Visual robot navigation incorporating causal models in deep reinforcement learning

By:

Nilda Gabriela Xolo Tlapanco

A Dissertation Submitted in partial fulfillment of the requirement for the degree of

MSc degree in Computer Science

at

Instituto Nacional de Astrofísica,

Óptica y Electrónica

September 2024

Tonantzintla, Puebla, México.

Advisored by:

Eduardo Morales, Ph.D., INAOE

Enrique Sucar, Ph.D., INAOE

Ernesto Goméz-Balderas, Ph.D., GipsaLab UGA

Contents

List of Figures

Acknowledgements

I want to express my sincere gratitude to my advisors Dr. Eduardo Morales, Dr. Enrique Sucar, and Dr. Ernesto Gómez-Balderas, and to the review committee members.

To Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), for allowing me to use their facilities and financial support.

To Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), since this research project was possible thanks to financial support (823574).

To the Embassy of France in Mexico for the financial support (140440T) and help in the stance in France during the second year of this master.

To my partner Raul Romero, for the support and love in my life and during my studies.

To my father Gabino Xolo Gozcon, my mother Maria Antonia Tlapanco Perez, and my sister Lizbeth Xolo Tlapanco for loving me and supporting me all the time.

To all my friends and people in INAOE and UGA who have helped me to continue and end my studies.

Abstract

The use of UAVs or drones has been applied for different domains and an interest in common of them is the development of an autonomous pilot, Reinforcement Learning (RL) can be used for visual navigation in drones but needs substantial computational resources, long training times, and issues like sample efficiency, robustness, and interpretability. A field that combines RL with causal models called Causal Reinforcement Learning promises to reduce the time for exploration and improve adaptation to novel situations or environments but is not proven yet in robotics tasks, for this reasons we propose adapting an RL algorithm to learn and use knowledge of a causal model to improve the action selection in the visual-based navigation. We present our results with the use of a Causal Bayesian Network defined and the learning of it in the RL training phase to use in a real-world drone.

Resumen

El uso de UAVs o drones se ha aplicado en diferentes dominios y un interés en común entre ellos es el desarrollo de un piloto autónomo. El aprendizaje por refuerzo (RL) se puede utilizar para la navegación visual en drones pero necesita importantes recursos computacionales, largos tiempos de entrenamiento y presenta problemas como eficiencia de las muestras, robustez e interpretabilidad. Un campo que combina RL con modelos causales llamado Aprendizaje por Refuerzo Causal promete reducir el tiempo de exploración y mejorar la adaptación a situaciones o entornos novedosos pero aún no ha sido probado en tareas de robótica, por estas razones proponemos adaptar un algoritmo de RL para aprender y utilizar el conocimiento de un modelo causal para mejorar la selección de acciones en la navegación basada en imágenes. Presentamos nuestros resultados con el uso de una Red Causal Bayesiana definida y su aprendizaje en la fase de entrenamiento de RL para su uso en un dron del mundo real.

CHAPTER 1

INTRODUCTION

The use of Unmanned Aerial Vehicles (UAV) or drones, has recently increased, in recent years, due to the growth interest in using them for entertainment, but also for military, agriculture (Radoglou-Grammatikis et al. (2020)), medicine (Balasingam (2017)), transport (Menouar et al. (2017)), delivery services (Gatteschi et al. (2015)), construction (Qasim et al. (2022)), and rescue applications (Mario Silvagni and Chiaberge (2017)). The principal disadvantage is the need for a human pilot to control it, multiple research work includes the use of different intelligent artificial techniques to develop autonomous pilots (Hasan et al. (2020),García et al. (2020), Hanover et al. (2024)), not only for the cases where the human pilot is not available but also to improve the human pilot behavior, one of the most popular techniques is Reinforcement Learning. Reinforcement Learning has been used to make an autonomous pilot for collision avoidance and seeking an objective/goal in different environments (Darwish and Nakhmani (2023), Shin et al. (2019), Çetin et al. (2019), Kersandt et al. (2018)), autonomous navigation usually needs data from multiple sensors but in the real world, the easiest, and in many cases the only, sensor to use is the onboard camera. In addition to this, a large amount of data and long training times are necessary to train the algorithm. A promising alternative is to use Causal Discovery to construct a causal model of the environment while the agent is trained, which can help accelerate the training phase.

1.1 Motivation

Reinforcement Learning proves to be a good option for visual navigation in drones. However, it needs a large number of interactions with the environment to learn good policies. At the same time, Causal Reinforcement Learning is still an emerging field that promises to reduce the time for training and a good performance in transfer learning but is not proven in robotics tasks. Their use can boost the development of several applications in robotics tasks, including autonomous drone navigation.

1.2 Justification

Causal reinforcement learning integrates causal inference into Reinforcement Learning, aiming to understand and utilize the underlying causal relationships in the environment. This approach can potentially reduce the number of necessary interactions by enabling the drone to make inferences about unobserved or less frequent scenarios from its understanding of causal mechanisms, thus speeding up the learning process. By understanding causal interactions within the environment, drones can navigate more effectively, avoiding obstacles and optimizing routes more reliably than with traditional RL methods. Additionally, these advancements can also be applied to other AI-driven robotic tasks, contributing to the broader field of artificial intelligence in robotics.

1.3 Objectives

The general objective of this investigation is to incorporate an algorithm of Causal Discovery into Deep Reinforcement Learning to use the knowledge from the causal model in the learning of the policy. The specific objectives are:

1. Design and implement an algorithm to extract the relevant information from RGB or RGB-D images to create a causal model of the environment.

2. Design and implement a Deep RL algorithm incorporating the previous algorithm and use this knowledge to enhance the robot's decision-making process and optimal policy.

3. To validate and demonstrate the effectiveness of the algorithm for visual navigation in mobile robotics by employing both simulated testing and real-world trials.

1.4 Problem Statement

Computer vision in robots is used to perceive the environment with the help of visual sensors, such as cameras that provide rich and detailed information about the robot's surroundings. By analyzing visual data, robots aim to perceive and understand the environment in a manner similar to humans. This perception enables them to navigate, identify objects, recognize landmarks, make decisions based on visual cues, detect obstacles in their path, identify and track objects, estimate their positions and distances, and determine whether they pose a potential collision risk.

The robot can learn to navigate using various strategies, one of which is Reinforcement Learning (RL). RL algorithms enable the robot to navigate complex and uncertain environments, discovering optimal strategies to achieve goals. These models adapt to changing environments and learn new behaviors through ongoing interaction. They update their policies to accommodate variations, handle new scenarios, and adjust to different conditions. RL algorithms have the potential to facilitate autonomous decision-making in dynamic scenarios, allowing the robot to make real-time decisions and optimal choices without human intervention. One of the disadvantages is the high-dimensional state and action spaces, as the dimen-

sionality increases, the learning process becomes more challenging, and the time required for convergence may become prohibitive. Most recently the use of Deep RL involves training artificial agents to learn and make decisions in complex environments through the integration of deep neural networks and RL algorithms, where the neural network is updated based on the observed rewards and the agent's experiences. However, training Deep RL agents can be challenging due to the need for substantial computational resources, long training times, and issues like sample efficiency, robustness, and interpretability.

Causal models are used to study cause-and-effect relationships and make predictions or interventions based on those relationships. They help in understanding how changes in one part of a system can propagate and affect other parts. These models are particularly useful for decision-making, planning, and analysis of complex systems.

Incorporating knowledge from a causal model and their construction into RL algorithms, e.g., Feliciano-Avelino et al. (2021) Zhu et al. (2023) Méndez-Molina et al. (2022), can greatly speed up the learning process by reducing the need for extensive exploration, but their use in robotics task has not been yet proved. In this research, we show how it can be used to obtain an autonomous pilot enabled to reach an objective and avoid collisions in a drone.

This research pretends to answer the following research question:

1. How to extract relevant information from RGB or RGB-D images to create a causal model?

2. How can causal models be integrated into deep reinforcement learning algorithms to improve their efficiency and effectiveness and help in designing safer navigation policies for mobile robot navigation in diverse environments?

3. How can causal models help in transfer learning scenarios, where a mobile

robot trained in one environment needs to adapt and generalize its navigation skills to novel environments?

1.5 Methodology

The methodology followed in this research was the following:

- To study and implement Deep Reinforcement learning to the visual navigation in a drone in simulated environments
- To study and implement an algorithm for Causal Discovery to be added to the DRL algorithm
- Evaluate the DRL agent with a predefined Causal Bayesian Network and a Causal Bayesian Network created during training

1.6 Structure

The rest of this document is structured as follows. Chapter 2 describes the theoretical concepts needed to understand the techniques used in this research, Chapter 3 consists of a review of the most closely related work of state of the art, Chapter 4 describes in detail the methods employed in this research, Chapter 5 presents the experimental results to validate our proposal. Finally, Chapter 6 presents our conclusion from the experiments and future research work.

CHAPTER 2

THEORETICAL FRAMEWORK

This chapter provides a general overview of the concepts utilized in this work, the definition of Unmanned Aerial Vehicle, Visual Navigation, and the method of Machine Learning: Deep Reinforcement learning and their background in the Markov decision process; the description of Deep Q-Learning the algorithm that we use to prove the incorporation of the Causal Discovery, the definition of Causal Discovery and the elements needed to obtain it: Causal Bayesian Network, Hill Climbing Search and BIC score. The definition of the combination of these fields in Causal Reinforcement Learning, the simulation tools, and the real drone used to prove our work.

2.1 Unmanned Aerial Vehicle

Unmanned Aerial Vehicles (UAV also known as "drones") are aircraft that operate without a human pilot onboard. Essentially, a UAV is a flying robot that can be remotely controlled or fly autonomously through software-controller flight plans in its embedded system. UAVs come in various shapes, sizes, and configurations, ranging from small quadcopters to large fixed-wing aircraft. Based on how its movement is controlled, it can fly with a human remotely controlling the UAV, or in the most

advanced cases, fly itself without any human intervention. They can be equipped with a variety of sensors, including an Inertia Measurement Unit (IMU), Global Positioning System (GPS), cameras, and other payloads to perform a wide range of tasks including aerial photography, videography, search and rescue, infrastructure inspection, disaster response, surveillance, and reconnaissance (Garg (2021), Management Association (2019)).

Avoiding collisions and autonomous pilot is a critical aspect of a drone operation to ensure safety and prevent damage to the drone and surrounding objects or people and is an important part of the search for a goal or target. We use a UAV or drone (as we will refer to it in our work), to build a policy to navigate autonomously. To control the drone's movements in its environment we will use the technique of Visual Navigation.

2.2 Visual Navigation

Visual navigation is used in robotics to replicate the human visual system, where visual information from the environment is used to navigate or guide movements. It can be divided into binocular (multi-ocular) and monocular according to the number of sensors used. Compared with monocular, binocular are more costly and slower in calculating, but more accurate in positioning. Monocular visual navigation only requires a camera, with a simple structure and convenient and flexible operation, which makes it suitable for robot navigation in the field environment (Zhai et al. (2019)).

Autonomous navigation in drones usually uses a variant of different sensors but we will focus on the use of binocular visual navigation, the data obtained will be used to train a Reinforcement Learning agent to learn to navigate by receiving visual inputs from the environment. These elements: the environment where the drone learns to navigate, the drone, the visual input from the drone cameras, all possible movements

that the drone can make and the goal can be modeled into Reinforcement Learning by a Markov decision process.

2.3 Markov decision process

Reinforcement learning studies sequential decision problems. Mathematically, we can formalize these problems as Markov decision processes. Markov decision process (also written MDPs) relies on the notions of state, agent, action, and reward. Are defined as controlled stochastic processes satisfying the Markov property and assigning reward values to state transitions, formally described by a 5-truple (S,A,T,p,r) where:

- S is a state space in which the process of evolution takes place.
- A is a set of all possible actions that control the state dynamics.
- T is the set of time steps where decisions need to be made.
- $P_a(s, s') = Pr(S_{t+1} = s' | S_t = s, A_t = a)$ denotes the probability of transition (at time t) from state s to state s' under action a.
- $R_a(s, s')$ provides the immediate reward after transition from state s to s' with action a.

The set T of decision times is a discrete set, which can either be finite or infinite (then we talk respectively about finite horizon or infinite horizon). A third case corresponds to the existence of a set of terminal states (or goal states). In this case, the process stops as soon as one of these states is encountered.
MDPs allow us to model the state evolution dynamics of a stochastic system when this system is controlled by an agent choosing and applying the actions a_t at every time step t. The procedure of choosing such actions is called action policy or strategy

and is written as π. A policy can decide deterministically upon the action to apply or can define a probability distribution over the possible applicable actions. Then a policy can be based on the whole history h_t of the process (it is called history-dependent) or can only consider the current state s_t.

Solving a Markov decision problem implies searching a policy in a given set that optimizes a performance (or optimality) criterion for the considered MDP. This criterion aims at characterizing the policies that will provide the best sequences of reward (Sigaud and Buffet (2013)). One way to solve MDPs, particularly when the state and action spaces are too large for tabular methods is the use of Deep Reinforcement Learning.

2.4 Deep Reinforcement learning

Deep Reinforcement Learning (DRL) is a combination of deep learning (DL) and reinforcement learning (RL). DL is a subset of machine learning that uses neural networks usually with many layers to analyze various types of data. DRL combines the perception capabilities of deep learning with the decision-making capabilities of reinforcement learning. Traditional RL uses tabular methods or simple function approximators for learning policies, which can struggle with complex or high-dimensional environments. By using deep neural networks, DRL can handle more complex scenarios with less engineered feature extraction. One of the most used Deep RL algorithms is Deep Q-Learning.

2.5 Deep Q-learning

Deep Q-Learning is based on Q-Learning from Watkins and Dayan (1992), a model-free RL algorithm enabling the agent to learn how to act optimally in the environment by using a value function, that stores and retrieves estimates of the rewards for different actions in different states. The Q-learning algorithm updates the Q-values using the Bellman equation;

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Where:

- $Q(s,a)$ is the state-action value function, representing the expected utility of taking action a in state s.
- α the learning rate $(0 < \alpha \leq 1)$ determines to what extent newly acquired information overrides old information.
- r is the reward received after taking action a.
- $\max_{a'} Q(s',a')$ is the maximum predicted reward obtainable from the next state s', over all possible actions a'.
- γ the discount factor $(0 \leq \gamma < 1)$. It represents the difference in importance between future rewards and immediate rewards. A higher γ means the learning process considers future rewards more strongly.

In traditional Q-Learning, the Q-values are stored and updated into a Q-table for each state-action pair. However, this approach becomes infeasible for large and continuous state spaces.
Deep Q-Learning addresses this problem by using a deep neural network to approximate the Q-values. The network takes the state as input and outputs Q-values for

each possible action.

A target network is used in Deep Q-Learning (DQN) to stabilize the learning process and make it more robust. The primary purpose of the target network is to provide stable target Q-values during the training process. The target network is a copy of the Q-network with frozen parameters, which means its weights are not updated during the training process. The target network is used to calculate the target Q-values for updating the Q-network, while the Q-network itself is used to select actions during exploration and exploitation.

The target network is updated periodically by copying the weights from the Q-network. This update frequency can be set based on a fixed number of iterations or a specific time interval. By using a separate target network with delayed updates, the training process becomes more stable.

Experience replay (Sewak (2019)) is employed in DQN to enhance the efficiency and effectiveness of the learning process by mitigating the correlation between consecutive experiences. It involves utilizing a memory to store tuples of information comprising the action taken a_t, the state observed s_t, the received reward r_{t+1}, and the subsequent state s_{t+1} after the RL agent performs an action in a particular state. When the memory size reaches a predefined limit, the tuples are randomly sampled. The target Q-value function is then computed using a separate Target Network. Subsequently, the Target Network model is trained for one epoch using the sampled states and the Q-values obtained from the target Q-value function. Finally, the action with the highest Q-value output is selected based on the ϵ-greedy methods.

Deep Q-learning often requires a large number of interactions with the environment to gather sufficient data to train the neural network. All this data be used to aim the decision action of the algorithm, one alternative to take advantage of them is the Causal Discovery.

2.6 Causal Discovery

Causal Modeling attempts to resolve questions about possible causes by providing explanations of phenomena (effects) as the result of previous phenomena (causes). The difficulty inherent in establishing causation among variables can be illustrated by examining a variety of definitions of causality. Several writers like Lewis-Beck (1980) and Klecka (1980) specify three conditions that must be met in order to infer the existence of a causal relationship between two variables X and Y. The first condition states that there must be a concomitant variation or covariation between X and Y, such that changes in X are associated with changes in Y. This condition can be visually represented in a graph by a direct arrow from X to Y, indicating that Y potentially depends on X. The second condition requires a temporary asymmetry or time ordering between the two, in a graph, this is represented by a directed edge from X to Y, ensuring that the cause precedes the effect. The third condition requires the elimination of other possible causal factors that may be producing the observed relationship between X and Y, this is graphically handled by either showing no direct connections between Y and these other variables or by introducing additional nodes and edges that control for these factors, thereby isolating the direct effect of X on Y (Asher (1976)). This causality could be seen like dependence and independence relations. A natural way to represent the dependence and independence relations between a set of variables is using graphs, such that directly dependent variables are connected, and the independence relations are implicit in this dependency graph. Causal Bayesian Networks provide a framework for representing the causal relationships to allow for various kinds of causal inference and prediction based on the model.

2.6.1 Causal Bayesian Network

Bayesian networks (BN) are directed graphical models that represent the joint distribution of a set of random variables. In the graph, the nodes represent random variables, and the arcs direct dependencies between variables, unlike Causal Bayesian network (CBN) where each node represents a variable and the arcs in the graph represent causal relations; that is, the relation A $\rightarrow$ B represents some physical mechanism such that the value of B is directly affected by the value of A. This relation can be interpreted in terms of interventions/setting of the value of some variable or variables by an external agent. Causal networks represent stronger assumptions than Bayesian networks, as all the relations implied by the network structure should correspond to causal relations (Sucar (2020)).

In addition to the structure, like a BN the CBN considers a set of local parameters, which are the conditional probabilities for each variable given its parents in the graph, if there is an arc from A to B (A is a direct cause of B), then A is a parent of B, and B is a child of A. Given any variable X in a CBN, Pa(X) is the set of all parents of X. Also, similarly to BNs, when the direct or immediate causes (parents) of a variable are known, the more remote causes (or ancestors) are irrelevant (Sucar (2020)). Given a Causal Bayesian network (structure and parameters) we can answer several probabilistic queries. Learning a BN includes two aspects: learning the structure and learning the parameters, when the structure is known, parameter learning consists of estimating the conditional probability tables (CPTs) from data and using this data to convert the CB into a CBN. For structure learning there are two main types of methods: global methods based on search and score, and local methods that use conditional independence tests. We will use a global method: Hill Climbing Search.

2.6.2 Hill Climbing Search

Hill climbing is a method of mathematical optimization that is used in numerical analysis. It is a member of the family of techniques known as local search. It's an iterative method that begins with an arbitrary solution to a problem and then seeks to discover a better answer by making incremental changes to the initial solution in order to see whether it leads to a better solution. If the modification results in a better solution, then another incremental adjustment is made to the new solution, and so on and so forth, until there are no more improvements that can be identified (Sabry (2023)). In our case this algorithm will be used to learn the structure of Bayesian networks, It starts with an optional initial network and iteratively modifies it by adding, deleting, or reversing edges to find the network structure that has the best score according to the specified scoring method. The score used in this work to compare the structure is the BIC score.

2.6.3 BIC score

The Bayesian information criterion (BIC) is a criterion for model selection among a finite set of models. It's based on the likelihood function and it's commonly used because includes a penalty term, the BIC is defined as:

$$BIC = log(P(D \mid \Theta_G, G_i)) - \frac{d}{2} log N$$

Where d is the number of parameters in the BN, N is the number of cases in the data, G_i is the candidate structure, and Θ_G the corresponding vector of parameters (probability of each variable given its parents according to the structure). An advantage of this metric is that it does not require a prior probability specification and has a compromise between the precision and complexity of the model. However, given the high penalty on the complexity of the model, it tends to choose structures that are too simple (Sucar (2020)).

Once we have the structure of our Causal Bayesian Network and the method of learning it from raw data, we can incorporate it into the Q-learning algorithm, this process is called Causal Reinforcement Learning.

2.7 Causal Reinforcement Learning

As we already mentioned, the combination of Reinforcement Learning and Causal Modeling is a relatively new field called Causal Reinforcement Learning (CRL). CRL is a suite of algorithms that incorporate additional assumptions or prior causal knowledge into RL to analyze and understand the causal mechanisms underlying actions and their consequences, enabling agents to make more informed and effective decisions for more effective model learning, policy evaluation, or policy optimization (Deng et al. (2023)). It is formalized as a tuple (M, G), where M represents an RL model setting, e.g., MDP, POMDP, MAB, etc., and G stands for the causal-based information regarding an environment or task, e.g., causal structure, causal representation or features, etc.

It is generally divided into two categories, the first category is based on the prior causal information, where such methods typically assume the causal structure regarding to the environment or task is given a prior from the experts, while the second category is based on the unknown causal information, where the relative causal information has to be learned for the policy (Zeng et al. (2023)). Our work belongs to the second category, we built an algorithm to learn the causal information from the interactions of the RL agent with the environment to later use it in policy construction.

2.8 Simulation tools

To simulate the environment needed for the training phase of our mobile robot, we select ROS and Gazebo tools.

ROS, or Robot Operating System, is an open-source middleware framework specifically designed for robotics applications. It provides a collection of tools, libraries, and conventions that aim to simplify the development of complex robot software (Quigley et al. (2009)). ROS offers a flexible and modular architecture that allows the creation and integration of various components such as drivers, algorithms, and sensors into a cohesive robotic system.

One of the key features of ROS is its communication infrastructure, which enables different parts of a robotic system to communicate with each other seamlessly. This communication is typically done using a publish-subscribe messaging model, where nodes (software modules) can publish data to topics and subscribe to topics to receive data from other nodes. ROS is available under an open-source BSD license, allowing the ROS community to consistently offer open-source modules for others to employ.

Gazebo is a powerful open-source robotics simulation software that provides a realistic and flexible environment for simulating robots, environments, and sensors. Gazebo simulates the physical interactions between objects in the simulated environment, including gravity, friction, collisions, and dynamics. This enables realistic modeling of robot behaviors and interactions with the environment. Provides support for simulating various sensors commonly used in robotics, such as cameras, LiDAR, GPS, IMU (Inertial Measurement Unit), sonar sensors, and depth cameras. Gazebo is integrated with ROS, allowing one to control and interact with simulated robots using ROS messages and services. The simulated drone can interact with the virtual world environments in 3D (Koenig and Howard (2004)).

We select the Parrot Bebop 2 drone like our mobile robot (Figure 2.1). We use the rotors simulator package from Furrer et al. (2016) which facilitates the development,

testing, and evaluation of algorithms and software for multirotor UAVs within the ROS framework in simulation environments. The package includes pre-built models of multirotor UAVs, such as quadcopters, hexacopters, and octocopters, and supports the simulation of various sensors commonly used on UAVs, including cameras, LiDAR, GPS, IMU (Inertial Measurement Unit), and sonar sensors, the package allows us to communicate with it using ROS topics, services, and actions. This enables to send commands to control the drone's flight, receive telemetry data, and stream video from its onboard camera.

,

Figure 2.1: Parrot Bebop 2 uses in the simulated environment.

We used Ubuntu 20.04.05 LTS as the operating system and the Robotic Operating Systems (ROS) in its Noetic Ninjemys version.
For the environment, we selected an open environment represented in Figures 2.2 and 2.3 where the drone can navigate more easily. The goal is in the drone's view and avoiding an obstacle is required to reach the goal.

2.9 The real drone

For the case to prove the algorithm in the real world, we implement the algorithm in the Tello DJI drone (Figure 2.4), which is a small and lightweight drone designed for recreational and educational purposes. It is known for its ease of use, affordability, and programmability. DJI provides a Python software development kit (SDK) specifically tailored for Tello. This SDK allows to communicate with the Tello drone using Python scripts (DJI (2018)).

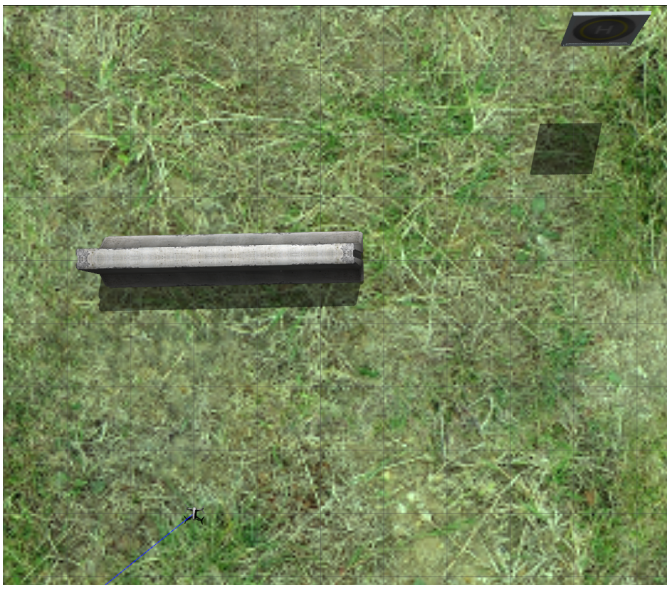


Figure 2.2: Aerial view of all the environment simulated.

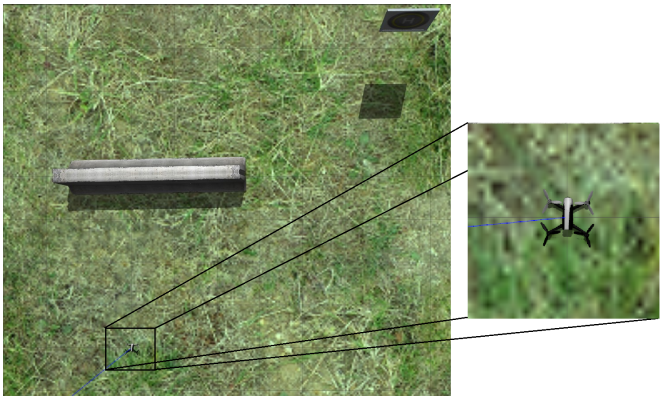


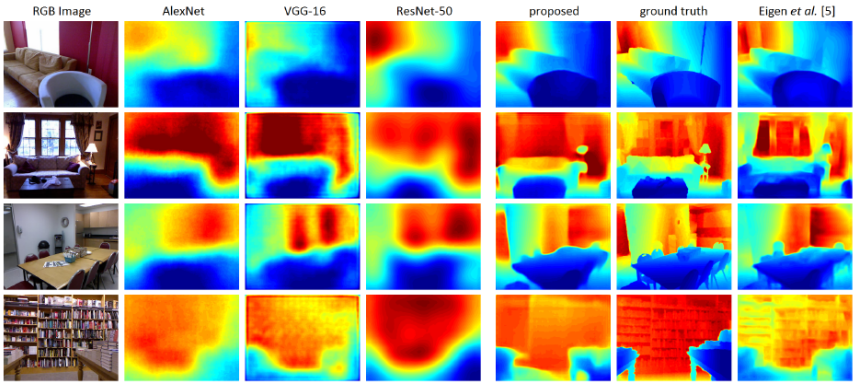
Figure 2.3: Initial position of the drone in the simulation environment, where the goal is in front of the agent, to reach it needs to avoid a obstacle.

The drone is equipped with an RGB camera, but it does not have a depth camera. To mimic the capabilities of the simulated drone, we use the algorithm of Deeper Depth prediction with fully convolutional residual network (Laina et al. (2016)) to estimate the depth map of a scene given a single RGB image, it runs in real-time on images or videos. In the evaluation, the model contains fewer parameters and requires fewer training data while outperforming the depth and time estimation.

,

Figure 2.4: Tello DJI drone uses for testing the algorithm in the real world.

Examples of their results are shown in Figure 2.5,



,

Figure 2.5: Examples of Depth map estimation from the Deeper Depth prediction with fully convolutional residual network. In the original article, they compare the performance of the network with others in the state of the art, the input is an RGB image and the output is the estimation of the depth map.

2.10 Summary

This chapter provides the principal aspects needed to understand the algorithm proposed in this work, as well as the tools needed to prove their functionality in simulated and real-world environments.

CHAPTER 3

RELATED WORK

This chapter provides a general overview of the state of the art in the field of Deep Reinforcement learning with drones, Causal Modeling, and the combinations of both fields Causal Reinforcement Learning.

3.1 Deep reinforcement learning

The ability to navigate autonomously is an important problem of mobile robots, we will aboard the problem in Unmanned aerial vehicles (UAVs or drones). Since the rise of the use of drones in different fields, many strategies with Deep RL have been used to develop an autonomous pilot for different tasks, independently of the task the ability to fly without collision with obstacles and achieve goals is an important aspect of it, Darwish and Nakhmani (2023) utilize a self-supervised model-free DRL approach using only a forward-facing depth-RGB camera, outperforms state-of-the-art deep Q-network algorithms in terms of learning policies and effectively intercept targets specified as a set coordinates. Shin et al. (2019) present a comprehensive study in which a drone adeptly navigates through a variety of 3D obstacles by plotting a path within a realistically simulated 3D landscape, employing deep reinforcement learning techniques. The study compares the racing performance of

the drone against human participants using a suite of deep RL algorithms, including Deep Q-Network (DQN), Double DQN, Dueling DQN, and Double Dueling DQN (DD-DQN). The drone utilizes two primary inputs: an RGB camera providing a first-person view of the landscape, and a depth map that offers detailed 3D environmental data. This dual-input system enhances the drone's navigational capabilities, allowing for precise and efficient pathfinding in complex terrains. Çetin et al. (2019) proposes a deep reinforcement learning (DRL) architecture to make drones behave autonomously inside a suburb neighborhood environment, where the depth image, the distance to the geo-fence (a virtual barrier on the environment), angle to goal and elevation angle between the goal and the drone are part of the state. Another similar approach where the state is built from the depth image captured at each step by the stereo-camera of the drone and the heading towards the goal is used to train three different DRL algorithms based on Q-learning is proposed by Kersandt et al. (2018). Their principal problem in training the agent in the Reinforcement Learning is the time and amount of episodes and steps needed to reach a stable policy, in addition, at the same time a lot of information that could be useful is wasted, this information between the agent and their interactions with the environment at each episode could be used to build Causal Models that can help in the transfer of task, interpretability, and explanation of the policy.

3.2 Causal Modeling

In the other hand, Causal Modeling (CM) needs rigorous data collection and expert knowledge. In Machine Learning has been mostly used to generate explainability with the collected data, Shi et al. (2022) introduce a model for interpreting causal relationships in a temporal-spatial context, capturing the causal connections between consecutive observations and decisions made by an RL agent. Ho and Wang (2021) proposes a system that leverages Human-Centered AI for using explainable

knowledge to construct the ethical causality in a simulated game theory. In Heyn and Knauss (2022) uses structural causal models to make human insight in causal relations explicit for uses this knowledge during AI system development. The causal information is often used to bring interpretability, Çetin et al. (2023) use the causal information to explain why the agent realizes a specific action based on the generation of a saliency map to identify the critical regions in an input image which influences the predictions made by the DQN agent by evaluating the gradients of the model's output with respect to the image and scalar inputs. Diehl and Ramirez-Amaro (2022) learn a causal Bayesian network from simulation data to learn a cause-effect model of the environment, generating causal explanations based on the obtained model.

3.3 Causal Reinforcement Learning

As we mentioned before Causal Reinforcement Learning in mainly divided into two categories. In the first category based on the prior causal information, we can find the work of Feliciano-Avelino et al. (2021) which demonstrates a better performance even with partial and spurious relationships in the causal graphical model in Q-Learning algorithms in light switch control tasks. Zhu et al. (2023) use Deep Q-Learning in the Emotional Pendulum and Windy Pendulum tasks. Gonzalez-Soto et al. (2018) introduced a decision-making approach for agents operating in environments characterized by uncertainty and underlying causal dynamics. This approach enabled the agent to form and continually update beliefs about the causal environment based on interactive outcomes. In their experimental setup, it was presupposed that the agent had prior knowledge of the causal framework governing the environment.
For the second category, where the causal information could be learning of the policy, we found the work of Méndez-Molina et al. (2022) where a Causal Dynamic Bayesian Network is learned for each of the agent actions and uses those models to improve

the action selection process in the coffee-task. For the same author Méndez-Molina et al. (2023) developed a framework for simultaneously learning and using causal models to speed up the policy learning in the online Markov decision process (MDP) proved in the Taxi-driver and Taxi Atari task. Pitis et al. (2022) applies a learned locally factored dynamics model to an augmented distribution of states and actions to generate counterfactual transitions for RL, and is used to train an offline RL agent to solve an out-of-distribution robotics manipulation task.

3.4 Summary

In this review of the state of the art, we can observe how the use of Causal knowledge can help deal with some problems inherent in the Reinforcement Learning, but the knowledge of the task or environment is not easy to obtain, when it's defined by an expert, its necessary to be careful that it is not too specific that it prevents its generalization. While learning causal knowledge online has been proven in different tasks, not been proven yet in a robotic task like visual navigation, due to this we propose a method to learn the causal model of the environment with a partial continuous environment.

CHAPTER 4

METHODOLOGY

Following the idea of the second category of Causal Reinforcement Learning where the relative causal information has to be learned for the policy, we will build the basic idea shown in the diagram of Figure 4.1, where the collection of the data recollected by the agent will be used to learn the Causal Bayesian Network to use it in the action selection in the context of Reinforcement Learning. In this chapter, we will describe the algorithm of Deep Q-learning (DQN) a Reinforcement Learning algorithm chosen to implement our work, the state representation used in the DQN state space, the definition of the Causal Bayesian Network, their learning and use in DQN.

4.1 Deep Q-Learning

We select Deep Q-Learning (DQN) as the algorithm of DRL to implement the visual navigation in the drone. As previously described, it is a reinforcement learning algorithm that combines Q-Learning with deep neural networks. The key idea behind Q-Learning is to estimate the value of each state-action pair (Q-value) in order to make optimal decisions. The action with the highest Q-value output is selected based on the ϵ-greedy method. The algorithm is described in Algorithm 1, the

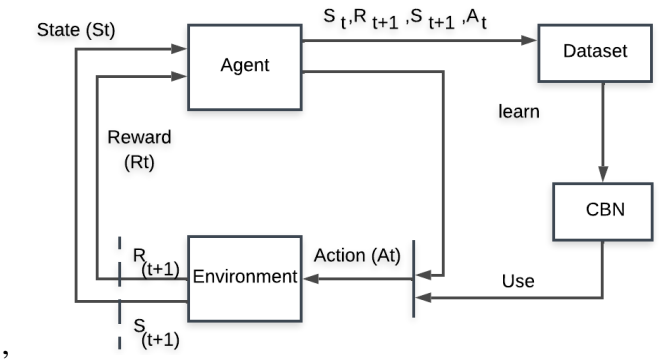


Figure 4.1: Diagram for the methodology following in the learning and use of the Causal Bayesian Network in the Reinforcement Learning context. The elements added are the Dataset and CBN blocks.

modifications on DQN are lines 7-9, where the Causal Bayesian Network is learned, and lines 10 and 18, where the Causal Bayesian Network is used, both are described later in this chapter.

4.2 State Representation

To represent the information needed for the construction of the Causal Bayesian Network, we decided to use 9 values: distance of five defined sections of the image (center, top left, top right, bottom left, and bottom right), a boolean value if the goal is in the field of view, distance and angle to the goal, and altitude of the drone. To obtain these values we need three sensors: an RGB camera to process the image to detect the goal, a Depth Camera to obtain the distances, and a barometer or similar to obtain the altitude of the drone.

The values for the distances are obtained by dividing the image into five sections, in each section we select a number specific of random points (is recommended at least 20 points), taking the smallest distance as the overall distance of the drone from the objects in this section. The sections are shown in Figure 4.2.

Algorithm 1 Deep Q-Learning Algorithm

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

for episode $= 1, M$ **do**

    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

    for $t = 1, T$ **do**

        if step % k == 0 **then**

            Learn the Causal Bayesian Network

        Set the probability of each action to reach a negative and positive reward with the evidence of the state t

        rand = random number

        if rand < explorationRate **then**

            action a_t = random action

        else

            action a_t = MaxIndex(qValues)

            Use of the Causal Bayesian Network

        Execute action a_t in emulator and observe reward r_t and image x_{t+1}

        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

        Set $y_j = r_j$ if episode terminates at step $j + 1$

        otherwise set $y_j = r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-})$

        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to θ

        Every C steps reset $\hat{Q} = Q$ by setting $\theta^{-} = \theta$

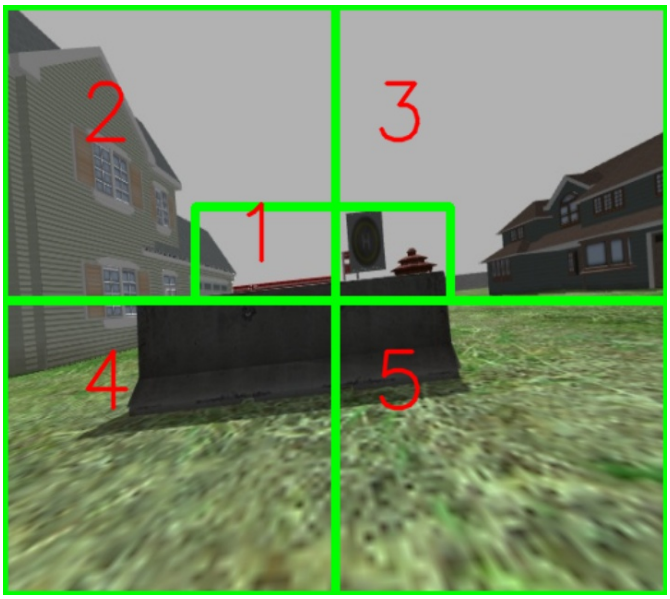


Figure 4.2: Sections used to obtain distances, in order: 1 (center), 2 (top left), 3 (top right), 4 (bottom left), 5 (bottom right).

The goal is an object defined previously, for this case, we select the signal of "Heliport" shown in Figure 4.5. Usually, in DRL algorithms the goal is defined with a set of coordinates, but we try not to depend on the agent seeing the goal all the time, and in real cases, access to the real coordinates requires the help of GPS or other sensors more difficult to access.

To enable the drone to recognize the goal, we trained a personalized YOLOv8 model (Redmon et al. (2016)). YOLO (You Only Look Once) is a popular series of object detection algorithms known for their speed and efficiency and permits training a new model from scratch to detect a specific series of objects. For goal recognition, we created a dataset combining images of the goal in the simulation environment labeled manually and images from the real world previously labeled of a dataset from the Roboflow's platform (Demiral (2023)) achieving 1070 images, examples of the dataset are shown in the Figure 4.3 and example of the testing results are shown in 4.4.

We trained the model for 50 epochs and obtained a precision of 0.943 and a

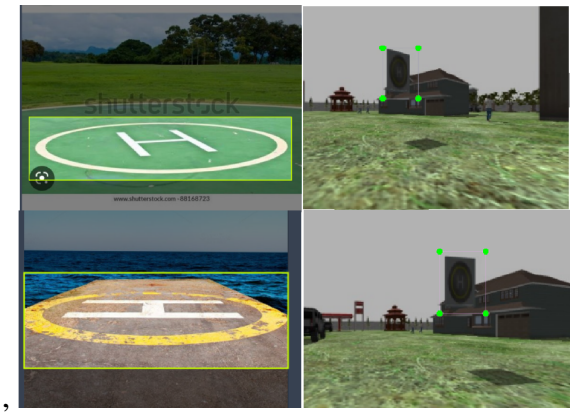


Figure 4.3: Examples of the dataset used to recognize the heliport signal, on the left side are the images for the real world, and the right side are the images taken in the simulated environment.

recall of 0.902 for goal detection. The YOLO model returns a list of detected objects, and each detection includes:

- Bounding Box Coordinates: these are the coordinates of the bounding box that encloses a detected object.
- Score: the confidence score of the detection, representing the probability that the detected object belongs to a particular class. It reflects the model's confidence in its prediction.

We put a threshold from the confidence score of the detection of 0.7, if the confidence score is bigger than the threshold value, the boolean value of the goal is set to True. Once the model detects the object and the bounding box coordinates are obtained the distance is calculated in the same way that the sections: with the depth image taking the bounded box of the goal to calculate the distance of the goal. We also establish an angle between the center of the image and the center of the goal (calculated with the bounding box coordinates), we can observe this in Figure 4.6. These values will be important to evaluate the reward function in the DRL algorithm.

Figure 4.4: Testing results of the YOLOV8 model to detect the goal in simulated and real-world scenarios after 50 epochs

Figure 4.5: An example of the goal that is used in the simulation environment.

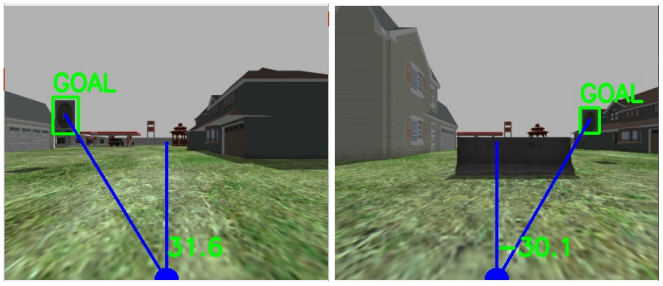


Figure 4.6: Measurement of angle and distance to the goal when is detected with the YOLOv8 model.

4.3 Causal Bayesian Network

For experimental purposes we construct the Causal Bayesian Network of the task to use in the reinforcement learning algorithm, taking into account the 9 values obtained from the RGB and depth image. To simplify and reduce the observation space we discretized the values as follows.

Distances from 5 sections (center, top right, top left, bottom right, bottom left) and to the goal:

$$\text{distance} = \begin{cases} \text{close} & \text{if } x \leq 0.5 \\ \text{far} & \text{if } x > 0.5 \end{cases}$$

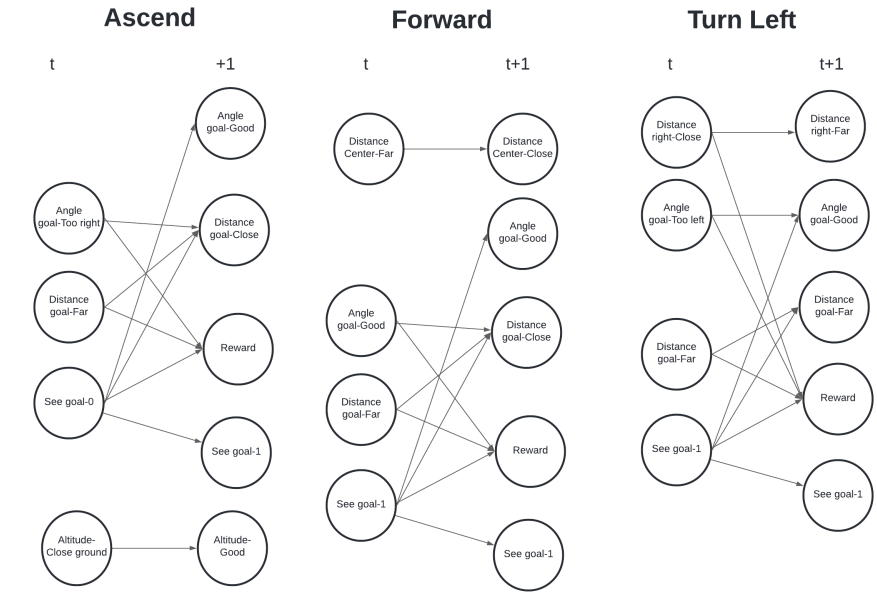
$$\text{goal in sight} = \begin{cases} 1 & \text{if the goal is recognized for the camera} \\ 0 & \text{the goal is not recognized for the camera} \end{cases}$$

$$\text{Angle to the goal} = \begin{cases} \text{good} & \text{if } -15 \leq x \leq 15 \\ \text{far left} & \text{if } x > 15 \\ \text{far right} & \text{if } x < -15 \end{cases}$$

$$\text{Altitude} = \begin{cases} \text{good} & \text{if } 0.5 \leq x \leq 1.5 \\ \text{close ground} & \text{if } x < 0.5 \\ \text{far ground} & \text{if } x > 1.5 \end{cases}$$

We first create manually an initial approximation of the Causal Bayesian Network to test how to incorporate it into the system. The learning of this CBN is part of the algorithms described later. We created a CBN for each action of the action set, we show the Causal Bayesian Network for the action ascend, forward, and turn left in Figure 4.7, the other CBNs for the missing actions can be found in the Appendix A.1. For simplicity, we do not show the variables in the state that do not change with the action. In all cases, knowing if the goal is in sight is important for the reward value and consequently is important to know the distance and the angle to the goal. When the goal is not in sight, the objective is to avoid obstacles,

in this case, the distance of the sections in the image is important to decide which action to take.



,

Figure 4.7: CBNs relating relational state variables at consecutive time steps (t) and (t + 1) for actions ascend, forward, and turn left showing the discrete value of the state (t) with higher probability to reach the state (t+1).

4.4 Learning Causal Bayesian Network

In the second part of the experiments, we learn the structure of the Bayesian Network while training the agent in DRL. The learning of the structure is done with the Hill Climb Search algorithm and the BIC score. Once we have the structure with the directed graph, we need to learn the parameters, for this, we calculate the Markov blanket of the reward node and delete synchronous links in the graph. To perform the inference with the observable state, we use the Variable elimination algorithm from Koller and Friedman (2009). The learning of the CBN is described in Algorithm 2, where k is a predefined number of steps to update the Causal Bayesian Network,

and $numActions$ is the number of actions in the actions set.

Algorithm 2 Learning Causal Bayesian Network

```
1:  At each step in the QDN algorithm save the state representation at time t, t+1, reward,
    and accumulated reward in the variable data
2:  if step % k == 0 then
3:      for i = 0 to numActions do
4:          Check for the previous model
5:          if there is a previous model then
6:              Start with a pre-defined structure for the Bayesian network
7:          else
8:              Start with a random initial structure for the Bayesian network
9:          model[i]= HillClimbSearch(BIC score, data)
10:         Delete synchronous links
11:         Calculate the Markov blanket for the node Reward
12:         Create/update CPTs
13:         inference[i] = VariableElimination(model[i])
14:
```

4.5 Use the CBN for DQN

If the CBN model is already known, it can be utilized from the onset of the learning process. However, when the CBN needs to be learned, the reinforcement learning algorithm must first accumulate sufficient data over k steps. This is necessary to ensure that the data collected is adequate for learning an accurate approximation of the true CBN.

Algorithm 3 describes how to use the CBN within DQN. At each step, they take the state elements present in the Markov blanket of the node reward like *filtered*

evidence, which is used in the inference of reaching each state value, we divide this inference into two arrays, one to the probability to transition to a positive reward *ProbAct* and *ProbActNeg* with their complement, the probability to transition to a negative reward. The lines 11-14 describes the normal behavior of DQN, after that depending of the probability of consulting the Causal Bayesian Network *ProbCausalModel* (lines 16-27) the model consults if the action selection made by the Q-Values have a high probability (with a threshold *ThresholdProb*) to reach a negative reward, if it is greater than the threshold then the action is eliminated like an option and another is selected randomly (lines 17-22), if not we consult if have a high probability (with the same threshold) to obtain a positive reward, when the probability is greater than the threshold the action is executed, if not, the action with higher probability to transition to a positive reward is selected (lines 23-25). ProbCausalModel and ThresholdProb are previously predefined like the exploration rate of the DRL algorithm.

Because the query of the probability of a positive and negative reward is after the comparison of the learning rate, benefited more from random action than the consult of the CBN, we define a variant of the Algorithm in 4 replacing from line 10 of the previous algorithm. In this case, the probability of consulting the CBN is first before the behavior of the normal DQN, once the model consults the CBN if the probability of the positive reward is less than the threshold we can consult the probability of the negative reward and eliminate from the possible actions, resorting to the random action only if the probability of obtaining a negative reward is greater than the threshold.

4.6 Summary

In the chapter, we have described the methodology to achieve our objectives. Our main contributions are: (i) the development of an algorithm that simultaneously

Algorithm 3 Use of the Causal Bayesian Network for DQN

```
At each step
initialize Evidence of the state t
for i = 0 to numActions do
    filter evidence of the Markov blanket of the node reward
    result = inference[i] for variable reward with filtered evidence
    ProbActNeg[i]=Probability of the action to obtain a negative reward
    ProbAct[i]=Probability of the action to obtain a positive reward
initialize ProbCausalModel and ThresholdProb
rand = random number
if rand < explorationRate then
    action a_t = random action
else
    action a_t = MaxIndex(qValues)
    rand = random number
    if rand < ProbCausalModel then
        if ProbActNeg[action a_t] > ThresholdProb then
            ActionNegative = action
            while action a_t == ActionNegative do
                action a_t = random action
        else
            if ProbAct[action a_t] < ThresholdProb then
                action a_t = MaxIndex(ProbAct)
```

Algorithm 4 Use of the Causal Bayesian Network for DQN second version

```
At each step
rand = random number
if rand < ProbCausalModel then
    action a_t = MaxIndex(qValues)
    if ProbAct[action a_t] < ThresholdProb then
        if ProbActNeg[action a_t] > ThresholdProb then
            ActionNegative = action
            while action a_t == ActionNegative do
                action a_t = random action
else
    rand = random number
    if rand < explorationRate then
        action a_t = random action
    else
        action a_t = MaxIndex(qValues)
```

learn a policy, using a DRL algorithm, and a Causal Bayesian Network for each action, (ii) the integration of a CBN into the learning process of DQN to guide and constrain the possible actions of the RL in two variants, and (iii) the use of this approach for autonomous navigation task of UAVs, as it will be shown in the following chapter.

CHAPTER 5

EXPERIMENTS AND RESULTS

In this chapter, we described the experiment setting done (i) in the DQN algorithm, (ii) to find the better value of k in the Learning Causal Bayesian Network and (iii) to prove the effectiveness of our algorithm proposed in the DQN training of 700 episodes with image representation, CBN defined, CBN-DQN PN, CBN-DQN P and CBN-DQN V2. We define the probability of consulting the causal model with a value of 0.7, and the threshold of the probability with 0.75. In the last part, we analyze the results comparing the CBN learned in the training to the defined, as well as the evaluation of the CBN learned through training.

5.1 Deep Q-Learning Setting

For the implementation of the DQN to visual navigation in the drone, we followed the implementation of Anas et al. (2022) and adapted it to our task and mobile robot (drone).

State space: The agent's state spaces consist of:

1. For the base experiment it consists of an image with dimensions 84x84x3 pixels, this is to evaluate the performance of the base algorithm without causal models.

2. For the learning and use of the CBN consists of an image with dimensions 84x84x3 pixels + 9 values of the state.

Action space: the agent can take eight discrete actions namely: *move forward, move backward, turn left, turn right, right, left, ascend* and *descend*. The actions *move forward* and *move backward* move the drone in the y axis, the actions *ascend* and *descend* move the drone in the z axis, the actions *right* and *left* move the drone in the x axis, and the actions *turn right* and *turn left* only change the orientation of the drone in the x axis.

Target: a goal image.

Reward design: calculated from the following formula.

$$\text{reward} = \begin{cases} 100 - \text{Distance to goal} - |\text{angle to goal}| & \text{if the target is recognized} \\ -9 & \text{otherwise} \end{cases}$$

Consists of a negative reward at each step that the goal is not recognized, when the goal is recognized obtain a reward of 100 minus the distance, and the angle to encourage these to be the smallest possible. Plus a positive reward of 1000 if reach the goal or a negative reward of -1000 to crash.

Termination conditions: 1) The agent reaches the goal within 0.2m of distance. 2) The agent collides with an obstacle. 3) The number of steps exceeds 200.

Network Architecture: The Prediction Network and Target Network have the same structure, as shown in Figure 5.1.

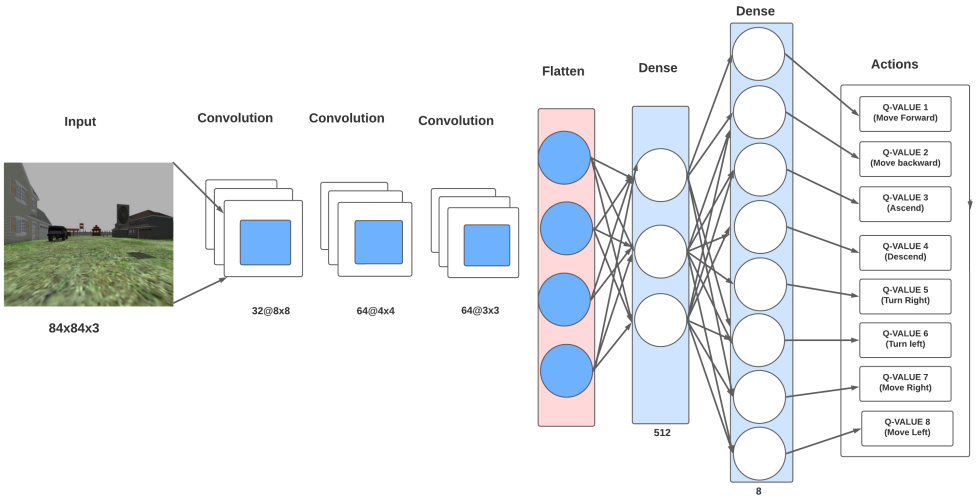
- Input layer: takes a 2D matrix with dimensions 84x84x3 corresponding to an RGB image of the environment.
- Convolutional Layers: all with activation function ReLU. Layer 1: Convolution 2D with 32 filters, a kernel size of 8x8, a stride of 4. Layer 2: Convolution 2D with 64 filters, a kernel size of 4x4, a stride of 2. Layer 3: Convolution 2D with 64 filters, a kernel size of 3x3, a stride of 1.

- Flatten layer: Flattens the output of the convolutional layers into a 1D vector. This is often done before transitioning to fully connected layers
- Dense layer: Dense layer with 512 units, this layer processes the flattened features from the convolutional layers.
- Output layer: dense layer with a size of 8 nodes which are the eight Q-values for the eight available actions that can be taken.
- Loss function: Mean Squared Error (MSE).
- Optimizer: RMSprop with a learning rate of 0.00025, a discount factor of 0.9, and an epsilon value of 0.00000001.

For the experiments on the learning and use of the CBN, we concatenate the 9 discretized values before the dense layer (Figure 5.2) and use these values to learn the Causal Bayesian Network. We use these values to represent the state to help the network to identify aspects of the environments that are not explicit in the image, like the distances obtained from the depth image and the altitude, these information is important for avoiding collision and the perception of the goal.

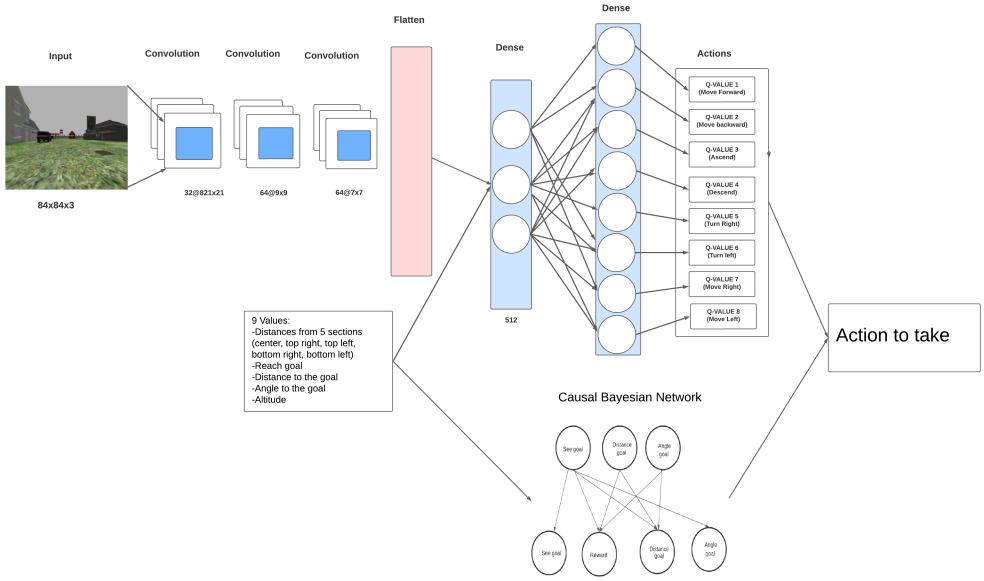
5.2 Experiments and Results

We begin our experiments by using the state representation for learning and using the CBN to determine the optimal value for k. The parameter k dictates the point at which the CBN learning algorithm starts its learning process. We evaluated its effectiveness across various k values: 100, 200, 400, and 600. Their performance is illustrated in Figure 5.3, the vertical axis represents the reward obtained in each episode after applying a moving average of size 20 to smooth out short-term fluctuations and highlight longer-term trends, rewards range from below -1000 up to approximately 1500. The horizontal axis tracks the number of episodes, ranging



,

Figure 5.1: Prediction and Target Network used in the DQN algorithm with a single image like input and eight values for the output, which correspond to the qValue of each action.



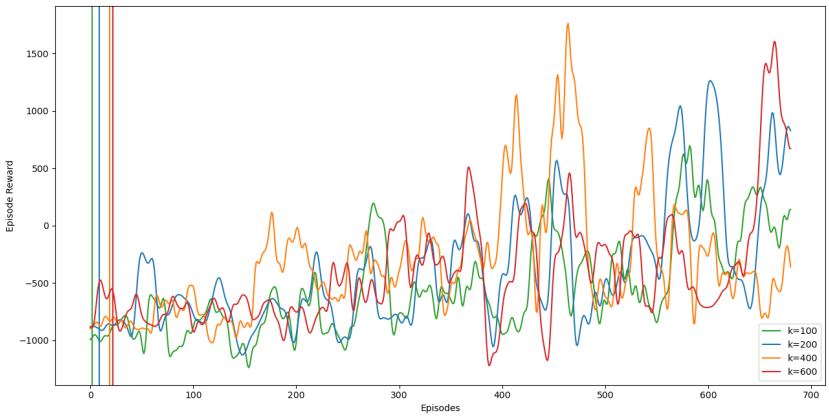
,

Figure 5.2: Prediction and Target Network used in the DQN algorithm of the second experiment, with the image and the 9 values of the state representation like input.

from 0 to 700. Here, k is critical as it defines the number of steps necessary to update the CBM, before the first k steps the behavior algorithm is the DQN, after k steps the algorithm has the first version of the CBN that can be consulted, with

little k the algorithm have a CBN from very early episodes which is updated each k steps.

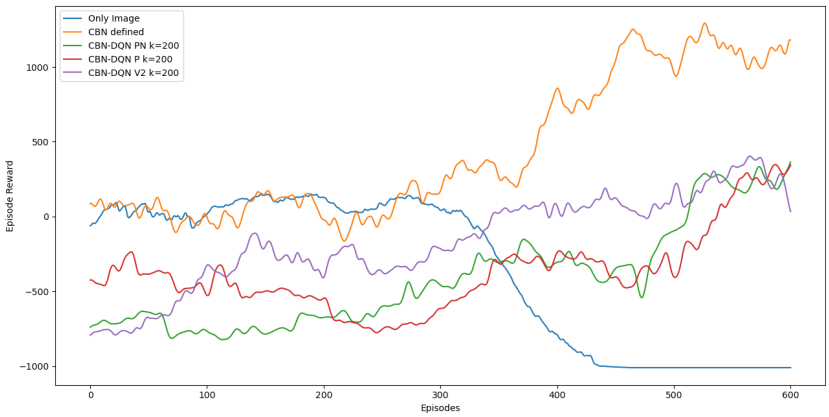
k=100 presents the worst performance as it is unable to reach high rewards, k=200 shows a mix of moderate and high peaks and seems to perform somewhat consistently in the mid-range, while k=400 and k=600 show more pronounced fluctuations with very high peaks suggesting periods of high rewards but also significant drops. Lower k values seem to have more stability compared to higher k values (remember that after k steps the CBN is updated), which exhibit more dramatic ups and downs. This could indicate that lower k values provide a more consistent learning experience or adaptation, whereas higher k values, while capable of achieving higher peaks, might introduce volatility. This may also be due to the number of episodes used in training, needing to carry out longer experiments but due to time constraints, we will take these as a starting point. For this reason, we select the value of 200 for k to compare with the other algorithms.



,

Figure 5.3: Episode reward obtained after 700 episodes in the learning and use of the CBN in the DQN algorithm with k equals to 100, 200, 400, and 600. The vertical lines indicate from which episode the CBN is learned and subsequently used and updated. We utilize a moving average of size 20.

To evaluate the performance of our algorithm, we compared the DQN algorithm as baseline (only image) with the proposed algorithm that adds the discrete state representation, using: (i) the defined Causal Bayesian Network (ii) learning the CBN (CBN-DQN P) (iii) a variant of the algorithm considering the probability of receiving a positive and negative reward from the CBN to decide which actions to take versus considering only the probability of positive rewards for the first version (CBN-DQN PN) and (iv) learning the CBN in the second version of the algorithm (CBN-DQN V2), all with k=200



,

Figure 5.4: Episode reward obtained after 700 episodes with base DQN, DQN with the CBN defined, CBN-DQN PN, CBN-DQN P, and CBN-DQN V2 with k=200, with a moving average of 50.

Their performance is illustrated in Figure 5.4, the vertical axis represents the reward obtained in each episode after applying a moving average of size 50. The best behavior is when the Causal Bayes Network is manually defined in advance because it is used from the beginning of the learning process and the CBN is more accurate. Using only the image as state representation has not adequate information to reach the goal and the network learns to "do nothing" to obtain a better reward compared to realizing actions without reaching the goal. In comparison, the CBN-DQN V2 is the closest to the behavior of the BCN defined compared to the use only the

action with a higher probability of carrying a positive reward (CBN-DQN P) and using both (CBN-DQN PN), this could be because the first version of the algorithm benefits more from performing random actions at the first episodes depending of the exploration rate, so might it requires more training episodes for better behavior compare to the version two of the algorithm where the probability of consulting the CBN is more dominated that the exploration rate.

Once we have the behavior of the algorithm, we can compare the number of common edges in the CBN learned at the beginning and the end of the training of the CNB-DQN V2 algorithm against the Causal Bayesian Network defined for each action, this can be observed in the Table 5.1. As can be observed the number of total edges at the beginning and in the end increases for all the CBNs, at the same the number of edges in common with the CBN defined, still, it's not a good number of edges in common, this can be because we want to build the CBN to obtain the best reward from only successful episodes which are not guaranteed when the algorithm is learning, especially in the first epochs of the algorithm, as a result, the CBN finds connections we didn't expect because of the mistakes it makes but they are useful for learning the CBN, our error is the possible the early network learning when the data for a good reward is not enough to make connections to our node of interest.

As additional information we can analyze the behavior of the CBN learning, we can observe the evolution of the CBN for CBN-DQN V2, for the action forward in Figure 5.5 they begin with 4 connections between the time t and time $t+1$ but for the end, it has 7, some interesting to note is the dependence between the reward that depended of angle for the same $t+1$ but this depends of see goal and angle goal for the previous time, when we define this relation with reward more directly. Another example is for action right in Figure 5.6 when begins only with 2 connections and ends with 8, in this case, the relation for the previous actions is repeated, here the altitude takes more relevance with more connections with the distance of the sections

Action	Total Edges given CBN	Edges in common with the first CBN learning /Total of edges	Edges in common with the last CBN learning /Total of edges	Percentage of best similarity
Ascend	14	1/9	2/12	0.14
Descend	14	2/8	2/14	0.14
Forward	14	1/7	2/13	0.14
Backward	14	1/9	2/12	0.14
Left	17	2/9	1/16	0.05
Right	17	2/7	2/13	0.117
Rotate left	16	2/9	3/15	0.187
Rotate right	17	1/8	3/13	0.176

Table 5.1: Comparison between the number of common edges in the CBN defined, the first and the last obtained in the training CBN-DQN V2 with the total of edges for each action and their percentage of similarity.

4, see the goal and the previous altitude. This behavior between the learning Causal Bayesian Networks helps us to note more dependencies that we don't take into account in the definition of the network and the lack of direct connections to the reward node can indicate there is not sufficient information relevant that involves seeing the goal and consequently to the angle and distance to the goal. The learning Causal Bayesian Network for the other's actions can be found in the Appendix A.2.

From the experiments we can conclude the following:

- A Causal Bayesian Network (given or learned) can be used to improve the performance of a reinforcement learning agent in robotic tasks with a partial continuous space state
- A Causal Bayesian Network can be learned from data obtained during the

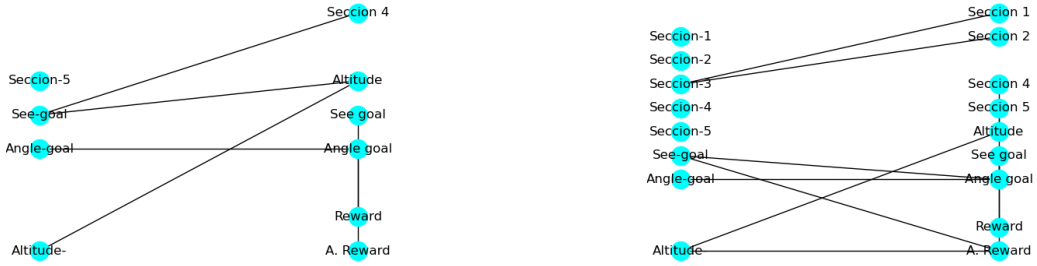


Figure 5.5: Comparison of the first and last Causal Bayesian Network for forward

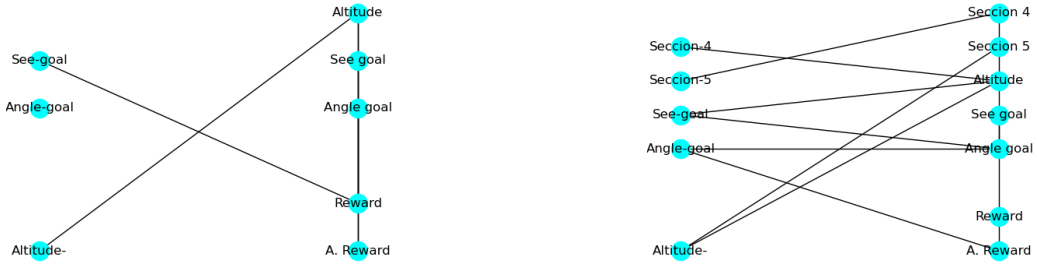


Figure 5.6: Comparison of the first and last Causal Bayesian Network for right

reinforcement learning process but needs more episodes to obtain a good model

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this work, we build two algorithms: one that learns a Causal Bayesian Network of the task and environment during the training of the DQN algorithm and another one (with two variants) that uses a CBN to guide the selection of actions in the DQN algorithm. We tested the algorithms in the task of autonomous navigation of a drone in a simulated environment. The results show that it is possible to learn partial CBNs, while learning a policy, but needs more episodes to reach a good number of common edges with the defined CBN and stability.

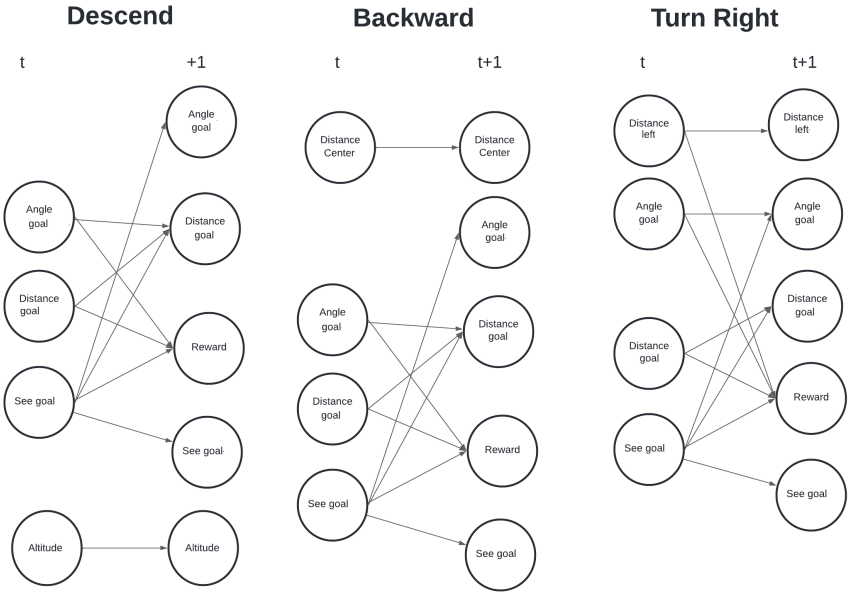
6.2 Future Work

During the development of this research, we struggle to have access to modern computers with powerful GPUs. As future work, we need to train our algorithms for more episodes to prove the stability of the algorithm and know if the Causal Bayesian Network reaches the accuracy of the model with the network previously defined. We also need to perform more tests on different environments to assess the generality of the proposed approach. We would also like to include an exploration strategy, in

case the objective is out of sight. Finally, we believe that the learned CBNs can be transferred to other, although similar, domains where the causal relationships are still valid. We would like to test this in other drone tasks.

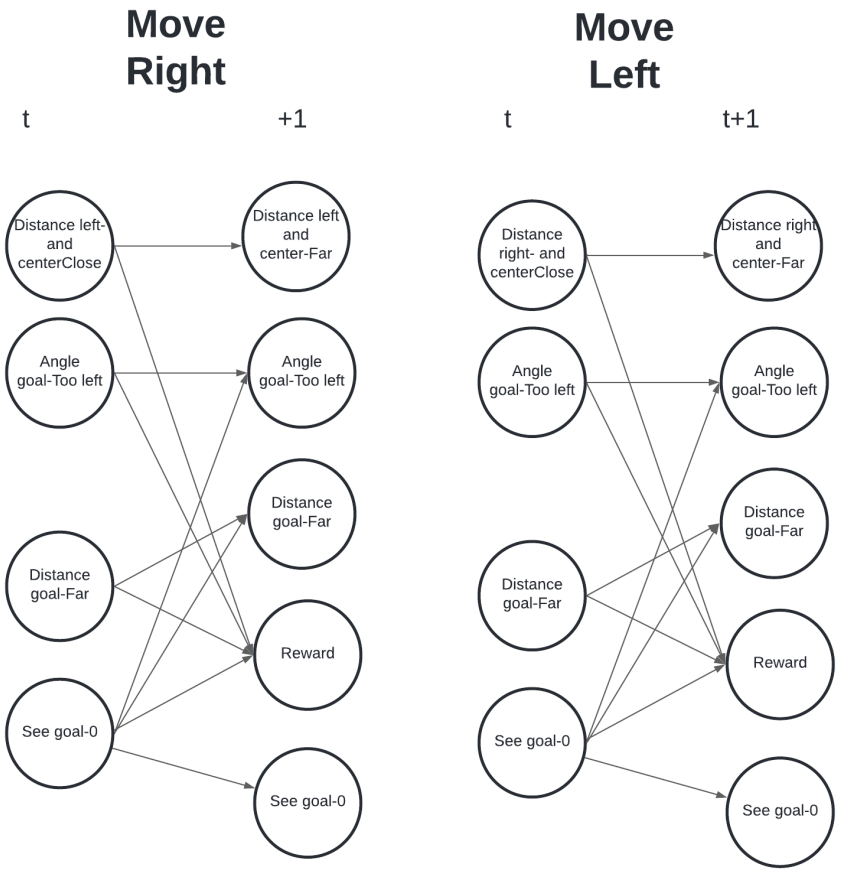
APPENDIX

A.1 Defined Causal Bayesian Network



,

Figure A.1: CBNs relating relational state variables at consecutive time steps (t) and (t + 1) for actions descend, backward, and turn right showing the discrete value of the state (t) with higher probability to reach the state (t+1).



,

Figure A.2: CBNs relating relational state variables at consecutive time steps (t) and (t + 1) for actions move right and move left showing the discrete value of the state (t) with higher probability to reach the state (t+1).

A.2 Learning Causal Bayesian Network

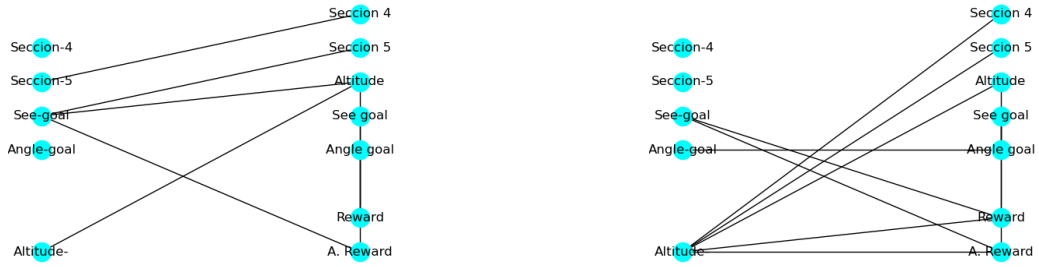


Figure A.3: Comparison of the first and last Causal Bayesian Network for ascending action, in this network we can note something interesting between the altitude and the values for the distance of sections 4 and 5, because these sections can indicate the distance between the ground, aspects important when the ascend action is taking.

Figure A.4: Comparison of the first and last Causal Bayesian Network for descending action, compared to their action complement (ascend) in this case the altitude is more important for the reward taking into account when the altitude is too near to the ground is consideration like the drone crash.

Figure A.5: Comparison of the first and last Causal Bayesian Network for backward action

Figure A.6: Comparison of the first and last Causal Bayesian Network for left action

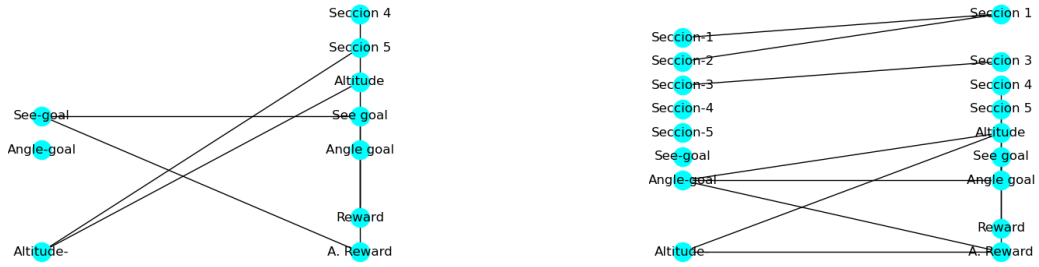


Figure A.7: Comparison of the first and last Causal Bayesian Network for rotate left action

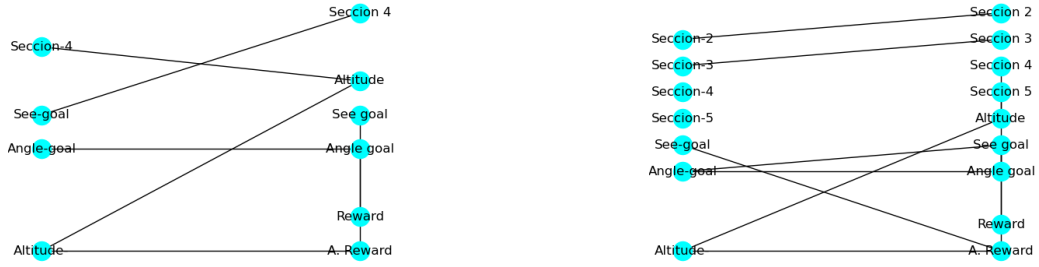


Figure A.8: Comparison of the first and last Causal Bayesian Network for rotate right action

Bibliography

Anas, H., Ong, W. H., and Malik, O. A. (2022). Comparison of deep q-learning, q-learning and sarsa reinforced learning for robot local navigation. In Kim, J., Englot, B., Park, H.-W., Choi, H.-L., Myung, H., Kim, J., and Kim, J.-H., editors, *Robot Intelligence Technology and Applications 6*, pages 443–454, Cham. Springer International Publishing.

Asher, H. (1976). *Causal Modeling*. Sage University papers. SAGE Publications, 2 edition.

Balasingam, M. (2017). Drones in medicine-the rise of the machines. *International Journal of Clinical Practice*, 71(9):e12989.

Çetin, E., Barrado, C., Muñoz, G., Macias, M., and Pastor, E. (2019). Drone navigation and avoidance of obstacles through deep reinforcement learning. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–7.

Çetin, E., Barrado, C., and Pastor, E. (2023). Explainability of deep reinforcement learning method with drones. In *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, pages 1–9.

Darwish, A. A. and Nakhmani, A. (2023). Drone navigation and target interception

using deep reinforcement learning: A cascade reward approach. *IEEE Journal of Indoor and Seamless Positioning and Navigation*, 1:130–140.

Demiral, I. (2023). H letter2 dataset. https://universe.roboflow.com/ilknur-demiral-jt5w4/h-letter2. visited on 2024-06-10.

Deng, Z., Jiang, J., Long, G., and Zhang, C. (2023). Causal reinforcement learning: A survey. *Transactions on Machine Learning Research*. Survey Certification.

Diehl, M. and Ramirez-Amaro, K. (2022). Why did i fail? a causal-based method to find explanations for robot failures. *IEEE Robotics and Automation Letters*, 7(4):8925–8932.

DJI (2018). Tello dji: Programmable mini drone for kids and adults. https://www.dji.com/tello.

Feliciano-Avelino, I., Méndez-Molina, A., Morales, E. F., and Sucar, L. E. (2021). Causal based action selection policy for reinforcement learning. In Batyrshin, I., Gelbukh, A., and Sidorov, G., editors, *Advances in Computational Intelligence*, pages 213–227, Cham. Springer International Publishing.

Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. (2016). *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham.

García, M., Caballero, R., González, F., Viguria, A., and Ollero, A. (2020). Autonomous drone with ability to track and capture an aerial target. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 32–40.

Garg, P. (2021). *Unmanned Aerial Vehicles: An Introduction*. Mercury Learning and Information.

Gatteschi, V., Lamberti, F., Paravati, G., Sanna, A., Demartini, C., Lisanti, A., and Venezia, G. (2015). New frontiers of delivery services using drones: A prototype system exploiting a quadcopter for autonomous drug shipments. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 920–927.

Gonzalez-Soto, M., Sucar, L. E., and Escalante, H. J. (2018). Playing against nature: causal discovery for decision making under uncertainty.

Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Penicka, R., Song, Y., Cioffi, G., Kaufmann, E., and Scaramuzza, D. (2024). Autonomous drone racing: A survey. *IEEE Transactions on Robotics*, 40:3044–3067.

Hasan, K. M., Abdullah-Al-Nahid, Alim, M. A., Maniruzzaman, M., Atiqur Rahaman, G. M., Ahsan, M. S., and Newaz, S. S. (2020). Design and development of an aircraft type multi-functional autonomous drone. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 734–737.

Heyn, H.-M. and Knauss, E. (2022). Structural causal models as boundary objects in ai system development. In *2022 IEEE/ACM 1st International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 43–45.

Ho, J. and Wang, C.-M. (2021). Human-centered ai using ethical causality and learning representation for multi-agent deep reinforcement learning. In *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*, pages 1–6.

Kersandt, K., Muñoz, G., and Barrado, C. (2018). Self-training by reinforcement learning for full-autonomous drones of the future. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–10.

Klecka, W. (1980). *Discriminant Analysis*. Quantitative Applications in the Social Sciences. SAGE Publications.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., and Navab, N. (2016). Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE.

Lewis-Beck, M. (1980). *Applied Regression*. SAGE Publications, Inc.

Management Association, I. (2019). *Unmanned Aerial Vehicles: Breakthroughs in Research and Practice: Breakthroughs in Research and Practice*. IGI Global.

Mario Silvagni, Andrea Tonoli, E. Z. and Chiaberge, M. (2017). Multipurpose uav for search and rescue operations in mountain avalanche events. *Geomatics, Natural Hazards and Risk*, 8(1):18–33.

Méndez-Molina, A., F. Morales, E., and Sucar, L. E. (2022). Causal discovery and reinforcement learning: A synergistic integration. In Salmerón, A. and Rumí, R., editors, *Proceedings of The 11th International Conference on Probabilistic Graphical Models*, volume 186 of *Proceedings of Machine Learning Research*, pages 421–432. PMLR.

Méndez-Molina, A., Morales, E. F., and Sucar, L. E. (2023). Carl: A synergistic framework for causal reinforcement learning. *IEEE Access*, 11:126462–126481.

Menouar, H., Guvenc, I., Akkaya, K., Uluagac, A. S., Kadri, A., and Tuncer, A. (2017). Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, 55(3):22–28.

Pitis, S., Creager, E., Mandlekar, A., and Garg, A. (2022). Mocoda: Model-based counterfactual data augmentation.

Qasim, A., El Refae, G. A., and Etter, S. E. (2022). Utilizing drones in long-term construction projects: the impact on cash flow management. In *2022 International Arab Conference on Information Technology (ACIT)*, pages 1–4.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Radoglou-Grammatikis, P., Sarigiannidis, P., Lagkas, T., and Moscholios, I. (2020). A compilation of uav applications for precision agriculture. *Computer Networks*, 172:107148.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788.

Sabry, F. (2023). *Hill Climbing: Fundamentals and Applications*. Artificial Intelligence. One Billion Knowledgeable.

Sewak, M. (2019). *Deep Reinforcement Learning*. Springer Singapore.

Shi, W., Huang, G., Song, S., and Wu, C. (2022). Temporal-spatial causal interpretations for vision-based reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10222–10235.

Shin, S.-Y., Kang, Y.-W., and Kim, Y.-G. (2019). Automatic drone navigation in realistic 3d landscapes using deep reinforcement learning. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1072–1077.

Sigaud, O. and Buffet, O. (2013). *Markov Decision Processes in Artificial Intelligence*. ISTE. Wiley.

Sucar, L. E. (2020). *Probabilistic graphical models*. Advances in computer vision and pattern recognition. Springer Nature, Cham, Switzerland, 2 edition.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Zeng, Y., Cai, R., Sun, F., Huang, L., and Hao, Z. (2023). A survey on causal reinforcement learning.

Zhai, G., Zhou, J., An, P., and Yang, X. (2019). *Digital TV and Multimedia Communication: 15th International Forum, IFTC 2018, Shanghai, China, September 20–21, 2018, Revised Selected Papers*. Communications in Computer and Information Science. Springer Nature Singapore.

Zhu, W., Yu, C., and Zhang, Q. (2023). Causal deep reinforcement learning using observational data.