q-Table at zero.
2) Choose an action.
3) Perform action.
4) Measure reward.
5) Update Q-table.

In the first three steps, a strategy called the epsilon greedy strategy is used. Epsilon greedy policy is a way of selecting random actions with uniform distribution from a set of available actions. At the beginning, the epsilon rates will be higher. The UAV will explore the environment and randomly choose actions. As the agent explores the environment, the epsilon rate decreases and the drone starts to exploit the environment. The actions are taken from an actions space.

In the last two steps, we need to update the Function $Q(s, a)$.

	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
s_0	490	490	490	700	490	490	490	490
s_1	490	490	700	490	490	490	490	490
s_2	490	490	490	700	700	490	490	490
s_3	490	490	700	1000	700	700	490	490
s_4	490	490	490	700	700	700	490	490
s_5	490	490	700	700	1000	700	700	490
s_6	490	490	490	490	700	700	490	490
s_7	490	490	490	490	700	1000	700	490
s_8	700	490	490	490	700	700	1000	700
s_9	700	490	490	490	490	700	700	1000
s_{10}	700	490	490	490	490	490	490	490
s_{11}	490	490	490	490	490	700	490	490
s_{12}	490	490	490	490	490	700	700	490
s_{13}	490	490	490	490	490	700	700	700
s_{14}	490	490	490	490	490	490	700	700
s_{15}	490	490	490	490	490	490	490	700

Figure 4.2: Example of an updated Q-table

We will repeat this again and again until the learning is completed. In this way the Q-table will be updated.

In conclusion, initially we explore the environment and update the Q-table. When the Q-table is ready, the agent will start to exploit the environment and start taking better actions.

Chapter 5

Simulation Results

5.1 Simulation Set-Up

In this chapter I will explain how the algorithm that we presented before has been implemented to maximize the target detection and mapping accuracy without exceeding the mission time and avoiding collisions. In addition, I will also show how it behaves in different cases of simulation.

Taking into consideration the iterative nature of the algorithm, we have chosen a maximum number N of interactions for each episode. The chapter has been divided in three sections: 1) Tools; 2) Implementation; 3) Cases of study.

In the first section will be discussed the computing environment chosen for the implementation of the code, that is MATLAB. In the second section will be described how the code has been implemented and the functions used for obtaining the results. In the last section various scenarios of simulation will be presented.

We consider a target detection and mapping problem performed by UAVs which autonomously navigate an indoor environment. In our case of study we consider multiple autonomous UAVs with radar capabilities. The scenario investigated is a situation where the UAVs reveal the presence of targets in an unknown environment. There are multiple targets that have to be detected. Obstacles are present in the environment, and they should be avoided by UAVs. The algorithm considers obstacle collision avoidance so that UAVs can navigate in cluttered environments. The algorithm has met its goal when the UAVs have found the largest number of targets within the mission time.

In this scenario the UAVs have to rapidly decide where to explore the environment while reconstructing a map of it and meanwhile detect the targets [12].

5.2 Tools

For the implementation of the algorithm, we used MATLAB R2020b.

MATLAB is a high-performance language for technical computing. It is an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. We utilized MATLAB because the libraries are specifically optimized for scientific purposes.

MATLAB allows us to implement and develop our algorithm, process images, create simulation videos easily and perform extensive data analysis and visualization.

5.3 Implementation

To implement the code, initially, we define some important parameters which are fundamental to define the initial conditions. We define the discount factor, the learning rate, the number of episodes and the number of agents respectively as:

$discount = 0.99;$
$alphaRL = 0.9;$
$Nepisode = 20;$
$Nagents = 2;$

To give the simulation a limit of time, that can be related to the battery-life of the UAVs, we define a mission time in which the tasks must be completed:

$T_M = 100;$

In the previous chapter two main reinforcement learning algorithm have been analyzed, that are Q-learning and SARSA algorithms. To decide which reinforcement learning algorithm we want to use in our implementation we use a flag:

$RLflag = 2;$

The value of the $RLflag$ represents the fact that, in our case of study, we choose the Q-learning algorithm as the selected reinforcement learning algorithm. This choice has been made by the fact that Q-learning is best suited for training an optimal agent in simulation because it directly learns the optimal policy, whilst SARSA learns a near-optimal policy during exploration.

Important parameters defined in the code are the number of targets in the environment and the number of agents that must navigate the surrounding searching for the targets.

The agents, before the end of the mission time, need to navigate the surrounding and search for the targets that are inside the indoor environment that we built for this experiment. The action chosen in the next step is decided based on which action will bring to the higher cumulative reward to achieve the goal in the shortest time possible.

An agent can perform four different actions, it can go left, right, up, or down. As said before, the algorithm chooses the actions that bring to the higher reward.

The agents navigate in an indoor environment that we created for this simulation. We generated a 20x20m map where each cell is 0.5x0.5m:

$Delta = 0.5;$
$SizeX = 10;$
$SizeY = 10;$
$xgrid = 0 : Delta : SizeX;$
$ygrid = 0 : Delta : SizeY;$

The environment was created thanks to a function called “createMap”. This function generates the walls, the obstacles in the surrounding and a number of dots representing the targets (in red) and the agents (in blue).

Because we adopted Q-learning, we should first initialize the Q-table. Initially, the Q-table, is a matrix where each element is zero-valued and then, after some interactions between the agents and the environment, the Q-table will be filled with useful data that represent the future rewards.

To understand the energy accumulated by each agent, we plot a detection map, that is generated taking into consideration the sum of the SNRs sensed by each agent. The resulting plot presents a number of peaks equivalent to the number of targets in the environment. These peaks are the point in the map where we find the targets.

The SNR is calculated thanks to a function called "*EvaluatePd*" that takes as inputs the targets and the coordinates of every point of the map and then in output we have two parameters, the SNR (signal-to-noise ratio) and the Pd (probability of detection).

Then we define a function called "*RLSLAM*". In "*RLSLAM*" we define the RL-based algorithm for trajectory optimization and environment mapping. We start by evaluating the policy for Q-learning and by calculating an expectation of the reward thanks to the function "*OFFPolicyEvaluation*".

The function "*OFFPolicyEvaluation*" follow three important steps:

a) The first step considers a greedy approach. We set "epsilon" according to the time instant. If good rewards are received, we will stop doing random actions and the actions will be calculated in a way to maximize reward. If the episode we find ourselves in is a number between 1 and $Nepisode/2$ then we use a non-greedy, i.e. random, approach, instead, if the episode is a number after $Nepisode/2$ then we go straight to the goal. The value that we obtain in output using the greedy approach, is a value called "epsgreedy".

"Epsilon" is a small value. With a small probability of epsilon, we choose to explore, not to exploit. In this case, the action is selected randomly, independent of the action-value estimates.

b) The second step consist in using the function "*getQtableStateIndex*". The function "*getQtableStateIndex*" takes as input the current agent position and in output we receive the row index of the Q table matrix.

c) The third step is updating the position of the agent.

So, initially, we define a variable "prob" that has a possible random value from 0 to 1. If the value of "prob" is smaller than the value "*epsgreedy*" found before, we have that the next position will be calculated as the past position considering a movement in a random action.

If “prob” is higher than the value “*epsgreedy*”, we calculate the maximum value of the Q-table and the output will be the action to choose in the next step. So, the next position will be calculated as the past position plus the output received from the calculation of the maximum.

This function is used together with a function called “AvoidWalls”. This last function checks the presence of walls (through agent proximity sensors) according to the current agent’s position.

“AvoidWalls” returns a constant that defines if there is a wall near the agent, or if there is an obstacle up, down, left, or right. If there is not a wall near the agent, the output value of the flag “OP” is 0, instead if there is an obstacle in the vicinity of the agent, the output value is 1. If the flag “OP” is equal to 1 then this is translated as a reward of -1000 in the Q-table.

All the values in the Q-table that have a value of -1000 are seen as value that cannot be changed, because are values that indicates the fact that we are near an obstacle or a wall of limitation, while every single value that has a different reward from -1000 is seen as a value to update. We use the Q-learning updating equation to update the values of the Q-table.

While the agents explore the environment, the Q-function gives us much better approximation through the continuous updating of the Q-table values. The Q-table contains values that represent the future reward. These values let us understand which is the best action to take in a certain moment in time, being in a certain state in the environment.

The function “$OFFPolicyEvaluation$” calls “*rewardReceive*” that is a function that calculates the reward for coverage, entropy and for the probability of detection. This function evaluates the expected reward, for each possible trajectory. Each type of reward is multiplied by a factor that represent the reward weight. The sum of these three components gives us the received reward.

The update of the Q-table utilizes another important information about its values. If the target is detected, so the received energy is higher than a certain value, the considered target must be switched off, because it has already been spotted. So, the algorithm stops the update of the Q-table for the target already identified.

In this way the agent continues to search for new targets in the environment instead of staying near the found target and not contribute to the achievement of the objective, that is finding the largest number of targets in the surrounding.

5.4 Case of study

In this section, five different scenarios will be described to understand how the algorithm behaves in different situations. We will consider three situations where we only change the number of targets and two situations where we change the value of the discount factor and the value of the learning rate.

To understand how the behaviour has improved during interactions, we will study the trajectories that have been followed in episode 1, that is the first episode, and in episode 20, that is the last episode. We expect that in the last episode, the UAVs have learnt the best trajectory to optimize their tasks.

First, we are going to analyze the trajectories of the first agent, and then the trajectories of the second agent. As the first agent we refer to the one at the bottom of the map while the one at the top is the second agent.

5.4.1 Two agents and N targets

In this subsection, I will simulate the behaviour of two UAVs that are in search of a certain number of targets that is different for each simulation. We consider three simulations, one where there is only one target in the environment and other two, where there are two and four targets respectively.

In the next, we will report a reference and SNR maps to virtually understand where the targets and obstacles are.

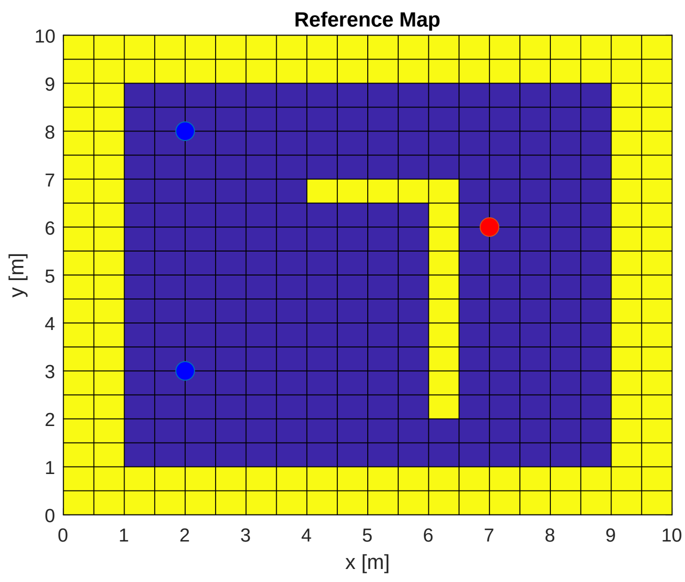

Figure 5.1: Case 1) Two agents and one target

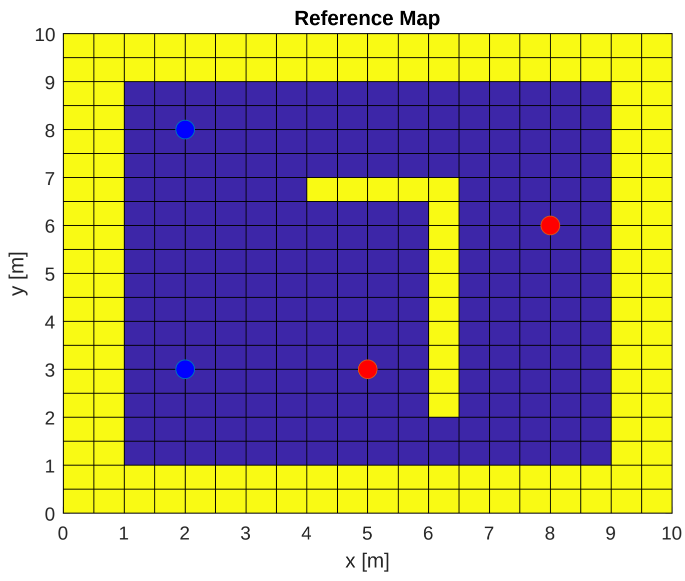

Figure 5.2: Case 2) Two agents and two targets

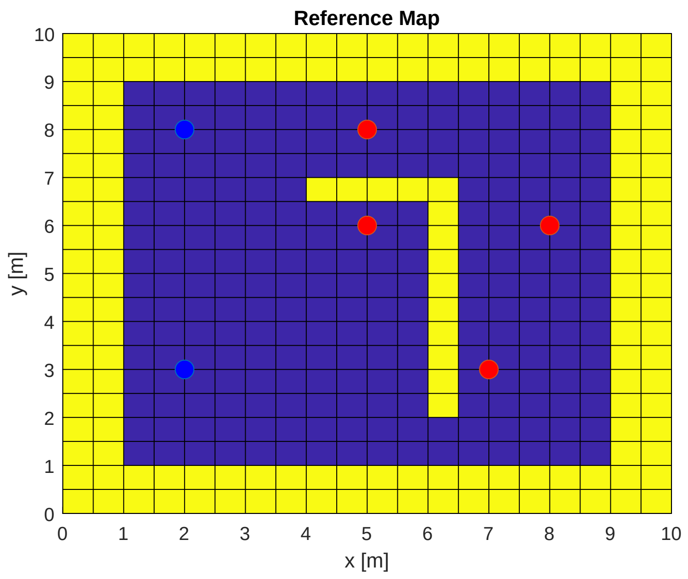

Figure 5.3: Case 3) Two agents and four targets

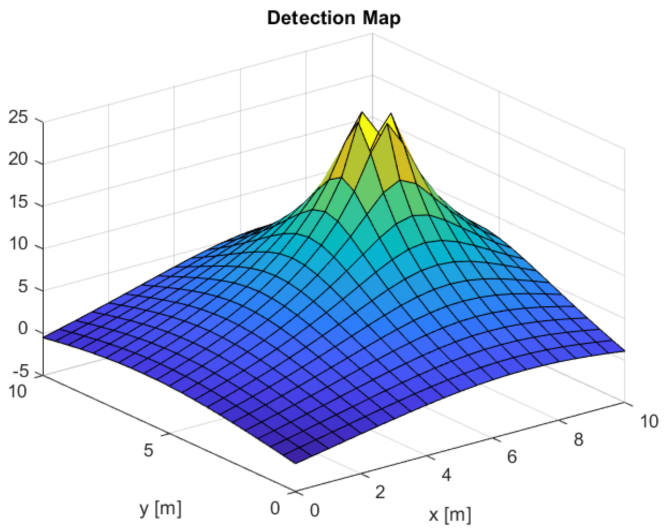

Figure 5.4: Case 1) Detection Map for one target

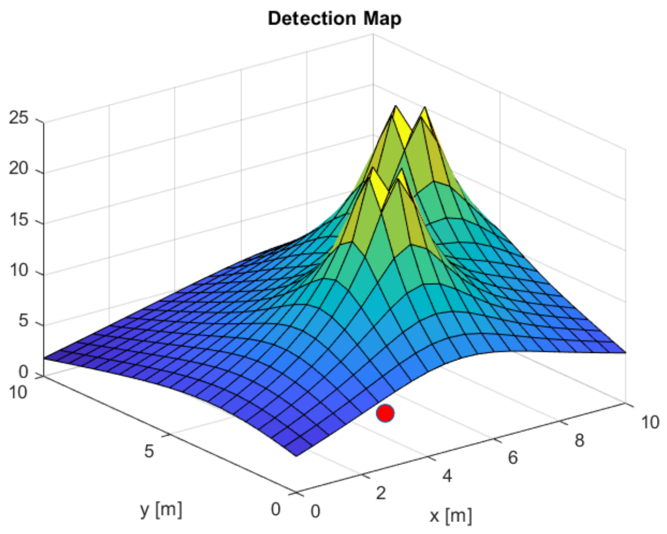

Figure 5.5: Case 2) Detection Map for two targets

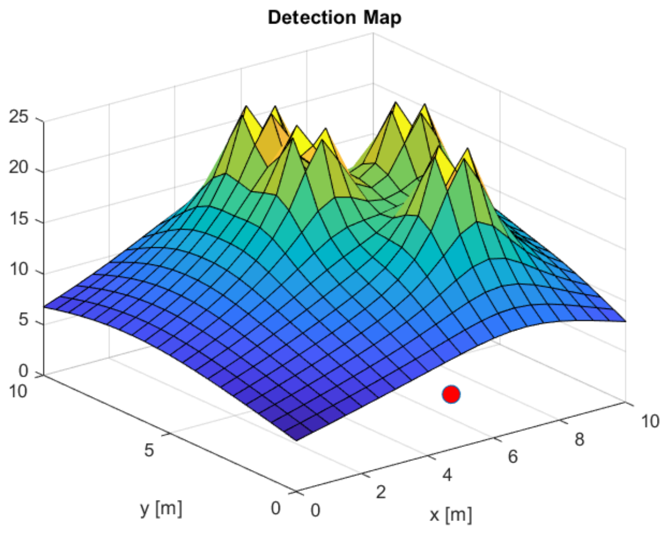

Figure 5.6: Case 3) Detection Map for four targets

Now, analyzing the first and last episode of the algorithm for each agent and case of study, we have:

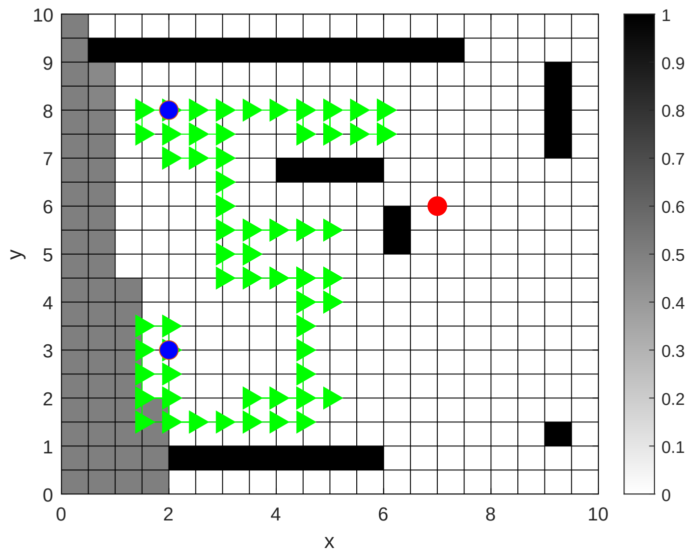

Figure 5.7: Case 1) First agent - Episode One

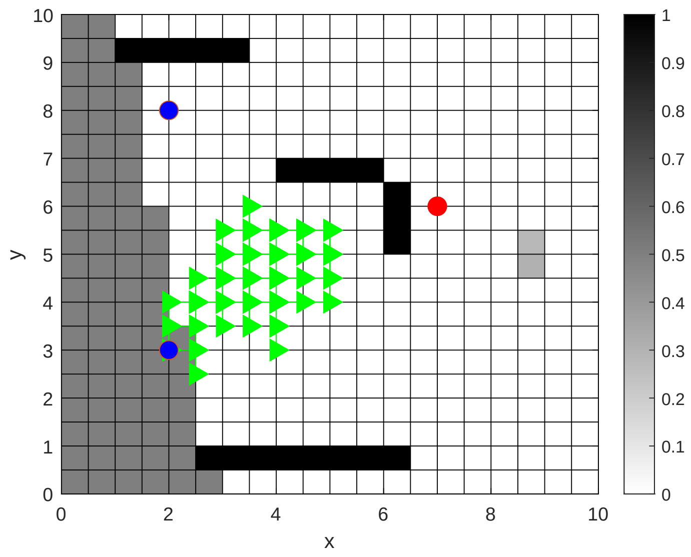

Figure 5.8: Case 1) First agent - Episode twenty

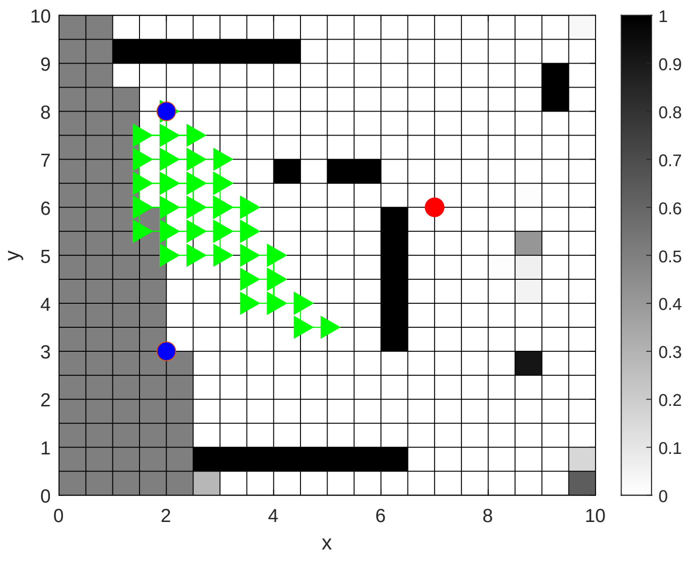

Figure 5.9: Case 1) Second agent - Episode One

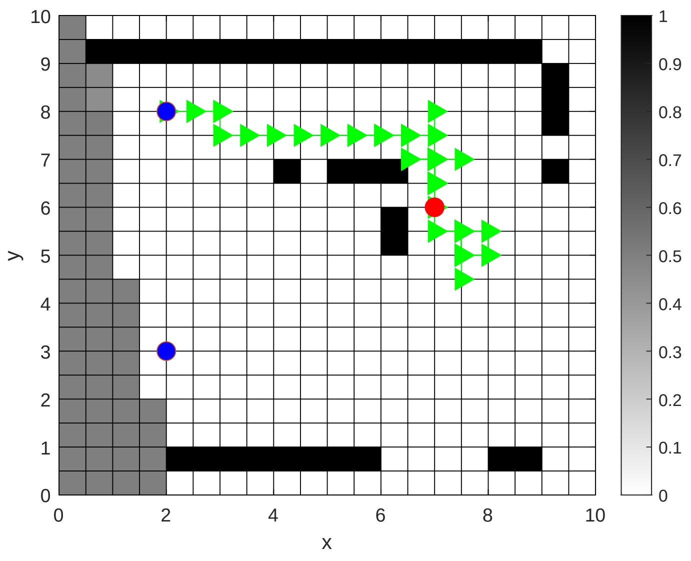

Figure 5.10: Case 1) Second agent - Episode twenty

For the first agent, in episode one (Figure 5.7), we can see by the trajectory, that the UAV navigates the environment in search of the target but once we arrive at the end of the first episode, the agent has not found it. At episode twenty (Figure 5.8), after the exploration phase, the agent privileges coverage compared to target detection. In each episode the UAV acquires more information and once it has arrived at episode twenty it has learned which are the actions that brings to the higher rewards. The second agent, like the first one, navigates the environment to find the target. In the first episode (Figure 5.9), we can see that the agent has not found the target but continues to explore the environment to acquire the largest number of information. Instead, in episode twenty (Figure 5.10), we can see that the UAV started to exploit the collected information, in fact, the UAV has been capable of finding the target and also infer a good trajectory for the identification of the target.

Now that we have described the first case of simulation, we can move on to the second case, where the number of targets increments from one to two targets in the environment.

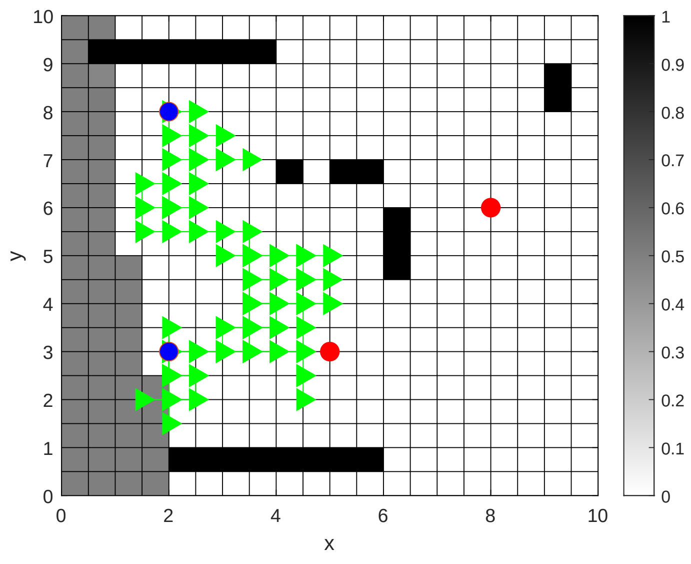

Figure 5.11: Case 2) First agent - Episode One

In episode one (Figure 5.11), the first UAV has mapped the environment and during the navigation has found a target. This does not stop the agent to search for the other target, so it continues to navigate until the episode terminates. At episode 20 (Figure 5.12), we can see that the walls are estimated correctly and the trajectory of the agent is directed to the target:

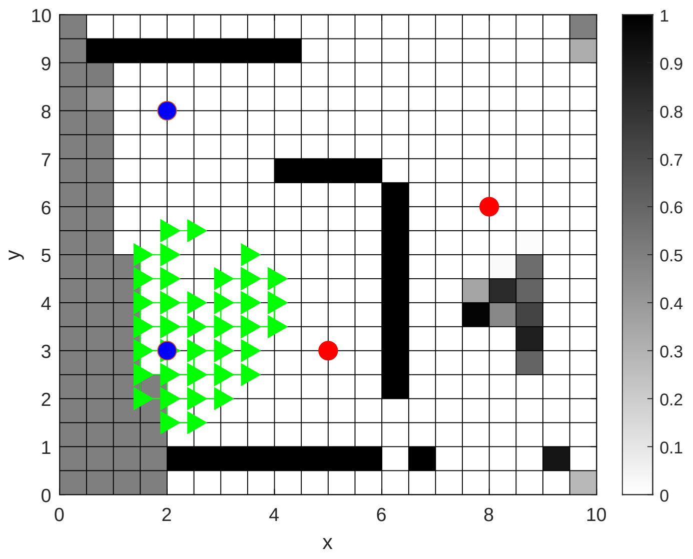

Figure 5.12: Case 2) First agent - Episode twenty

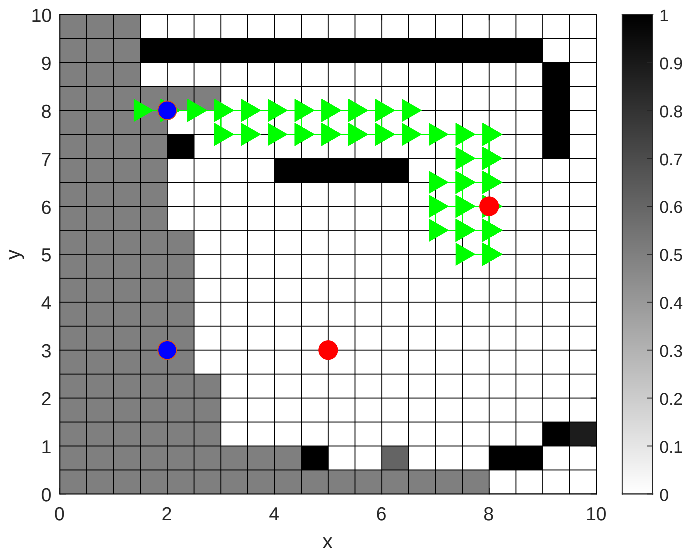

Figure 5.13: Case 2) Second agent - Episode One

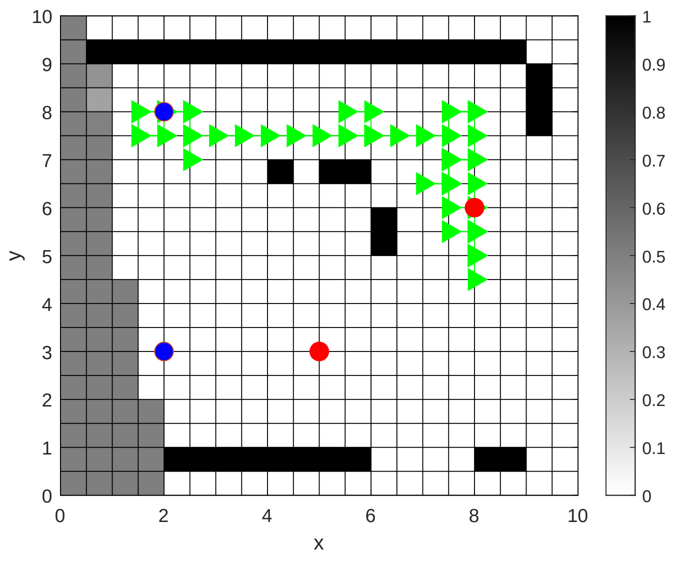

Figure 5.14: Case 2) Second agent - Episode twenty

The second agent, from episode 1 (Figure 5.13) to episode 20 (Figure 5.14), updates the trajectory to directly go to the target, because in this way it receives a higher reward.

The last simulation, increments the number of targets to four targets in the environment. The behaviour of the UAVs is the same as in the other cases: navigate the environment for reconstructing a reliable copy of the map and finding the highest number of targets in the surrounding without exceeding the mission time.

In this case, we can see clearly, how the UAVs, from episode one (Figure 5.15) to episode twenty (Figure 5.16), use the information acquired during the exploration phase to choose the actions that bring to the highest reward, that are the actions that bring the agents closer to the target.

In fact, as in each case we have simulated, initially, we have that the agent explores the environment and chooses random actions, but once the exploration phase is terminated, the exploitation phase starts, where we use the Q-table as a reference and the agent selects the action based in the maximum value of the Q-table.

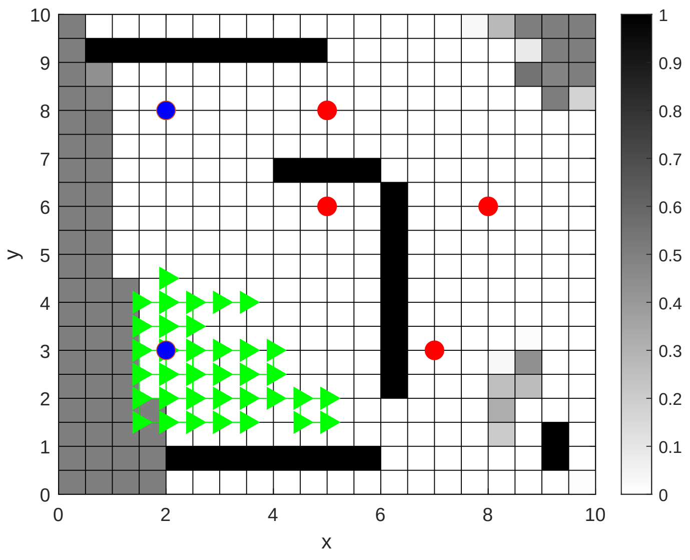

Figure 5.15: Case 3) First agent - Episode One

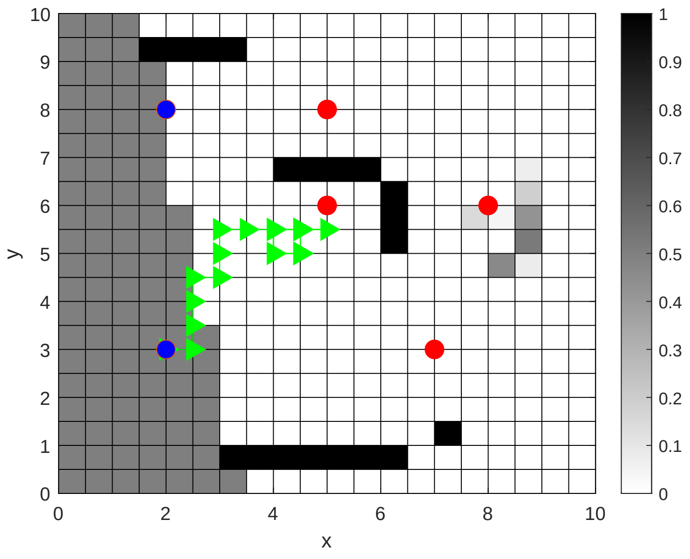

Figure 5.16: Case 3) First agent - Episode twenty

And for the second agent we obtain the same behaviour.

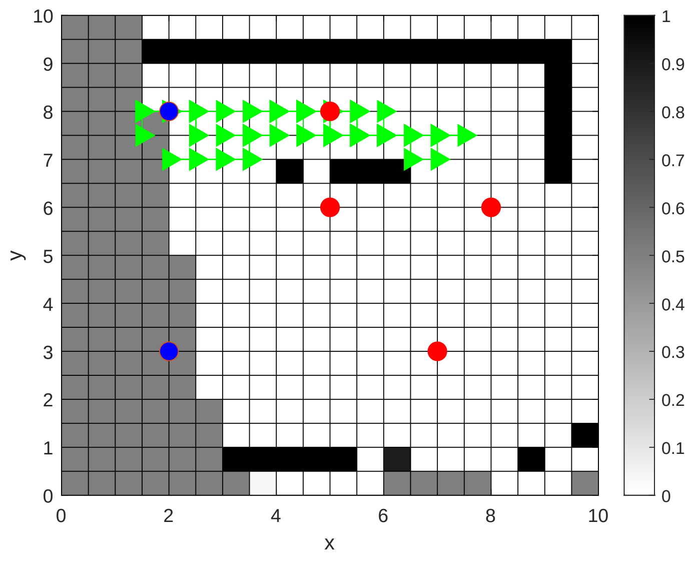

Figure 5.17: Case 3) Second agent - Episode One

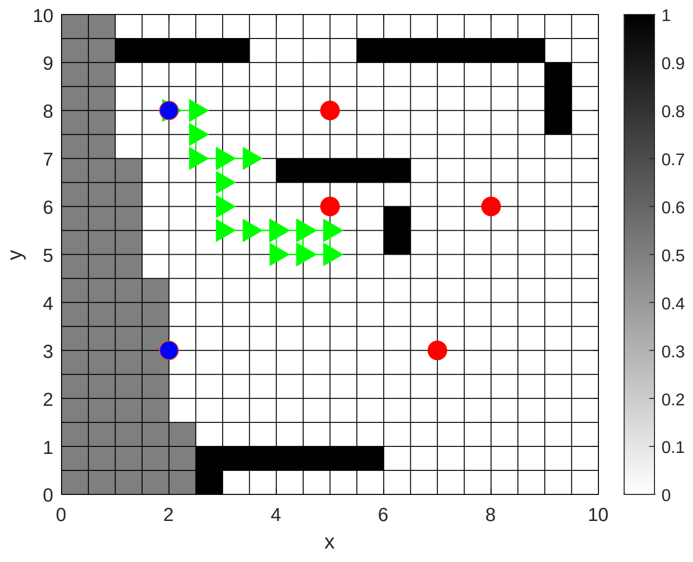

Figure 5.18: Case 3) Second agent - Episode twenty

Now we are going to study two simulations in which we consider two agent and two target, and we are going to change two important learning parameters that are the discount factor and the learning rate.

5.4.2 Two agents and two targets with low value of the learning rate

In this case, we have two agents that navigate the environment in search for two targets. The only thing that changes from the above cases is that here, we consider a learning rate reduction.

The learning rate is basically the speed at which a machine learning model learns. The learning rate is important for the update of the Q-values. In fact, together with the discount factor, is one of the most important parameters in the Q-learning updating equation. A value of 0 of the learning rate means that the Q-values are never updated, so nothing is learned. A high value of the learning rate instead, means that learning can occur quickly.

The value of the learning rate in the cases discussed above was set to 0.9. In this case the learning rate has been set to 0.3.

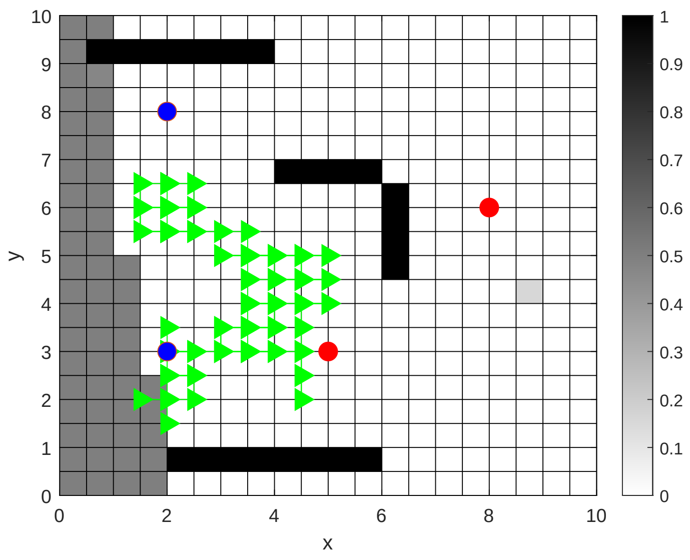

Figure 5.19: Case 4) First agent - Episode One

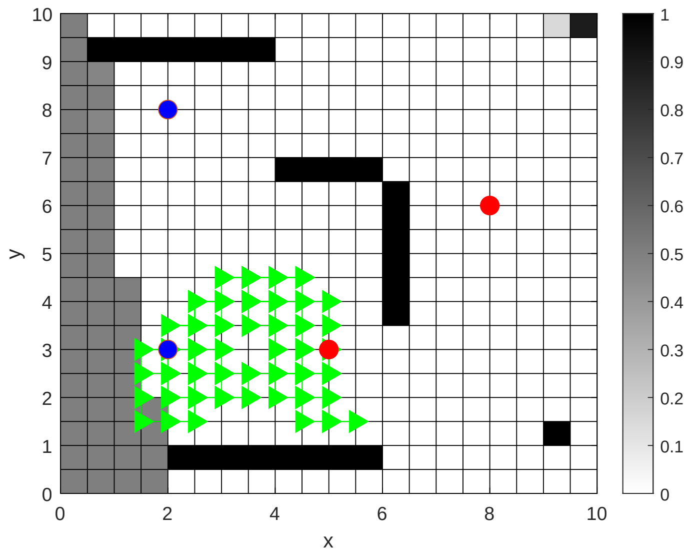

Figure 5.20: Case 4) First agent - Episode twenty

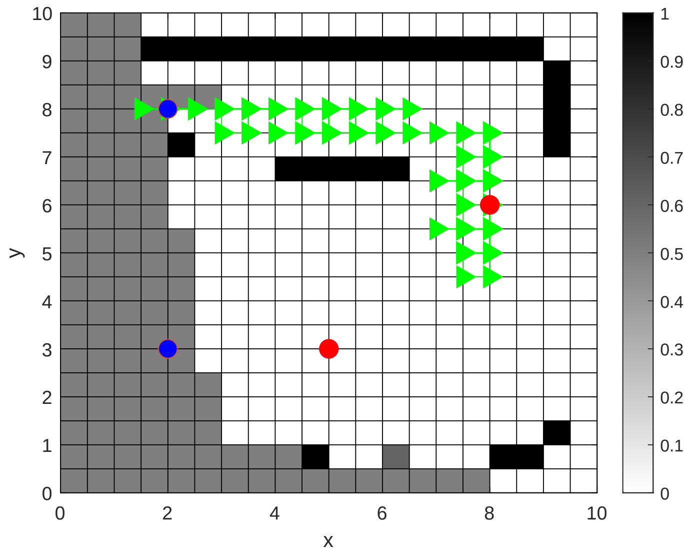

Figure 5.21: Case 4) Second agent - Episode One

When changing the learning rate, we see a slightly change from episode 1 (Figure 5.19 and Figure 5.21) to 20 (Figure 5.20 and Figure 5.22).

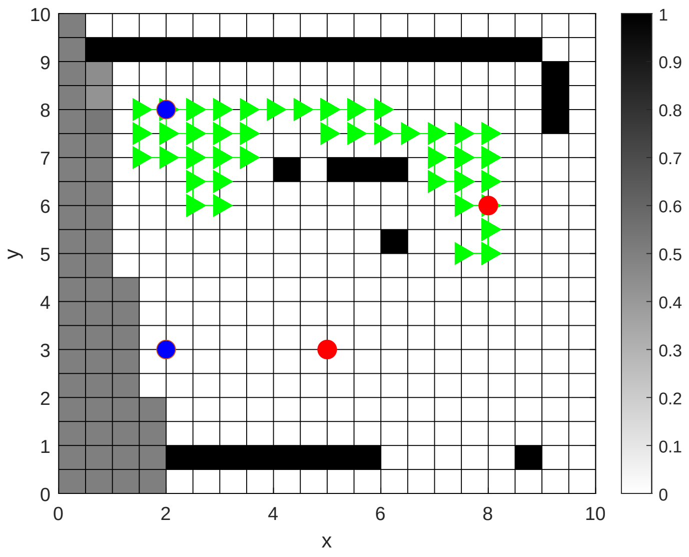

Figure 5.22: Case 4) Second agent - Episode twenty

Reducing the learning rate, slow the update of the Q-table, because the increase of the values is reduced. The learning rate is the value by which the update component, that is the subtraction of the old information from the new information, is multiplied. This is why the learning rate affects so much the update of the Q-values.

5.4.3 Two agents and two targets with low value of the discount factor

In this simulation, we have two agents and two targets but the change from the cases above, is that we reduce the discount factor while leaving unchanged every other parameter. The discount factor determines the importance of future rewards. So, essentially, the discount factor determines how much the reinforcement learning agents cares about rewards in the distant future.

Before we considered a discount factor of 0.99, while in this case we set the value of the discount factor to 0. When the discount factor has a value of 0, the agent will only learn about actions that produce an immediate reward.

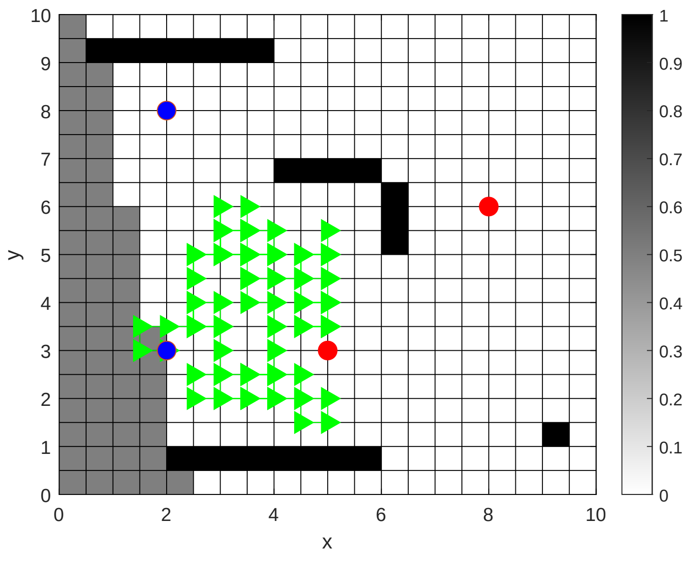

Figure 5.23: Case 5) First agent - Episode One

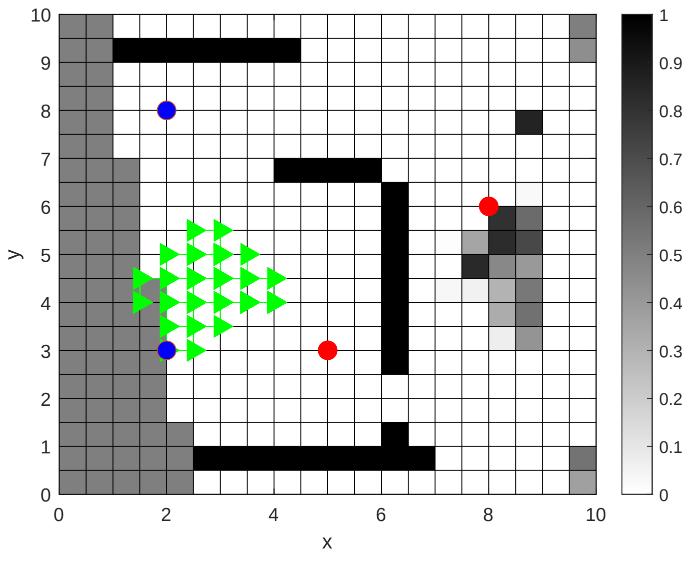

Figure 5.24: Case 5) First agent - Episode twenty

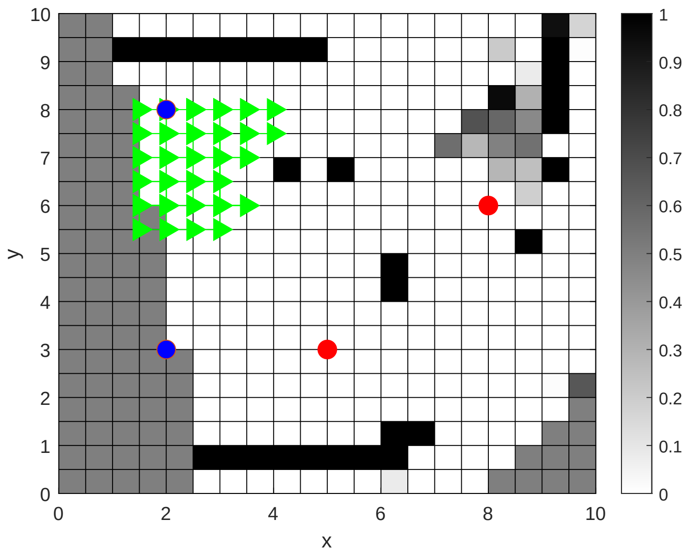

Figure 5.25: Case 5) Second agent - Episode One

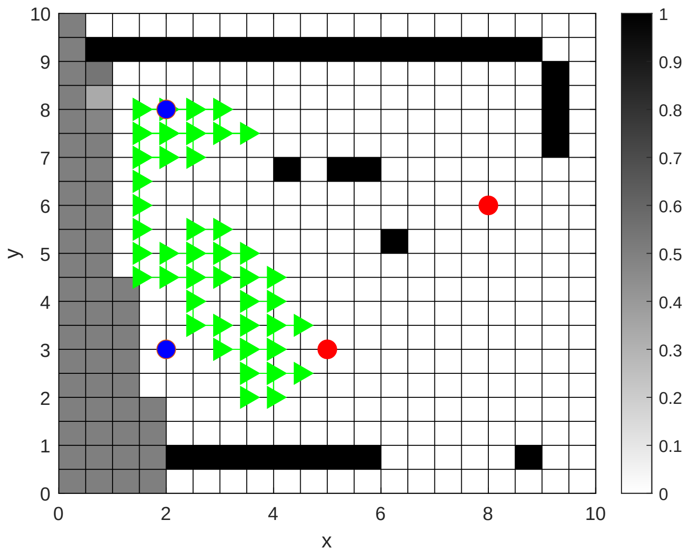

Figure 5.26: Case 5) Second agent - Episode twenty

Chapter 6

Conclusion

In this thesis aspects related to the implementation of a reinforcement learning algorithm for the autonomous navigation of UAVs and for the detection of multiple targets in an unknown indoor environment have been analyzed. I have first explained the concept of reinforcement learning and, more specifically of Q-learning. Then, I have described its implementation on MATLAB and analyzed the obtained results. Thanks to this algorithm, we have obtained useful and important results for the comprehension of the behaviour of UAVs. In particular, we have drawn important configuration by changing some key learning parameters, as the learning rate, the discount factor and the number of targets in the environment.

The algorithm that we implemented is based on a specific reinforcement learning algorithm, that is Q-learning. Q-learning is an off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. More specifically, Q-learning allows to learn a policy that maximizes the total reward.

Through simulations we have seen that the algorithm that we implemented completely follows the steps that characterize the Q-learning algorithm and allow us to reconstruct a reliable copy of the map and find a good trajectory to detect the targets present in the environment.

In addition, thanks to the underlying statistical models, we have understood the importance of two parameters, that are the discount factor and the learning rate. These two values, are fundamentals for an optimized update of the Q-values.

The simulation's results are very useful since we can see that the agent exhibits interesting capabilities in choosing the trajectory while achieving reliable performance in terms of detection and mapping accuracy in a given mission time.

A future development for this technology could be its implementation in emergency situations, such as during fires or in destroyed indoor environment. In such cases, the UAVs enter the environment without needing the presence of human operators. Thanks to this technology we could reduce the risk of severe injuries that could be caused if emergency operators would enter the perimeter, and also, the environment could be mapped in a much more effective way and faster by UAVs in comparison with human help. So, UAVs can reconstruct a map of an unknown environment while minimizing the error of detecting the presence of a target and this is a fundamental concept in situations where people are in danger.

Acknowledgments

First of all I would like to thank Prof. Davide Dardari and Anna Guerra for helping me during the thesis period. Thank you for the opportunity that you gave me by working with you, it has been a pleasure for me.

A big thank you goes to my family for being my strength during these years of study. You made me who I am today and I could not ask for more.

To my friends, that made everything easy and fun even on the worst days, you are literally the most beautiful thing that happened to me, I will be forever grateful to you. You will always be in my heart.

List of Figures

Bibliography

[1] D. Dardari, E. Falletti, and M. Luise, *Satellite and terrestrial radio positioning techniques: a signal processing perspective.* Academic Press, 2012.

[2] F. Nekoogar, *Ultra-wideband communications: fundamentals and applications.* Prentice Hall Press, 2005.

[3] D. Dardari, A. Conti, U. Ferner, A. Giorgetti, and M. Z. Win, "Ranging with ultrawide bandwidth signals in multipath environments," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 404–426, 2009.

[4] A. Ahmad, P. Claudio, A. Alizadeh Naeini, and G. Sohn, "Wi-fi rss fingerprinting for indoor localization using augmented reality." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 5, no. 4, 2020.

[5] S. Haykin, "Cognitive radar networks," in *Fourth IEEE Workshop on Sensor Array and Multichannel Processing, 2006.* IEEE, 2006, pp. 1–24.

[6] ——, "Cognitive radar: a way of the future," *IEEE Signal Processing Magazine*, vol. 23, no. 1, pp. 30–40, 2006.

[7] C. Wolff, "Frequency-modulated continuous-wave radar (fmcw radar)." [Online]. Available: https://www.radartutorial.eu

[8] M. I. Skolnik, "Introduction to radar," *Radar handbook*, vol. 2, p. 21, 1962.

[9] M. Lotti, G.Pasolini, A. Guerra, M. Caillet, R. D'Errico, and D. Dardari, "Radio simultaneous localization and mapping in the terahertz band," 2021.

[10] M. C. Kemp, "Millimetre wave and terahertz technology for detection of concealed threats-a review," in *2007 Joint 32nd International Conference on Infrared and Millimeter Waves and the 15th International Conference on Terahertz Electronics.* IEEE, 2007, pp. 647–648.

[11] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995.

[12] A. Guerra, F. Guidi, D. Dardari, and P. M. Djuric, "Reinforcement learning for uav autonomous navigation, mapping and target detection," in *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2020, pp. 1004–1013.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[14] J. Torres, "The bellman equation," 2021. [Online]. Available: https://towardsdatascience.com/the-bellman-equation-59258a0d3fa7