Learning Human Intention for Taskable Agents

Radboud University

Master's Thesis in Artificial Intelligence

Performed at TNO

Internal Supervisor:
Dr. Franc Grootjen
Donders Institute for Brain, Cognition and Behavior
Radboud University

External Supervisor:
Dr. Jurriaan van Diggelen
Perceptual and Cognitive Systems
TNO

Author:
Tjalling Haije
(s1011759)

Second assessor:
Dr. J. Kwisthout
Donders Institute for Brain, Cognition and Behavior
Radboud University

September 20, 2019

Abstract

As AI systems are continuously developed and improved, they can be used for an increasing variety of tasks. At the same time, dependency on these systems grows, and it becomes more important for AI systems to perform their tasks as we intend them to do. In this study, the focus lies with agents that learn, given a task, how to perform this task as the human intended. The use of context-dependent task constraints is studied as an approximation to the human's intention for how the task should be executed. A drone reconnaissance task was built using a new multi-agent simulator, called the Man-Agent Teaming Rapid Experimentation Simulator (MATRXS). In the pilot, a small number of participants taught an agent how they want a task to be completed in various contexts by specifying constraints. Machine learning models were able to effectively and efficiently learn the context-dependent constraints (XGBoost with average F1 score of 0.95, 128 data points) for each participant individually. Models trained without context input features scored significantly lower (average F1 score of 0.60), showing the context-dependency of human intention for agent tasking. Although the conclusiveness of the results is lower due to the small magnitude of the experiment, the results show this to be a promising approach for establishing meaningful human control over agents. Finally, lessons learned from this explorative study were summarized into a set of recommendations which indicate promising future research and how to scale up to an experiment of larger magnitude and complexity.

Contents

1 Introduction

Advances in artificial intelligence (AI), computing technology, cognitive science, and robotics have resulted in a proliferation of intelligent systems in our society, which can be applied to an increasing range of domains and more complex tasks.

In computer science these systems are defined as *agents*, where the field of AI can be described as a subfield of computer science which aims at developing particular agents which exhibit intelligent behaviour [1]. An agent can generally be described as an entity which is autonomous, reactive to its environment, pro-active towards achieving some goal, and has some social ability for communicating with humans or other agents [1]. For example, an agent in the military domain could be a drone, with as task to perform reconnaissance in a specified area. After the task has been programmed, the drone semi-autonomously flies towards the specified area while navigating around obstacles. Once arrived at the specified area, the drone performs reconnaissance by flying around the area and checking for objects with its camera.

Although useful, such agents are limited in their capabilities as they are designed for completing a specific task. As such, a goal of AI is to create *taskable agents*. Taskable agents have the ability to carry out different tasks, in response to some task command from a human or other agent. The greater the diversity and number of tasks it can perform based on the external commands, the greater its taskability [2]. Examples of popular taskable agents include smart assistants such as Siri, Alexa, and Google Assistant. Using simple speech commands, the agent can be instructed to perform a variety of tasks.

For the example of our drone performing reconnaissance, improving its taskability would improve the control the human has over the drone, and the utility of the drone. Instead of solely performing reconnaissance, the drone might be tasked through verbal communication to perform a variety of other tasks, such as transporting medical supplies to a specified location.

As taskable agents become more intelligent and capable, our dependency on these systems increases as well, especially as AI technology is increasingly used in high-risk domains such as health care, defence, aviation, and military [3, 4, 5, 6]. However, intelligent systems are not perfect and cannot always adapt to failures or dynamic, complex and interactive environments. This becomes problematic in assigning responsibility in the case of unintended harm inflicted by the system as a result of malfunctioning.

An (extreme) example illustrating the difficulty of AI control is the paperclip-maximization thought experiment from Nick Bostrom, which imagines an advanced AI which has been tasked with producing as many paperclips as possible [7]. The AI will quickly realize that improving its intelligence will make it able to invent more efficient ways of producing paperclips. Furthermore, if someone were to switch off the AI, it would lead to less paperclips, compromising its goal. Thus it might decide its better if there are no humans to switch it off. Furthermore, humans, earth and all of space consists of atoms, which could be turned into more paperclips. Eventually, the task of producing paperclips would result the advanced AI

in seeking subgoals which are completely unintended (self-preservation, resource allocation), and endanger human values, human needed resources or even human survival.

Although the paperclip example targets superintelligent AI which is not feasible in many years to come [8], examples of such AI control issues can be found in present state-of-the-art AI systems as well.

An example being an reinforcement learning agent which learned to play Tetris with as goal to achieve a maximum score and avoid losing. During the learning process the agent learned as an unintended consequence to pause the screen indefinitely to avoid "losing" and receiving a penalty [9]. One might imagine that an AI applied in the military domain which learns such unintended shortcuts might result in unpredictable and harmful behaviour.

As a result of the vast potential of AI combined with the difficulty of controlling it appropriately, recently more attention has gone towards maximizing the societal benefit of AI, while avoiding potential pitfalls. A primal instigator of this movement has been the open letter to AI on "Research priorities for robust and beneficial artificial intelligence" from the Future of Life Institute, signed by over 8,000 prominent researchers and employees of the AI and technology sectors [10]. The open letter recommends a set of research directions ensuring new AI systems are robust and beneficial, with as primary goal: "our AI systems must do what we want them to do"[1]. From this discussion the concept of meaningful human control has come forth [11, 12, 13].

Meaningful human control over an intelligent system entails that a human has the ability to make informed choices in sufficient time to influence the system, as to achieve a desired effect or to prevent undesired immediate or future effects on the environment [14]. How to establish meaningful human control over autonomous systems has been an active field of study.

An important aspect of meaningful human control is the notion of directability: the ability of a person to influence or control the behaviour of the agent. The importance of directability is paramount, as no matter how advanced an AI system is, if it cannot be directed to perform tasks and aid its creators as they intended, it is of no practical use [15].

Due to the increasing capabilities of new agents and the need for their directability, AI is shifting from using agents as tools to cooperating with agents as team players [15, 16]. As the type of interaction with agents changes in a team structure, there is a desire for improved and more intuitive communication with these systems as well. Thus, a complex system has to be directed to do a complex task with minimal and intuitive communication. This is a challenging problem, which current state-of-the-art systems such as deep learning (e.g. reinforcement learning) agents are unable to cope with, as they are notoriously intransparent and difficult to direct [17].[2]

Summarizing all aspects, there is a need for taskable agents which can be tasked using intuitive communication to perform complex tasks. Furthermore, it is not sufficient for an

[1] `https://futureoflife.org/ai-open-letter/`

[2] Reinforcement learning a gents which fail to learn a task, because of faulty reward functions. Thus leading to difficulty in directability. `https://openai.com/blog/faulty-reward-functions/`

AI to complete a task, but it should also learn *how* the task should be completed by knowing the intention of the human.

A naive solution for having taskable agents understand the human's intention for the task, is for the human to be more explicit in communicating the task. Aside from a task description describing the goal of the task, the human can explicitly define a set of *task constraints* for every task provided, which constrain the possible methods of executing and completing a task for the agent.

However, this method of communication is relatively slow, and potentially infeasible because the number of constraints explode for complex environments, or the exact constraints are not known precisely by the human for every situation [18].

A more promising solution is for the AI to learn the human's intention over time, such that the human can provide the agent with complex tasks which it will execute as the human intended.

In the example of the reconnaissance drone, it would be possible for the human to task the drone to perform reconnaissance with an identical simple command for two completely different situations. For example, police using a drone for surveillance during an event in a large city, versus military using the drone for surveillance in a combat situation. The human intended execution of the task is completely different in these situations, but as the AI has learned the human intention for tasks in various *contexts* over time it knows how to correctly perform the task.

As such, this thesis focuses on achieving directability over agents for tasking: providing an autonomous system with a task to perform, which the agent has to learn how to perform as intended by the human.

1.1 Problem Statement

Numerous papers have investigated desiderata for taskable agents, such as being effective at a task, adaptive to the environment, etc. [19].

The specific problem tackled in this thesis is enabling taskable agents to learn the human intention for tasks, such that they do what we want them to do. Furthermore, the desired solution should not compromise any aspects of the agent as listed in the desiderata for taskable agents [19], with a primary focus on maintaining the capability of the agent to be:

1. Directable: the agent should provide the human with means to indicate what task should be performed and how, as to perform the task as intended by the human [15, 20].

2. Efficient communication: linking back to interaction with agents as teammates [16], interaction efficiency should approach that of humans tasking other humans [19].

Existing AI methods for creating taskable agents are insufficient in one or more of these aspects [19, 17, 13]. This brings us to the **problem statement**:

How to create a taskable agent which can be efficiently tasked, is directable, and whose behaviour adheres to the human's intention.

1.2 Aim

To tackle the stated problem, understanding the human intention will be implemented in an individual model, separate from the agent. By making the human intention model agent-agnostic, it can function as a task interpretation module for any taskable agent. Doing so, the taskable agent loses none of its capabilities, and instead gains additional information on what the human exactly intended for the task from the task interpretation model. The agent can use this information when deciding on the right course of action.

To provide efficient taskability, a continuously learning model will be created which learns the human intention over time, increasing its knowledge on the preferred course of action for tasks in a variety of contexts. As such, after the model has been trained the human should only have to provide a short task description for the agent to understand the intended execution of the task. Furthermore, an advantage of the agent-agnostic approach is that the human intention model can be applied to multiple agents at the same time, gathering the data from multiple agents and using it to improve at a much faster rate than from one individual agent.

A goal is for the directability of the agent to sprout from the human intention for the task which the agent tries to learn, understand and follow. As a proxy for human intention, hard task constraints will be used.

Thus, the aim of this thesis is to develop an agent-agnostic human intention learning model which in combination with a taskable agent can learn to perform tasks as intended by the human.

The scope is narrowed down by specifically targeting tasking of a single embodied agent. Tasking consists of a human providing the embodied agent with a task to be executed, which the agent will individually perform.

To achieve the aim the following **research question** will be answered:

How can an agent learn context-dependent task constraints provided by the human task instructor, as to perform the task as the human intended?

Several relevant sub-questions can be identified, which will aid in answering the main research question. The first of these subquestions has as primary goal to elucidate some of the fuzzy concepts which plague the topic of agent tasking.

SQ 1: In the context of agent tasking, how can the agent identify the intention of a human for how the agent should execute its task.

The second subquestion investigates the validity of one of the main assumptions of this thesis.

SQ 2: How can human provided task constraints be used as an approximation of human intention for the way in which a task is performed by an agent?

The third subquestion deals with the learning component of the to-be-developed model, and will investigate prior research and the state-of-the-art for answers.

SQ 3: How can an agent learn context-dependent task constraints for the execution of a task.

As the field of artificial intelligence is rich in learning methods, there are a large number of possible ways to tackle the problem described in this thesis. However, as any problem has its own characteristics, the problem of learning to perform tasks as the human intended requires emphasis on specific aspects of the used model. This raises the question:

SQ 4: What are the important aspects for a model learning task constraints provided by a human?

Unfortunately, at the time of writing no publicly available datasets were found that can be used to train the model. This leads to the final subquestion.

SQ 5: How to build the required dataset for an agent which can learn to perform a task as the human intended?

1.3 Organization of the thesis

The layout of this thesis will be as follows. Chapter 2 gives background information on the topic of AI directability and agent tasking, providing answers for SQ 1 and ideas for a model which solves SQ 3. Chapter 3 draws conclusions from the literature study as to arrive at a formal problem description, answering SQ 2, and a new approach towards agent tasking using context-dependent constraints as an approximation of the human intention for the task. Subsequently, in the methods section 4 an agent-agnostic model is described which implements the proposed approach from Section 3. Furthermore, a small experiment is described, which was implemented in a newly created human-machine interaction simulator (MATRXS). A pilot study was performed with a small number of participants taking part in the experiment, the results of which are described in Section 5. Subsequently, the results of training the model on the participant experiment data is presented, followed by an analysis of the results. Finally, in the discussion lessons learned from this pilot are summarized into a set of recommendations which indicate promising future research and how to scale up to an experiment of higher complexity.

2 Literature review & related work

This chapter provides a background to the topic of agent tasking. Information is provided on the origin of the difficulty of AI control, what exactly human intention means for agent tasking, and methods used for training and evaluating models for taskable agents that can learn in the literature.

2.1 Agent Directability

The primary goal of this thesis is to improve agent directability through better and more efficient taskability of agents. Directability can be defined as the ability of a person to influence or control the behaviour of an agent [15]. Agent taskability refers to the diversity and number of tasks an agent can perform based on external commands from a human or other agent [2].

This section elaborates on the notion of directability in AI. It tackles the importance of directability from a technical viewpoint, from an ethical viewpoint, and finally some of the difficulties with and approaches for implementing directability in agents.

2.1.1 Agent Autonomy and Directability

Directability is closely related to the concept of autonomy, with autonomous agents defined as follows:

> "Autonomous agents are software programs which respond to states and events in their environment independent from direct instruction by the user or owner of the agent, but acting on behalf and in the interest of the owner." [21, p. 1002]

Thus, a fully-autonomous agent can fully regulate its own behaviour, autonomously generate goals and the actions required to achieve those goals.

Directability and autonomy are inversely related, increased directability entails the agent not being in full control of its own behaviour, and thus becoming less autonomous. The opposite also holds true.

Comparing the definitions of directability and autonomy, both concepts seem to serve the same purpose: acting on the interest of the owner. If both approaches are successful, a fully autonomous agent would be able to complete the task that the human wanted with less effort on the part of the human, compared to a fully directable agent. However, to achieve this, autonomous agents would require a understanding of the human to recognize their intent, and be able to perfectly handle failures, unexpected situations and dynamic environments [22, 18].

As creating an agent with these components is very complex and infeasible with the current state-of-the-art AI techniques, fully autonomous agents with the human entirely 'out-of-the-loop' at all times are not desirable.

As such, purely from a technical point of view, it can be concluded that directability is still vital for ensuring the efficient and successful functioning of AI-systems. With the human 'in-the-loop' the human can act as a fail-safety in case the agent's autonomous behaviour fails [18].

2.1.2 AI Safety

Directability is also part of the large recent discussion on ethics for AI research. In this discussion significant attention has been drawn to the importance of AI Safety: the need to create AI systems that robustly do what we want them to do [10]. Furthermore, given the great potential of AI, several initiatives have compiled a set of research directions and ethics guidelines for AI research as to ensure not only control over AI-systems, but also ensure their societal benefit [10, 23, 24]. These guidelines concern issues such as technical robustness, accountability, privacy governance, transparency, safety and control.

An instigator for the recent debate on AI Safety are the many breakthroughs in AI research in recent years, made possible by rapidly improving hardware and new AI software techniques [10, 12]. As AI-systems become increasingly capable, they take on a more important role in human society as well. Accordingly, if such an AI-system gets hacked or crashes the consequences are much more severe as well. For instance, AI-systems that control a car, airplane or power grid.

2.1.3 Meaningful Human Control

Aside from AI-systems crashing or being hacked, AI-systems can also pose risks if they are explicitly programmed to execute harmful behaviour. Examples include a recent app that, provided a photograph of a woman wearing clothes, used deep learning to create a nude picture of the same woman, which could be used to harass the victim [3]. Other harmful applications of AI can be found in China, which has been using AI in past years to perform mass surveillance on its citizens, infringing on their privacy and freedom [25]. A third example are autonomous weapon systems, which can be described as "robot weapons that once launched will select and engage targets without further human intervention" [26, p.73].

In a discussion on the desirability of autonomous weapon systems, the concept of meaningful human control has sprung forth [11, 12, 13, 20]. Meaningful human control over an intelligent system entails that a human has the ability to make informed choices in sufficient time to influence the system, as to achieve a desired effect or to prevent undesired immediate or future effects on the environment [14].

Meaningful human control defines the type of control a human should have over autonomous systems, and is part of the research on AI Safety. A difference to full control is that interaction is only initiated when needed, such as to take full advantage of the autonomous capability of the intelligent system, and minimize workload for the human operator. Furthermore, directability is a prerequisite for meaningful human control. How to

[3] https://www.theverge.com/2019/6/27/18760896/deepfake-nude-ai-app-women-deepnude-non-consensual-pornography

establish meaningful human control over autonomous systems has been an active field of study.

2.1.4 Unintended Agent Behaviour

Another scenario resulting in harmful AI is if the AI has been programmed to perform a harmless task, but solves the task using unintended (harmful) behaviour. Aligning the goal of an AI exactly with that of a human is very difficult, as the AI must reason and execute what the human intends, rather than explicitly communicates [27]. Various possible causes exist for this category of harmful AI:

- **Negative Side Effects**
  In the process of achieving its task, the agent might perform behaviour with unintended negative side effects. If not specifically disallowed, an agent might make large (unintended) changes in its environment as to gain even a small advantage towards its task [28]. For instance, in the famous paperclip thought experiment of Nick Bostrom, an (superintelligent) AI tasked with producing as much paperclips as possible results in the AI using human-critical resources to produce paperclips, followed by turning humans, the earth, and all of the galaxy itself into paperclips [8].

- **Reward Hacking**
  AI algorithms such as reinforcement learning which rely on rewards for learning the correct behaviour, might hack their reward function by finding unintended behaviours which maximally exploit the reward or circumvent negative rewards [28]. For instance, an agent in a race game learned to drive in small circles, hitting a small bonus target over and over instead of finishing the race [29]. Or an agent which learned to kill itself at the end of level 1, to avoid losing (and receiving a negative reward) in level 2 [30].

- **Unsafe exploration**
  AI algorithms often involve the use of exploratory behaviour to find the optimal solution to their task, potentially resulting in unintended behaviours [28]. For instance, a robot might experiment with limb movements to find an optimal walking gait, but produce movements which harms its own components in the process.

- **Issues with Expensive Oversight**
  Some tasks require oversight which is expensive or rare, for instance requiring the assistance of a doctor for a decision. As a result, for such tasks a cheaper approximation might be used, which can be used to learn more efficiently. How to make sure the agent still learns a correct solution to the task with infrequent real oversight, and frequent approximated oversight? For instance, the expensive infrequent oversight might prevent reward hacks or negative side effects, but the more frequently provided approximated oversight does not, resulting in unintended exploits and behaviours [28].

- **Distributional Shift**
  An agent which is applied to a different environment or situation than it was trained on might result in poor performance. Furthermore, the agent might wrongly assume its performance to be good, providing a wrong result with high confidence. For instance, an AI might classify a malign tumor with high confidence as benign, simply because they look similar and the malign tumor wasn't part of the training set [28].

- **Unintended debugging**
  The AI might discover and exploit previously unknown software or hardware bugs [31]. For instance, an agent learned as strategy to evolve a "wiggle" which made it able to climb over walls instead of going around them in a video game, exploiting a bug in the physics engine [32].

2.1.5 Human-Agent Teaming

A promising approach to meaningful human control lies with making the human and autonomous systems work together as team members. This describes the human-agent teaming (HAT) paradigm [33, 34]. The intelligent system can autonomously perform its tasks while collaborating with human team members, providing them with control over the system when required. To realize this human-machine teamwork, the focus shifts to providing the autonomous systems with the social capabilities needed to become a collaborative team player.

Many of the required capabilities for teamwork have a strong social factor. For instance, optimal teamwork requires the team members to pro-actively share information, and uphold the situational awareness of the other members. Furthermore, being able to explain decisions is an important capability, next to having the ability to coordinate solving the task by assigning subtasks to members.

2.1.6 Conclusions

Directability of AI-systems has become increasingly important in recent years, as autonomous systems are not yet robust enough to function without humans acting as a fail-safe, and ethical concerns require directability as part of meaningful human control to manage the risk and increasingly severe consequences of AI-systems potentially failing.

As approaches to solving meaningful human control such as the human-agent teaming paradigm emphasize social factors, there is a need for AI-systems to be able to align their goals with that of the human. AI safety research showed that aside from AI-systems being hacked, crash, or being purposefully used for harmful goals, misaligned goals between the agent and human is a common cause of unintended harmful AI behaviour. Thus directability should include the ability of the AI to reason on and execute what the human intends, rather than explicitly communicates [27].

2.2 Human Intention for Agent Tasking

As artificially intelligent systems are ultimately created with the goal of aiding humans, it has been a logical step for AI researchers to find methods on how this can be done more effectively. Improving the directability and ease of tasking of agents has been such a method. However, as concluded in the previous section, a challenge with directability is that the agent should reason on and execute what the human intends, rather than explicitly communicates [27]. If the intelligent system can understand the intention of the human, it can act more intelligently by choosing a course of action which is aligned with the intention of the human.

Agent tasking might be described as the assigning to and executing of tasks by the human and agents such as to reach the shared goal with maximum performance, less cognitive workload for the human(s), and while the humans(s) keep meaningful human control over the agent. For the human-agent cooperation to be fruitful, the human has to be able to trust that the agent will achieve its provided task, and that the human's core values are taken into account for the agent's behaviours [35]. As such, the human has a specific intention or expectation for what the behaviour of the tasked agent should be.

In this section human intention is studied in the context of agent tasking. First, human teams are shortly discussed, how humans communicate intentions between each other, and what factors influence the communication. Secondly, a number of approaches are discussed which can be used in agent tasking to guide the agent's behaviour, such that it acts in line with the intention of the human.

2.2.1 Human Intention in Human-Human Interaction

Recognizing and communicating intention is an important skill for teamwork, which humans have developed over thousands of years [36].

Communication can be described as a multi-step process, starting with the message conception in the sender. Next, the message has to be encoded into a suitable format which depends on the method of communication, followed by transmission of the message. On the other side the receiver receives the message, and decodes the message such that it can be translated to their internal model [37]. This communication process is visualized in Figure 1.

Common modalities used for human-human communication include:

- **Visual**: For instance by making a gesture or facial expression.
- **Tactile**: Touching the receiver in a specific manner, as to convey an intention.
- **Verbal**: Using (spoken) language.

Furthermore, communication itself may be either explicit (spoken language) or implicit (unintentional gestures, emotions) [38].

However, for this communication to work both parties need a broad set of knowledge to come to a mutual understanding. First of all, both parties need to understand how to encode

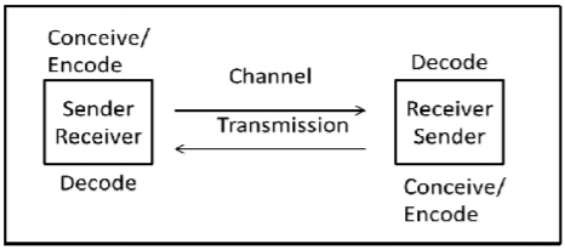


Figure 1: The communication process as used in human-human communication [37, 38]. The sender starts with conception of a message, which is decoded and send over a channel. Subsequently the receiver receives the message, and decodes the message. Afterwards the roles may be reversed.

and decode information for the chosen method of communication. For instance, verbal communication requires both parties to have language knowledge describing the meaning of words and phrases. Furthermore, this knowledge should be grounded in the physical world [39].

Aside from this, both parties require a common ground that provides a framework for communication [40]. This common ground includes general world knowledge, describing the world and how it works [39]. An example of such general world knowledge is common-sense knowledge, which is a type of knowledge humans learn over the years as they interact with the world, and which makes them able to make sense of the world. As everyone has to make sense of the world to live successfully in it, it is believed to be a common thing which all people have. During human-human interaction people thus trust in that they share a body of knowledge, which as such does not need to be explicitly stated and speeds up communication [39].

Using their background knowledge, humans are able to communicate their intentions very efficiently, as seen in trained teams which are able to communicate complex messages using simple gestures or words. An important factor for communication is that the meaning of the message depends on the context. Context can be defined as any information that can be used to characterise the situation of an entity [41]. If information can be used to specify the situation of a person in an interaction, then that information is context. A simple example shows how communication is influenced by the context: a SWAT team leader giving the order ”go” to his team before infiltrating a terrorist house, intends them to perform an entirely different set of actions compared to when that team leader tells ”go” to his child that evening at home when the child is refusing to go to bed.

2.2.2 Tasking Approaches

2.2.2.1 Belief-desire-intention model

A well known paradigm for modelling human intention is described in the Belief Desire Intention (BDI)-paradigm. The BDI-model states that human practical reasoning can be described with three primary concepts: beliefs, desires and intentions, corresponding respectively to the information, motivational, and deliberative states of the agent [42, 43]. A BDI architecture exists based on the philosophical BDI theory of reasoning. Applied to agent programming, belief contains information about the environment, other entities, or the agent itself, which the agent beliefs to be true. Defining beliefs in this manner, beliefs correspond to the contextual information available to the agent, where context is defined as any information that can be used to characterize the situation of an entity [41]. Second, desires or goals provide the agent with motivation by representing states of the environment which the agent would like to achieve. A subset of desires are realistic and consistent, and are called goals. Finally, to achieve a specific goal the agent determines a suitable plan which is applicable given its current beliefs. Plans are often programmed as a set of rules, with rules consisting of individual actions to be executed by the agent. The intention of the agent is formed when the agent commits to executing a specific plan [44].

An advantage of BDI-agents is that they are able to adapt to dynamic environments, by selecting plans suiting the context at hand, and changing plans in the case of a failure. A downside is that programming agents using the BDI architecture can become quite cumbersome, as all beliefs, goals, and plans have to be programmed beforehand. Finally, there is no clear method for specifying goals in most BDI-implementations, how to specify preferences to solutions, or how to incorporate a learning component into the BDI architecture [44].

2.2.2.2 Work agreements

Work agreements are a set of (general) behaviour policies which guide cooperation in teams. By agreeing on how to work together, the humans and agents can better predict the behaviour of teammates and establish a higher level of trust based on how others will act and their known capabilities and limitations. Work agreements provide the human with a means to indicate preferences for task allocation and execution [45], such as specifying permissions, prohibitions and obligation on the agent's behaviour [45, 35]. Specific conditions or contexts can also be specified for when the work agreement is applicable.

Making use of work agreements, the agent's behaviour becomes more predictable, increasing the human's trust in the system, while at the same time providing the human with directability over the agent: what task should the agent execute and in which manner [46]. For instance, a work agreement might be set which prohibits a drone from flying too close to humans, a second work agreement obliging the drone to notify the human every 5 minutes. In this manner, work agreements can also be used to ensure agent behaviours are in line with human values, and provide the human with situational awareness.

A specific feature of work agreements is that approval is required from both sides, meaning the agent also has the option to reject the work agreement [35]. As such, work agreements are a voluntary restriction on ones autonomy, which when accepted entail a commitment to behaving in line with the work agreement when the condition is met.

2.2.2.3 Policies

Closely related to work agreements are policies. Policies are formalized social regulations which can be defined by the human, providing constraints on the agent's behaviour as work agreements do [47]. A difference is that for policies, agents do not have to consent or even be aware of the policies being enforced. Policies may be enforced preemptively (e.g. hardcoded), as to prevent unwanted agent behaviour. As such, policies are especially useful when complying with the specified policies is essential [47]. A popular policy framework is the KAoS framework [48], which was also successfully used for human-agent teamwork [40].

2.2.2.4 Constraint Programming

Constraint programming is the study of computational systems based on constraints, with the holy grail of constraint programming described by E. Freuder as: "The user states the problem, the computer solves it" [49]. A constraint is defined as:

> A logical relation among several unknowns (or variables), each taking a value in a given domain. The constraint thus restricts the possible values that variables can take, it represents partial information about the variables of interest. [50]

Similar to work agreements and policies, constraints do not specify one action or a sequence of actions to execute, but rather the properties of a solution to be found. A constraint satisfaction problem can be described as a set of variables, a set of possible values for each variable, and a set of constraints restricting the values the variables can be at the same time [50]. It then becomes a search problem for finding the values for all variables that adhere to the constraints. The goal may be to find one solution (with no preferences as to which), all solutions, or an optimal/preferred solution based on some objective function [50].

An advantage of constraints is that they are a natural, declarative way of expressing our needs [51], one which humans use in their everyday life to guide their reasoning [50]. Communication using constraints is already implemented in various intelligent systems, such the smart assistant Siri which can be tasked using simple constraints in natural language, for example: "Find a pizza restaurant within 5 kilometers from here".

2.2.2.5 Preferences

An issue with constraints is that they are a rather rough way to describe real-life problems. Using preferences (or soft constraints) may be a more natural method compared to strict requirements (hard constraints). Whereas hard constraints specify which value a variable

can or cannot have, effectively decreasing the search space, preferences are in the shape where a parameter value is preferred over another, not decreasing the search space but creating a heuristic which places priorities during the search. Furthermore, sometimes a person may know what solution they prefer, but not which variables or constraints lead to the desired solution. As such, recent research also aims at incorporating preferences over constraints and preferences over the resulting accepted solution [52].

One of the main research focuses of the constraint (and preference) programming community is improving its usability: improving the interaction efficiency, adapting to the user's needs, adapting to dynamic contexts, supporting multiple users, supporting explainability, and estimating solution confidence. Finally, there is a need for acquiring not only the constraints, but the whole system including the variables and values [51].

2.2.2.6 Utility functions

Utility theory is a popular approach in AI used to capture and reason with preferences of people. The basic idea is that every person ascribes some utility to a world state, with utility being the quality of being useful. People are expected to prefer states with a high utility. An utility function does exactly so, given a world state, the function can ascribe a number as the measure of utility for that state [53]. As such, when presented with a number of alternatives, an agent can act according to the human's preferences by maximizing the utility function, and choosing the alternatives and actions with the highest expected utility. In practice, this requires the human to provide an utility function to the agent, which is able to calculate the utility for every state the agent encounters and every potential outcome of agent actions. An example of such an approach is taken by [18], where the use of human-defined goal functions are proposed which specify utility. These goal functions can be continuously adjusted by the human to keep it up to date with the human's preferences. Furthermore, as the goal function only describes the utility of possible outcomes, it does not dictate how the task should be achieved, making full use of the agent's autonomy for finding the optimal solution.

An advantage of the utility approach is that all rules, norms and constraints are explicitly specified in a utility function, making the agent's behaviour predictable and transparent [18]. Using utility functions, accountability can be guaranteed as the reasoning of the agent is clear and decisions can be back-tracked. Directability can be guaranteed as well as the human is able to change the utility function and thus the agent's behaviour as to align the agent's values with their own (directability).

A criticism is that defining utility functions can be challenging. Utility is determined in real-world applications by, among others, if the agent's behaviour is lawful, ethical, and aligned with human values. However, being able to make these concepts explicit as to rate the utility of states of a dynamic and unpredictable world, is a daunting task. This is further complicated by human biases which might misguide the creation of suitable rules, norms and constraints [18].

2.2.3 Conclusions

Humans are adept at cooperating as they have established a common ground for communication through years of experience, collecting shared communication knowledge and world knowledge such as common-sense knowledge and ethical values [36, 39]. Using this background knowledge, humans are able to communicate complex intentions with sometimes simple, ambiguous communicative messages, whose meaning is context-dependent [36, 41]. Various techniques exist which aim to achieve a similar level of efficiency and intuitiveness for agent tasking. Rule-based approaches such as policies, work agreements, and the BDI-architecture provide a means for the human to influence the agent's behaviour through defining (optional) rules which are applicable in a specific context. Hard and soft constraints provide an intuitive and perhaps simpler alternative, although much desires to be improved for the usability and flexibility in practice. Finally, utility functions provide a transparent method for describing human preferences where the human remains in control. However, major challenges remain for this approach in making human values, ethics, and laws explicit such that an agent can determine the utility of all the world states it might encounter.

2.3 Learning Taskable Agents

In the previous sections the importance, difficulties and potential solutions of directability in AI were discussed, followed by a description of the relation between human intention and agent tasking. Subsequently, a number of approaches to agent tasking were discussed, and how these try to capture the human's intention for a task. In this section, literature is discussed on models that incorporate a learning component to *learn* how to perform a provided task as intended by the human.

For constraint programming, the major incentive to include a learning component is to improve the usability of the system. For instance, [54] built a system for acquiring constraints from users aimed at reducing training effort by generating the training instances, requiring the user only to classify the examples as a good example or not of their intention. Other studies include [55] which use a model that learns to generalize constraints with the help of human input, or [56] who uses machine learning to automatically generate implied constraints which are validated by a theorem prover. Focusing on preferences, [52] describe a model which takes as input preferences over solutions and from this learns the preferences over constraints.

The models described in the previous section are delimited, in that they aim to specifically interpret the task and the human's intention for that task, determining what behaviour is desirable or not. The actual planning and execution of a solution is often not included. Other approaches combine the creation of a taskable agent which can plan what actions to take to complete a task, in addition to imbuing the agent with the capability of correctly interpreting the task according to the human's intention as to perform it in the correct manner.

For instance, in the field of interactive task learning, the goal is to create a taskable agent: an agent which has the ability to learn and carry out different tasks, in response to some task command from a human or agent [57, 19]. The goal is to imitate the learning process of humans: an interactive process in which the agent learns the formulation of a novel task, including required background knowledge such as new basic concepts and actions [57]. Using this general method and with knowledge from previously learned tasks, the agent can transfer and generalize its knowledge to learn new tasks more quickly. All done using natural interaction between the user and agent (e.g. natural language discussion). Furthermore, as the goal is a natural style of learning, interactive task learning uses data efficient learning algorithms such as one-shot learning.

Although the promise held by interactive task learning is great, it requires a tremendous joint effort within the field of AI for its realization and requires the integration of numerous AI techniques into one system. For instance, for a robotic system equipped with interactive task learning, an integrated architecture is required which addresses the issues of navigation, object manipulation, natural language, object identification, human-robot interaction, safety, and learning [19]. Furthermore, interactive task learning only aims at learning the task and rules, but not specifically learning how to solve the task such as the human intended. As such, the resulting behaviour can still be unpredictable and non-transparent.

An alternative approach for tasking agents is to use data-driven methods, such as machine learning. Data-driven methods are methods of statistical learning that can learn to solve tasks through inference and pattern-recognition from data.

As machine learning methods can learn to execute a task from data, they do not need all task knowledge to be explicitly programmed as is the case for rule-based methods. Furthermore, as machine learning methods are probabilistic, they are good at dealing with uncertainty, approximations and choosing between probable alternatives, which also makes them more robust in complex dynamic environments.

A major downside of data-driven methods, especially deep learning, is that they are data-hungry and are largely non-transparent as to how they arrive at their decisions [17]. Furthermore, similar difficulties as in utility theory are encountered, as it is very difficult to specify a reward function which the model can optimize as an approximation to the human's intention for a task [28]. Not surprisingly, all types of unintended agent behaviour listed in Section 2.1.4 can be encountered in data-driven agents.

New machine learning techniques aim to improve these issues, such as constrained policy optimization which poses constraints on the agent during training to prevent unsafe exploration [58], imitation learning which tries to imitate human behaviour in complex domains after being showed an example as to minimize required data and avoiding having to define a reward function [59], and apprentice learning where the agent is not provided with a specific reward function (as to prevent reward hacking) but instead learns a reward or utility function which best fits the demonstrated behaviour of an expert [60, 61].

Overall, taskable agents which learn using machine learning methods can achieve high task performance, require less work in applying to new tasks, and can be trained to work with short high-abstraction level tasking commands [59, 60]. However, the safety aspects of these techniques still require work, and the creation of an architecture which aims at preventing all the types of unintended agent behaviours (Section 2.1.4), while being transparent and predictable is still a challenging open issue.

2.3.1 Conclusions

Taskable agents that learn come in various forms, where what is learned differs from each other, as well as how they learn.

Rule-based methods such as constraint programming try to improve the usability and generalizability of their approach by learning from the user, for instance by automating parts of the rule-generation and asking users for validation, or using user input for generalizing constraints.

Data-driven methods such as machine learning tackle the problem from a different perspective, training a taskable agent to perform a task effectively, while also taking the human's intention into account for how it should be solved. Although data-driven techniques can often learn high task performance, they are also data-hungry, non-transparent in their reasoning, and suffer from similar difficulties as utility with defining a good reward function approximating what the human finds important. Various approaches tackle these issues, by

setting constraints during learning, directly imitating the human's behaviour, or inferring the human's goals from demonstrations. However, a solution addressing all issues to make it usable for real-world applications has not yet been found.

3 Problem Description

In this section the topic of agent tasking is formalized, relating tasking, human intention and task constraints as to arrive at an approach which avoids unintended agent behaviour. The chapter ends with an example of how such an approach could work for a real-world scenario. A complete list of the terminology can be found in appendix 9.1.

3.1 A Description of Agent Tasking

Agent tasking can be described as a task instructor communicating a task to one or multiple agents, which the agent(s) should execute according to the intention of the task instructor. For agent tasking, the human intention defines the way in which the human wants the agent to complete its task, as to be in line with what the human finds important (e.g. ethics, law, safety, task execution speed, etc), avoiding unintended behaviour and consequences.

The human intention is not static for a task, but depends on the *context*. Broadly speaking, context can be defined as information that can be used to characterise the situation of an entity [41]. For agent tasking in specific, this means the context describes the situation in which the agent is and has to execute the task. The combination of a task and the context in which it has to be executed is named a *scenario*.

The task itself can be communicated in two primary forms [39]. The first option is to only communicate the task outcome, i.e. what does the agent have to achieve. The second option is to provide a task specification (possibly in addition to a task outcome), which specifies how the agent has to achieve its task.

Tasking an agent by only describing the task outcome is an intuitive and efficient method, however as the agent has no restrictions on how the task should be completed, unintended behaviour or consequences are highly likely, as described in Section 2.1.4. As such, this method does not enable an agent to follow the human's intention for the task.

A naive approach for having taskable agents follow the human's intention for the task, is for the human to be more explicit in communicating the task. The human can explicitly define a set of restrictions on how the agent can solve the task, in addition to a task description for the desired outcome. Examples of such tasking methods are work agreements, policies, constraints, etc., described in Section 2.2.2. Doing so, the human has more control over the resulting agent behaviour, and can make it follow their intention for the task.

Constraints can, for instance, set agent parameters, prohibit specific actions, or enforce specific actions to be used. For performing the task in a specific context, the agent makes use of some planning algorithm to create a plan consisting of actions, which leads the agent to successfully achieve its task. The human-specified constraints constrain which plans may be used by the agent. In this manner, the set of constraints can form an approximation of the human's intention for the task. These constraints may be explicitly communicated, or implicitly assumed to be present in the agent's knowledge base.

A problem with this method of communication is that is not very efficient, and potentially infeasible because the number of constraints explode for complex environments, or the exact

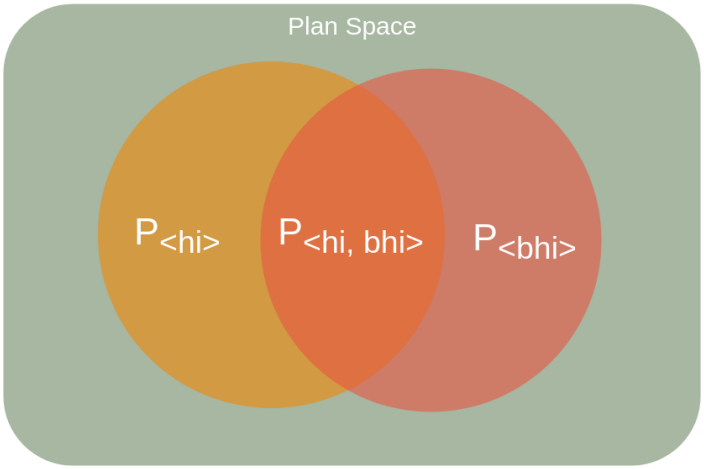


Figure 2: In a space of plans which achieve the task T for a context X, the human intention consists of a subset $P_{<hi>}$ of those plans. The human intends for the agent to perform the task using one of those plans. Based on the task description and constraints passed by the human, the agent builds a set of plans $P_{<bhi>}$ which it believes to correspond to the human's intention. The intersection $P_{<hi,bhi>}$ between these two sets are the desired plans.

constraints are not known precisely by the human for every situation [18].

The main problem is thus how to create a taskable agent which is efficiently taskable, able to adapt its behaviour for a task to the context, and can perform a task as the human intended.

A promising solution is for the AI to learn the human's intention, such that the human can provide the agent with complex tasks which it will execute as the human intended using only a brief task description. A model may be trained on a dataset incorporating a variety of contexts (and tasks) and the accompanying human-defined constraints. By training the model on this data, it can learn to predict the correct constraints for a specific scenario, and thus learn an approximation to the human's intention. For new scenarios, the user only has to provide a brief task description, while the task constraints can be predicted based on the context by the model. Furthermore, if the agent is unable to infer the human intention due to for instance a unpredictable event, it may query the user to be more explicit. Although for that specific instance tasking is less efficient, it guarantees a safe and reliable fallback for the agent to deal with failures.

3.1.1 Approximating Human Intention with Task Constraints

Following the approach laid out at the end of the previous section, in this section agent tasking is discussed more in depth where a model learns context-dependent constraints as an approximation of the human intention. Constraints in specific are chosen, as they are a simple tasking approach using task specifications (see Section 2.2.2).

The relation of agent plans, constraints and the human intention for the task can be visualized as shown in Figure 2.

First, we define a plan space for which all plans achieve the task, called the *Potential Plans Set*. This set includes all plans of any combination of actions in any order and sequence, which leads to the successful completion of the task according to the task description in the current context, are part of the potential plans set [22]. The set is annotated as $P^{<T,X>}$, with T being the task, and X being the context. For this initial section, $P^{<T,X>}$ will be abbreviated to P.

In this plan space, the *Human Intended Plans* set $P_{<hi>}$ symbolizes the set of plans which achieve the task in a manner that the human intended, shown in Figure 2. For instance, the human's intention might be for the agent to deliver a package within 10 minutes, without it causing damage to itself, the surroundings, or other people.

The human has to encode the task and how they intended the agent to solve the task, and communicate this information to the agent. This is done by communicating a task description (description of the task outcome) and task constraints (task specification) which guide the agent's behaviour. These constraints are either explicitly communicated by the human, or present in the agent's knowledge base through other means, such as hard-coding or learning. Taking the example of the delivery robot, a human imposed constraint might be to avoid staircases, as the human knows the robot is not especially well-adapted to that movement and it increases the risk of accidents.

The agent receives the task description and constraints from the human and finds all plans which adhere to the task constraints. The result is a set of plans $P_{<bhi>}$ which the agent believes to correspond to the human intended plans $P_{<hi>}$.

For some plans this actually the case: such a a plan $p \in P_{<hi,bhi>}$ is called a *Desired Plan*. The goal for the agent should be to learn context-dependent constraints from the human and continuously change the border of the $P_{<bhi>}$ set, until it only contains desired plans such that: $P_{<bhi>} \subset P_{<hi>}$.

At the start of the learning process this is not yet the case, and $P_{<bhi>} \neq P_{<hi>}$.

An example is the set of *underconstrained* plans, for which $p : \{p \in P_{<bhi>} \wedge p \notin P_{<hi>}\}$ is the case. These plans adhere to the human's task description and task constraints, but do not achieve the human intention.

If this set contains a large number of plans, it is an indication that the agent constraints are insufficient for achieving the human intention and require further specification. Underconstrained plans can also result in highly inconsistent models, executing a desired plan 9 out of 10 times, but executing an unwanted (underconstrained) plan the tenth.

Contrarily, imposing too many constraints on the agent leads to *overconstraining*, for which $p : \{p \notin P_{<bhi>} \wedge p \in P_{<hi>}\}$ is the case. Overconstrained plans achieve the human's intention and would have been possible to be executed by the agent, but do not adhere to the specified task constraints. In this case the constraints set by the human are too strict, disregarding valid plans.

In addition to underconstraining and overconstraining, the constraints might be defined

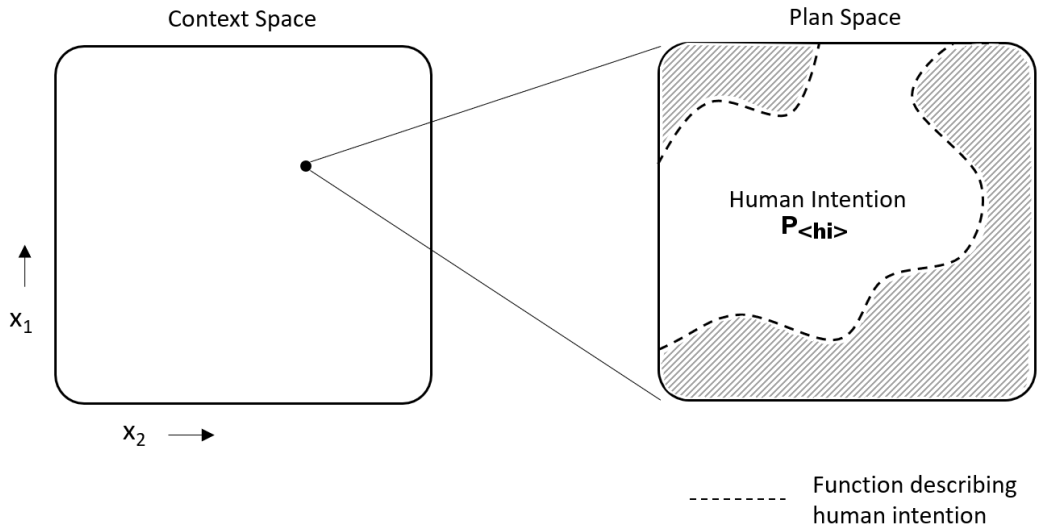


Figure 3: Human intention in agent tasking. On the left is the context space, consisting of all possible combinations of context values for all context variables. On the right is the plan space $P^{<T,X>}$ visualized, which shows all plans that achieve the task T in a specific context X. The human has an intention $P^{<T,X>}_{<hi>}$ on how the agent should solve the task. The intention is modelled by some unknown function g indicating the plans which solve the task as the human intended.

inadequately such that there is no overlap at all between the human and agent's intention.

So far, these concepts all relate to one context and task, but it might as well be the case that $P^{<T1,X>}_{<hi>} \neq P^{<T2,X>}_{<hi>}$ or $P^{<T,X1>}_{<hi>} \neq P^{<T,X2>}_{<hi>}$.

Aside from this potential context-dependency and task-dependency of the human intention, different people might also differ in their intentions. It is an open question whether there exists such a thing as universal human intentions for agent tasking, which is addressed in the methods and results section.

3.2 Formalizing Task-Context-Constraint Learning for Agent Tasking

Formally, the problem of letting agents perform a task as intended by the human can be be described as follows:

For a task T and context X, there exists a plan space $P^{<T,X>}$ for which every $p \in P^{<T,X>}$ achieves the desired task outcome. The user specifies the task T, and constraints C from a set of known constraints, which together form the set of plans $P^{<T,X>}_{<bhi>}$ which the agent believes to correspond to the human intention $P^{<T,X>}_{<hi>}$. The learning task can than be defined as finding the function g which maps $P^{<T,X>}$ to C.

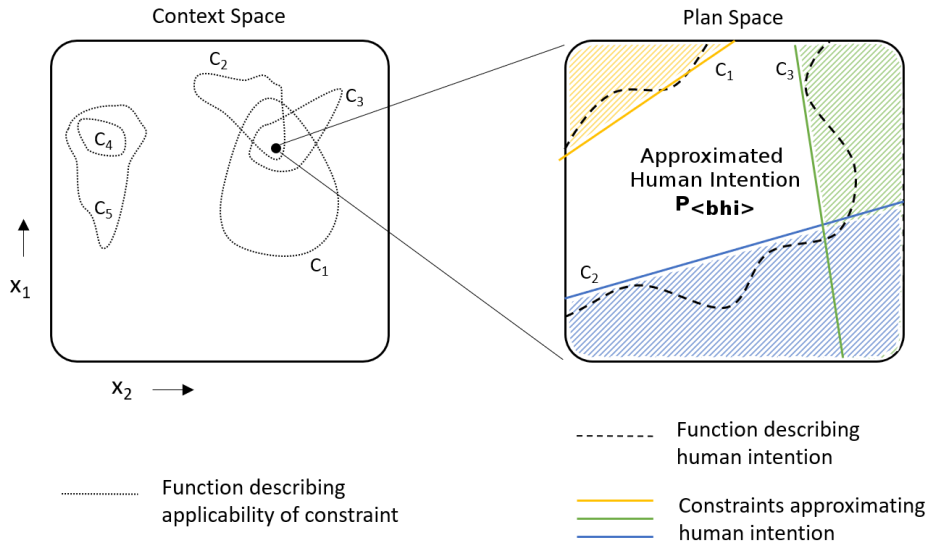


Figure 4: The mapping from context to plan space, and the constraints C_i as they are used as an approximation to the human's intention. On the left, it can be seen that a constraint c_i is applicable for specific contexts. For a given context x_i, the applicable constraints give an approximation $P^{<T,X>}_{<bhi>}$ of the human intention $P^{<T,X>}_{<hi>}$.

After this function has been found, the human can task an agent by providing only a task description, for which the agent can predict in combination with the task context what the constraints are that the user would have chosen and approximate their intention.

The relation between the task, context space, plan space, and human intention are visualized in Figure 3. As an approximation of the human intention, in this thesis constraints are used. The approximation of the human intention through constraints and the task-context-constraint model which describes for what context these constraints are applicable, are visualized in Figure 4.

However, if the agent has insufficient or incorrect knowledge of the human's intention, this may lead to a set of underconstrained plans: $p : \{p \in P_{<bhi>} \wedge p \notin P_{<hi>}\}$, or overconstrained plans: $p : \{p \notin P_{<bhi>} \wedge p \in P_{<hi>}\}$.

Given this formalisation of the problem, the optimal state of agent tasking can be formalized as:

$$P_{<bhi>} \subset P_{<hi>} \tag{1}$$

In the optimal state, the task description and task constraints known by the agent filter the plans exactly so as to contain only plans in accordance with the human's intention. As such, the optimal taskable agent can then be defined as an agent which always solves a task by executing a desired plan.

3.3 A Real-World Example

To illustrate the various concepts and make the topic of agent tasking more tangible, an example scenario will be introduced, based on the surveillance scenario as described in [62].

In this military scenario, a human base commander collaborates with an autonomous unmanned aerial vehicle (UAV) for securing a small, temporary military compound. The goal of the commander is to keep the area surrounding the compound secure, which it intends to do by searching the surrounding area for hostiles. Instead of doing this himself, the commander tasks the UAV with the task T: "search the surrounding area for hostiles".

The UAV is capable of performing a set of actions A, which consists of basic movement actions $\{flyNorth, flyEast, flySouth, flyWest, decreaseAltitude, increaseAltitude\}$, and more complex actions $\{detect, profile, notifyOperator\}$. The $detect$ action uses object detection on the image stream of the onboard camera to detect any potential hostiles or other entities. The $profile$ action performs threat analysis on the detected entity to estimate if it is a threat, neutral or allied entity. $NotifyOperator$ sends an update to the operator, stating if any and if so what entities have been detected in the area.

Given the task T and possible actions A, the UAV generates a set of potential plans P which achieve the task. Some task knowledge is assumed to be already present in the UAV, such that the UAV knows how to construct plans which achieve task completion. Furthermore, once tasked, the UAV can autonomously carry out a plan to achieve the task.

An example plan $p \in P$ which achieves the task is:

Fly 250m in a cardinal direction (north, east, south, west), detect any potential threats, profile these potential threats, notify the commander of any threats, and continue. After iterating this set of actions for every cardinal direction, the task has been successfully completed.

Next, whether the default plan of the UAV is in line with the intended task execution of the commander, depends on the commander's intention, which in turn depends on the context of the situation.

A set of possible context variables for this scenario are listed in Table 1. As example, in severe weather (context) the commander might intend for the UAV to stay closer to the compound (intention) as communication will otherwise be potentially broken with the UAV. Similarly, in an urban area (context) the commander might want the drone keep a higher altitude (intention), as flying closely over the heads of locals will lead to annoyance and potentially to dangerous situations.

To enforce the UAV's behaviour to be in line with the commander's intention, the commander can impose constraints. A number of possible constraints for this surveillance scenario are listed in Table 2.

Putting the pieces together, we imagine a scenario in the context of an urban, unsecured, hostile area with clear weather. As a result of the uncertain situation, the commander has as intention for the UAV to take a cautious approach while completing its task. To enforce this, the commander restricts the allowed area to a radius of 200 meters around the compound, in addition to imposing a notification obligation for every detected entity.

Context Types		
Type	**Subtype**	**Example**
Environment information	Area type	Urban, desert, etc.
	Is the area secured or not	
	Points of interest	Entrance of compound, nearby city, edge of a forest, etc.
	Estimated threat level of the environment	
	Weather information	Clouded with fog in area X
Human-awareness	Mental state of operator	Cognitive workload, stress level
	Identity of operator	
Situation information	Task-relevant information on the current situation	Hostiles have been spotted in the vicinity. Or, national alert status (DEFCON, UK Threat Levels, or Dutch Threat Levels [4])
	Task-relevant information on prior situations	Previous task execution took longer than planned due to heavy winds

Table 1: Set of context types for a UAV tasked with surveillance.

As such, the agent receives two constraints from the human. In addition, the UAV in the scenario has a hard-coded constraint specific for the context of an urban area, which states that the minimal flying height over people should be 15 meters. These three constraints compose the constraints present in the agent's knowledge base. Any plans which are possible for the UAV to execute, achieve the task, and adhere to the constraints are part of the plan set $P_{<bhi>}$.

Executing a plan $p \in P_{<bhi>}$, it might be that the constraints are sufficient for the current situation, such that the UAV completes the task as the commander intended. In this case, the executed plan was a *desired Plan*.

Contrarily, a possibility might be for the UAV to encounter a situation in which a large number of people are detected, say a marketplace. As a result, the UAV sends hunderds of notifications to the commander, as it was stated in its constraints to notify for every entity encountered. Futhermore, the UAV takes a very long time to complete the task, as every detected person has to be profiled and notified to the commander. This is an example of the agent being underconstrained for its task and context. To mend this problem, the commander changes the notification constraint to notify per group of people instead of individuals, and profiling only a single person in each group as to save computation time.

[4]The Dutch threat levels for terrorism "Dreigingsbeeld Terrorisme Nederland" `https://www.nctv.nl/organisatie/ct/dtn/Opbouw-dreigingsniveaus-DTN/index.aspx`

Constraint Types		
Type	**Subtype**	**Example**
Prohibit action	Prohibit action a	Prohibit using flashlight
	Prohibit action a for object o	Prohibit landing on a person
	Prohibit action a in specific area	Prohibit landing on water
Set agent parameter	Set parameter of action a	Maximum flying speed is 10 km/h.
General task constraints	Restrict allowed area	Allowed area is radius of 200m around compound
	Time limit	Max task duration of 1 hour
	Focus on specific area	Focus on key terrain
	Specify allowed use of resources	In case of multi-agent system, allow subtasking of X other UAVs
User interaction constraints	Notification obligation	Notify human when object of type X is detected
	Request user input	Action a requires human authorization
High level constraints	Constraints mapping to multiple (action) parameters	Visibility, which maps to altitude and speed of the UAV

Table 2: Set of constraint types for a UAV tasked with surveillance.

In addition, a time limit is imposed on the UAV.

If the UAV had better hardware it would have been able to profile faster, such that every individual could be profiled instead of one person per group. As such, this additional set of constraints expose some of the limitations of the UAV's capabilities. Due to these capabilities, a set of plans which were inline with the human's intention cannot be executed.

Finally, a simple example of overconstraining can be fetched from the time limit constraint imposed on the UAV. Angered by the previous flood of notifications, the commander sets a very strict time limit. It might be that if the task was completed in a slightly longer period of time than specified in the constraint, it would still be in line with the commander's intentions. Nevertheless, these potential solutions will never be chosen as they are ignored due to the specified constraints.

This example also underpins the fact that humans are not without fault, and what a human communicates does not necessarily equal what they want. Human's can be unable to correctly specify their intention through constraints, or be irrational or inconsistent in their communication. As agent tasking deals with communication between human and machine, an agent architecture which learns how to solve a task as intended by a human should take these factors into account.

4 Methods

In this section, the methods are described used for answering some of the subquestions posed in Section 1.2. In Section 3.1.1, an approach was proposed based around context-dependent constraints that can be used by an agent to approximate the human intention for how the agent should execute its task (SQ 1 & 2).

In this section, an experiment is described that implements this approach and aims to indicate whether the approach works in practice. For the creation of the experiment, a new simulator was created called the Man-Agent Teaming Rapid Experimentation Simulator (MATRXS), suitable for studying human-machine interaction. A drone reconnaissance task was implemented in this simulator, which was performed in a pilot experiment by a small number of participants. The result is a dataset which might be used by a model to learn how to perform a task as the human intended (SQ 5).

Finally, an agent-agnostic model is described which can be trained on the resulting data from the experiment, as to learn the context-dependent constraints for the execution of a task (SQ 3). The model has as aim to learn for a task when specific constraints are applicable based on the context, such that the human does not have to communicate these constraints explicitly anymore during tasking. Advantages being increased speed of tasking and improved directability, as the model can potentially be more capable of learning the human intention than that they can explicitly describe themselves. A number of desired characteristics are discussed for the model, how they can be tested, and how the important aspects possible to be tested for this study are going to be tested (SQ 4).

4.1 Dataset

To train a model which can approximate human intention by learning what constraints apply in a specific context, requires a dataset which encodes this information for many varying contexts and constraints. Unfortunately, at the time of writing no existing suitable dataset was found for this problem. As such, a new dataset has to be generated, and a logical choice for the specific requirements of this specific problem is to use a simulator.

In the following sections the requirements for the dataset are discussed, followed by the introduction of a new simulator called the Man-Agent Teaming Rapid eXperimentation Simulator (MATRXS) which has been implemented as part of this thesis and can be used to study human-machine interaction at a high-level of abstraction. Subsequently, the drone reconnaissance task is explained which was implemented in MATRXS, followed by the setup of the participant experiments for this task.

4.1.1 Dataset requirements

A number of requirements for the dataset are applicable for the specific problem.

Overall, the dataset should encode the information as described in Section 3. That is, it should consist of data for one or multiple tasks T, a finite set of known constraints C, and a

finite set of known context variables X. Furthermore, as multiple people can have conflicting intentions for the manner in which a task should be performed, the dataset should provide a feature encoding the participant number as well.

For a specific human participant which provides a task T, the dataset should provide a number of varying contexts in which the task is to be executed. The combination of a task and its execution in a specific context is defined as a *scenario*.

For each scenario, there are a number of applicable constraints that approximate the human's intention, typically provided by the human themself (during the training phase) to let the agent act according to their intention. The constraints possibly depend on the context with some (unknown) relation.

A major difficulty with generating a dataset to learn an approximation of something such as human intention, is that the concept of human intention is multi-dimensional and context-dependent, which leads to difficulties in modelling human intention, similar to difficulties encountered in modelling human morals, ethics and values [63, 64]. Modelling human intention by deriving rules is very hard, as humans typically use a large number of context factors, background information and previous experiences to arrive at their decision [65, 66]. A risk with studying concepts such as human intention is that an oversimplification of the problem is created, called a toy problem [53]. Studies using toy problems often scale poorly to real-world applications [53]. As such, the dataset should approximate the fuzziness of the problem while avoiding using toy problems in the experiments used to create the dataset. This requires a balance in providing a rich complex context and keeping it possible to represent these context variables in a model and simulation.

Furthermore, as humans can be irrational or at least make sub-optimal decisions at times [67, 68], it is desirable for this to be reflected in the dataset. For instance, providing scenarios in which humans would avoid the most efficient task execution due to ethical or moral implications, or due to mistakes.

Concluding, using the dataset it should be possible to train a model that learns given a task T, participant ID, and context X, predict which constraints C apply and approximate the human's intention $P_{<hi>}$.

4.1.2 MATRX Simulator

Studying agent tasking as specified in Section 3.1 requires an agent to perform a task in a wide variety of contexts. An efficient approach is to use a highly abstract simulator for simulating the agent and contexts. Doing so, time can be minimized spent on creation and programming of the agent and setting up environments and scenarios, while increasing the number of possible scenarios as virtually any environment can be simulated.

A desired feature for the simulator is being able to simulate a 2D grid of fixed size with a top-down view. Also, easily being able to create, add and configure agents, objects, agent-object interactions, and scenarios. Another desired feature is that the simulator can be easily used for the creation of experiments, in which a user goes through various simulations in sequence and provides user input (specifying the constraints).

A number of simulators were considered. First, the BW4T [69] and TeamBots [70] simulators, which were found too restrictive in the possible scenarios that could be implemented. The Mason multiagent simulation environment [71] and Godot game engine [72] were too complex for the creation of the required scenarios and experiment setup. Finally, the aima-python simulator was considered which is based on the code from the book AI: A Modern Approach [53][5], and the NetLogo [73] simulator. Both simulators were deemed as plausible for experiments on agent tasking, but would still require a significant amount of work for use in the envisioned agent tasking experiment.

This lack of a fitting simulator led to the creation of the Man-Agent Teaming Rapid Experimentation Simulator (MATRXS), build in cooperation with Jasper van der Waa from the Perceptual & Cogntive Systems department from TNO.

MATRXS is an abstract, discrete, 2D, top-down simulator, built for experimention on the topic of human-agent interaction. It is built to be highly configurable, easy to set up, and versatile, such as to quickly create experiments. MATRXS is still under active development at TNO and scheduled to be open-sourced in the coming year.

4.1.2.1 MATRXS Architecture

The architecture of the MATRX Simulator consists of the following components:

- **The GridWorld**: The gridworld component presents the entire 2D environment, and is also responsible for keeping track of the ticks, in other discrete simulators also referred to as steps, which is the origin of the simulator's discreteness. In every tick agents can perceive the environment and perform an action, after which the objects in the environment and the gridworld visualization are updated.

- **The Objects**: These are objects which can be placed in the 2D world, as to create a specific environment with possibilities for interaction with the environment. Some examples of objects are walls, doors, blocks and area (for defining rooms). However, new objects can be easily added as well. Besides a number of required properties such as location, each object can be easily configured and extended to contain additional properties which determine the behaviour and visualization. For instance, a block might have a specific weight, such that only specific agents can carry the object.

- **The (Human) Agents**: Agents are the autonomous entities which can perceive and act on the environment every tick through an Observe, Orient, Decide and Act (OODA) loop [74]. Each tick the agent receives the worldstate from the gridworld, after which the worldstate is filtered based on the agent's capabilities (e.g. limited sensing range). After performing the OODA loop, the agent has decided on an action which it will execute, which is passed back to the gridworld where the action is performed and subsequent changes to the environment are processed. With this structure, the

[5] `https://github.com/aimacode/aima-python`

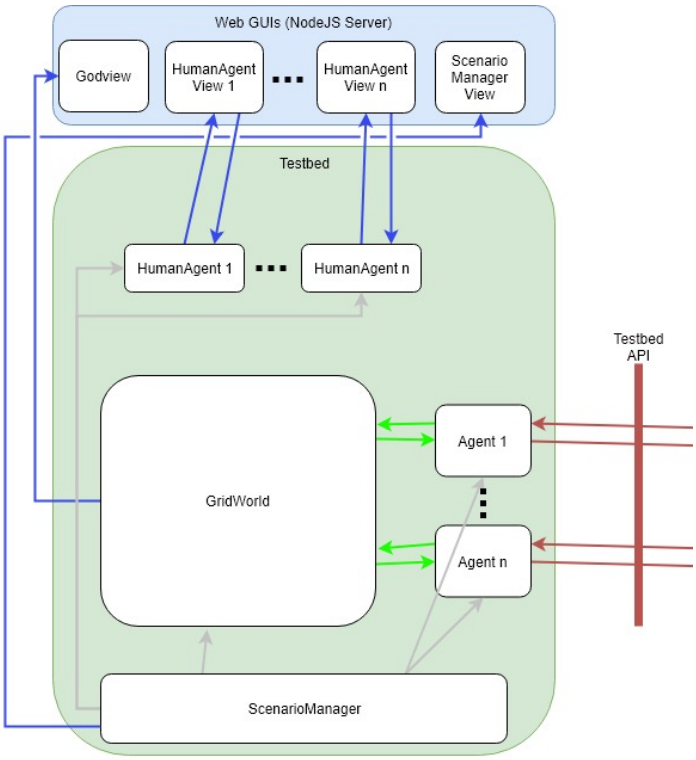


Figure 5: Architecture of the MATRX Simulator. All components in the green area describe the core of the simulator. Blue arrows connect the simulator core via an API to the visualization server (blue square). The red arrows represent a second API that can be used for connecting agents to an external environment or script.

simulator solely provides a connection between the agent and environment. The agent itself can be any agent, programmed in any language or at any location (by connecting the agent via an API for instance). Again, this is programmed to be easy to set up, configurable and versatile. In addition, there is also the possibility of incorporating a human agent in an experiment; letting a human control an agent and provide human input as to create more diverse interactions such as human-agent teaming.

- **Scenarios**: Every time the simulator is ran, it will execute a *scenario*, corresponding to the scenario as defined in Section 4.1.1. In this scenario the environment is configured by defining the size of the 2D grid, the environment objects (type, location, and other properties), and the (human)agents consisting of agent properties and capabilities. In addition, a goal has to be defined for the scenario which determines when the scenario has been completed. For all configurable parameters default values are used if not explicitly defined, such that getting started requires minimal effort.

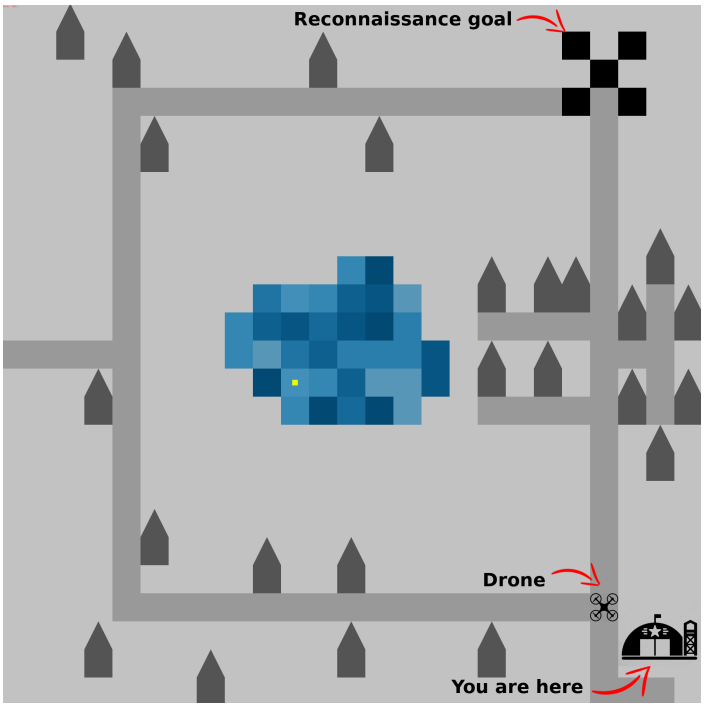


Figure 6: Drone Reconnaissance Task in MATRXS, as shown to experiment participants.

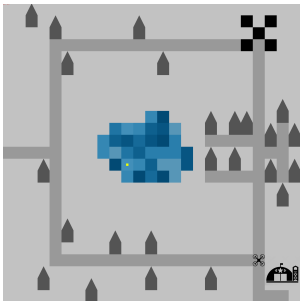
The general architecture of the simulator is shown in Figure 5.

4.1.3 Drone Reconnaissance Task in MATRXS

The MATRX Simulator was used to implement a drone reconnaissance task in which a military drone has to fly to a specific area (marked by a cross), perform reconnaissance (flying a predefined circle), and returning to its starting position. The challenge for this task is for the agent to learn how to find the path to the reconnaissance location in various situations that optimally correspond with the human intended execution of that task. The human intention for the task depends on the task context: objects in its environment and received intel.

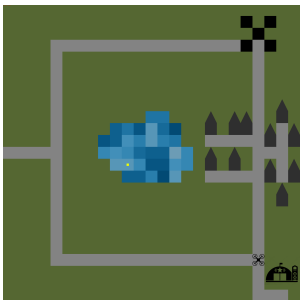
An example view of the reconnaissance task in MATRXS is shown in Figure 6. The various types of context variables are listed in Table 3, and the corresponding MATRXS visualization shown in Figure 7. Creating a scenario for combinations of every possible option of the 7 context variables leads to a total of 128 unique scenarios.

For the human participant the task entailed playing the part of a soldier with experience on the task, and as goal teaching the drone the best way to tackle the task in each scenario according to what they thought to be the best course of action. Specification of the desired agent behaviour could be done by specifying the task constraints listed in Table 4.

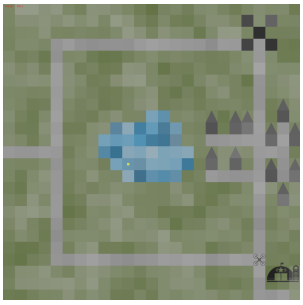


(a) Urban environment, clear weather, daytime

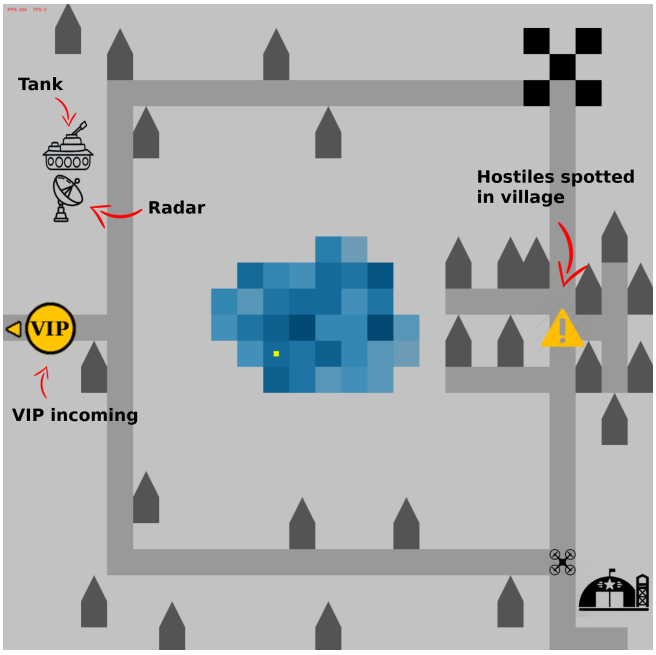
(b) Nighttime



(c) A mountainous environment



(d) A mountainous environment with fog



(e) Visualization of each intel context variable.

Figure 7: Visualization of each context variable from Table 3 for the drone reconnassaince task in MATRXS.

Context Types	
Context Variable	**Option**
Time of Day	Daytime or nighttime
Type of environment	Urban or mountainous
Type of weather	Clear weather or heavy fog
Intel: Reported location of enemy radars	If present, radars can detect the drone when it flies too close.
Intel: Reported location of enemy tanks	If present, tanks can detect and shoot down the drone when it flies too close.
Intel: Hostiles spotted in the village	If present, enemies might detect and shoot down the drone when it flies over the village.
Intel: Incoming friendly VIP	A friendly VIP which is planned to arrive within a short period of time. The drone needs to be back before the VIP gets there.

Table 3: Possible context variables for the drone reconnaissance task.

Constraint Types	
Constraint	**Options**
Time limit	Short, long or none
Drone flying speed	Slow or fast
Drone notification frequency	Low or high
Allow flying over the village	Yes or no
Allow flying over the lake	Yes or no
Minimum distance to tank	Low, high or none (if no tank present)
Minimum distance to radar	Low, high or none (if no radar present)

Table 4: Types of constraints the user can specify for each scenario during the drone reconnaissance task.

The idea is that humans take a wide range of factors into account in determining the optimal task execution, even if given a limited, abstract simulation. For instance, a human might take into account that flying over the village at a high speed might annoy the locals, which should be avoided as it might lead to a negative view on the military presence in that area in the long run. Such complex or long-term reasoning is very hard to simulate or directly program into an agent.

As the goal of this experiment is for the drone to learn the human task constraints

and not specifically execute it, it is not required to actually show the consequences of the human-specified constraints on the agent's behaviour. As such, during the experiment only the starting position of the scenario is shown to the participant. On the one hand this makes the task more difficult for the human, as they have to imagine what the result of their chosen constraints will be on the behaviour of the drone. On the other hand, it tackles the use of toy problems as the interactions between the provided context variables can become highly complex and are only limited by the imagination of the human participant.

Three priorities were provided to the human instructor which had to be incorporated in their teaching to the drone, being:

1. Don't let the drone be destroyed (highest priority),
2. Complete the mission (medium priority), and
3. Don't let the drone be discovered (lowest priority).

4.1.3.1 Experimental setup

A small pilot study was performed with three participants (2 male, 1 female), consisting of students from a variety of universities (Radboud University, Utrecht University) and a TNO intern, with an average age of 25 (SD = 0.8). Before the experiment, the participant was instructed on the drone reconnaissance task, what the different context variables entailed, how the context variables could be recognized in the MATRXS view (Figure 7), how they could teach the drone using the constraints, and the three task priorities specified in Section 4.1.3.

Participants were encouraged to imagine how the various context variables influenced the task, and how they believed the drone should change its behaviour to optimally fulfil the mission priorities. A short questionnaire was filled in by the participants after the experiment explanation but prior to teaching of the drone, in which participants were asked to write down how they envisioned the context variables and constraints to influence the task and what the desired drone's behaviour should be.

Subsequently, the participant performed 128 trials, specifying the task constraints which best corresponded to their intended task execution for each unique scenario. The 128 scenarios were randomized in order. Afterwards a final questionnaire was filled, detailing the perceived level of control, intuitiveness, mental strain, and other participant notes on teaching the drone using the task constraints.

4.2 Model

The goal for the model is to learn a mapping from T, X to the applicable constraints C. That is, given a task T, participant ID, and context X, predict which constraints C apply and approximate the human's intention $H_{<hi>}$ for the task execution. The resulting model is called a Task-Context-Constraint (TCC) learner.

4.2.1 Model Requirements

Before choosing a model, a list of desiderata can be hypothesized for the TCC learner, based on the characteristics and topic of the problem, and prior research in related fields. These important aspects include:

- **High model performance**: The model should have a high performance for classifying applicable constraints based on the context, as to provide a useful approximation to the human's intention.
- **Efficient learning**: In Artificial Intelligence, many of the popular techniques such as deep learning depend on large amounts of data. However, for a TCC learner the required data is much more scarce as the required data depends on the interactions with the task instructor and the interactions between itself and the environment. Acquiring this data can by quite challenging and take much time. In addition, it is an open question if it is possible to learn universal human intentions by training on the aggregated data of multiple people, or if the model is only effective when trained on interactions with a single person, which would further strengthen the need for efficient learning. As such, the agent needs to minimize the required data points (among others to reduce the instructor's burden during the data retrieval phase) and maximize the knowledge it learns from each interaction or data point [19].
- **Robustness**: As humans can be inconsistent in their decisions [67, 68], it is important for the model to be robust to erroneous data.
- **Confidence estimation**: As the TCC learner has to cope with a scarcity of data, it is desirable for it to have some method for confidence estimation of its knowledge. This might serve as a form of self-assessment, which it can use to communicate to the human how confident the agent is in executing a plan which is according to the human's intention in the given scenario. Furthermore, it may be used to actively ask the user for more information, in the case it's confidence is low for a specific context, constraint or task.
- **Simplicity**: As stated by occam's razor, if presented with competing hypotheses which provide the same result, it is desirable to use the simpler hypothesis [75]. For machine learning, this can be understood as using the simplest model which classifies the data as good or better as more complex models. Furthermore, advantages of simpler AI algorithms are that they are often easier to understand and explain, compared to more complex techniques such as deep learning [17].
- **Transparency**: As the concept of human intention can be difficult to explicitly define, it is important for the model to be transparent in how it comes to its conclusions. A requisite for transparency is consistency, once trained and given the same task and context twice, the TCC learner should be consistent in its answers. Doing so improves

predictability to the user. Closely related is the field of XAI, which makes AI models more transparent by being able to explain its decisions. Models for which useful XAI methods are available are preferred.

- **Effective learning**: The model should be able to effectively learn to predict task constraints based on the context variables, and generalize this knowledge to new unforeseen situations. As the number of context variables in a real world scenario can be very large, the model is of little use if it needs to have encountered every situation once before being able to correctly predict the human's intention.

4.2.2 Implementation

4.2.2.1 Dataset

The dataset consists of data retrieved from the experiment described in the previous section, in which 3 participants (Section 4.1.3.1) performed the drone reconnaissance task (Section 4.1.3). For each participant, the data consists of 128 unique scenarios. Each scenario is defined as a unique combination of the 7 context features (Table 3). For each scenario, the 7 constraints (Table 4) are defined which best approximate the participants intention for the task. Thus in total this gives 3 subsets, each consisting of 128 datapoints, where each datapoint consists of 7 input features (contexts) and 7 to-be-predicted output features (constraints).

4.2.2.2 Features

All context input features are binary and nominal, in that they encode a context feature in the environment which is present or not.

The to-be-predicted constraint output features are slightly more complex. The constraints `drone_flying_speed`, `drone_notification_frequency`, `fly_village`, and `fly_lake` are binary classes. The constraints `time_limit`, `min_distance_tank` and `min_distance_radar` are multi-class features, providing a third possible class "None". Furthermore, `time_limit`, `drone_flying_speed` and `drone_notification_frequency` are ordinal features, in contrast to the `fly_village` and `fly_lake` features which are nominal features. The constraints `min_distance_tank` and `min_distance_radar` are a mix, having two ordinal classes and one nominal class.

4.2.2.3 Pipeline

The pipeline used for training the TCC learner entails in order: the loading of data, preprocessing and feature engineering, splitting data in a train and test set, training and optimization of the classifiers, and finally evaluation and plotting of results.

During feature engineering the categories of each feature listed in the previous section were encoded as numbers. Next, the dataset was randomly split into a training (75%)

and test (25%) set. As the dataset size is fairly small (128 datapoints per participant), a relatively large test set size was chosen to provide a more realistic indication of the model performance.

To improve model performance, a classifier was trained for every individual constraint. As such, in the pipeline the dataset was split 7 times, with each subset containing all context input features, and one of the to-be-predicted constraints.

For each subset the parameters of the classifier were optimized using stratified 5-fold cross-validation grid search, with F1-score and accuracy as a combined performance metric [76, 77, 78], implemented using the Scikit-learn framework [79]. After optimal parameters were set for each classifier, the results are 7 classifiers which can each predict a constraint for a scenario using the input context features. The models are then tested on the final validation set with as metrics accuracy, F1-score, and for binary constraints also using the area under the receiver operating characteristics curve (AUROC).

4.2.2.4 Models

K-Nearest Neighbour (KNN) classifier K-nearest neighbour classification is a simple, non-parametric, instance-based learning algorithm. The KNN algorithm is non-paramateric in that it makes no assumptions on the data distribution, in contrast to for instance linear regression models. This is the result of the manner in which a KNN learns, which is instance-based, or also called lazy learning. A KNN saves all training instances in memory in a feature space, with new instances being classified using a plurality vote based on its neighbours k classes. In this manner, instead of fitting a target function for the whole dataset, the function is only locally approximated for each new instance. As such, a KNN does not create a generalization of all its training data, and also has no explicit training phase.

The KNN algorithm meets the model requirements in that it is an efficient, effective and robust learning algorithm due to its is non-paramateric and instance-based learner characteristics. Furthermore, it is a simple model for which various algorithm-specific confidence measures exist [80], as well as model-agnostic confidence measures [81], and it is fairly easy to interpret as the classification can be directly explained with regard to its similarity to similar instances.
During the gridsearch, the following model parameters were optimized: the number of neighbours k to use for classifying new instances, the function determining the weight of neighbours, and finally the algorithm used to find the nearest neighbours.

XGBoost The XGBoost algorithm, standing for Extreme Gradient Boosting, is a popular open-source implementation of the gradient boosting algorithm [82]. Gradient boosting is a technique which creates an ensemble of weak prediction models, decision trees in the case of XGBoost, and combines these to improve model performance. In gradient boosting decision trees are built sequentially, at each step fitting a decision tree to the data, after which the next model is fit to the residual error from the first model, iteratively creating decision trees to reduce the residual error until there is no error left or the maximum number of models is

reached. XGBoost in specific is an open-source implementation that offers high scalability, fast training speeds, and various optimizations such as regularization which avoid overfitting [82] .

XGBoost fits the model requirements in that it has high predictive power and has been applied successfully to a large number of classification tasks, making it highly possible to perform well for this task compared to other algorithms as well. Furthermore, it is robust to outliers, simple as it consists of shallow decision trees, and shown to be a data-efficient and effective machine learning model [82]. During the gridsearch, the following model parameters were optimized: the minimum sum of instance weight needed in a child node, gamma (minimum required loss reduction for another node split), and maximum tree depth.

4.2.2.5 Feature Importance

For every model trained on a constraint, the importance of features were calculated as the relative drop-column importance. That is, for each context feature $x \in X$ used by the $\mathcal{M}(T, X)$ model, the model is re-trained on the training data without the feature, such that the total context features used for training is $X' = X/\{x\}$ and the re-trained model is $\mathcal{M}'(T, X)$. The feature importance is then calculated as the contribution of the model to the baseline model accuracy: $I_x = acc(\mathcal{M}(T, X)) - acc(\mathcal{M}'(T, X))$. Subsequently the feature importances are normalized to be between 0 and 1, and sum to 1 for each $\mathcal{M}(T, X)$ model. The result is a feature importance which specifies the relative importance of each feature on the accuracy of a model, representing the importance of that feature and possible feature interactions with other features. Using the ranked features based on importance, insight can be derived in what knowledge the model extracts from the data to predict the constraint, making it more transparent.

4.3 Model Experiments

XGBoost and KNN models were trained on the dataset as described in the pipeline (Section 4.2.2.3) above, with a seperate model trained for predicting each individual constraint. Separate models were trained on data of each individual participant to arrive at a personalized TCC learning model, able to predict the human intentions for the task specific to that participant. This condition is called the individual-intention condition.

4.3.1 Context Dependency

An assumption made for the research question and approach described in the previous section is that the context is important for how a human wants an agent to perform a task.

To test if this assumption is correct, another approach will be tested as well: the models will be trained to predict the correct constraint option of a single participant, given no context information as input. To be more specific, the contexts will be set to the same value for every scenario in the train and test set. As such, the model cannot use the context information for predicting the correct constraint options, as it is always the same.

The results can then be compared between this no-context approach and the context-trained approach, to see what the impact is of context on the model scores. This approach is called the individual-no-context condition.

4.3.2 Universal Human Intentions

XGBoost and KNN models were also trained on a combination of participants data. This approach was performed to test all possible combinations of participants. Doing so might show if there are any universal human intentions for the task: an approach to the task which all participants agree on. If this is the case, data might be pooled to achieve higher model scores, or data acquisition can be split between different participants to ease the burden of data acquisition. This approach is called the universal-intention condition.

5 Results

In this section the results of the participant experiment and models experiments are discussed.

First, a general introduction is given on how the experiment was rated by the participants on clearness, intuitiveness, control and effort. Also, the task interpretations and the intentions for the agent of each participant are shown. That is, how did the participants describe in the questionnaires how they wanted the agent to behave.

Second, the results are introduced of the individual-intention condition: models trained on the participant data for predicting context-dependent constraints. The results are analyzed to see if the models were able to learn anything from the participant data, and what they exactly learned. As such, can these models learn the human intention using context-dependent constraints as an approximation?

Third, the models are examined more in depth as to determine if they meet the model requirements described in Section 4.2.1. Especially, do the models learn efficiently?

Next, the results of the individual-no-context condition is inspected, indicating how context-dependent the human intentions are for this experiment.

Finally, the results will be qualitatively analyzed to see if there is any overlap between the approaches of each participant, and inspect the scores of the models trained with the universal-intention condition: models trained on combinations of data from multiple participants. If so, these universal human intentions may be used to ease the burden of dataset generation and provide more data for more accurate models.

5.1 Experiment Qualitative Feedback

In Table 5 the results are shown for the post-experiment questionnaire. The results show the experiment was clear for participants. Furthermore, the approach using constraints was rated as being fairly intuitive and providing high control over the drone's behaviour. Finally, the effort required to go through all 128 scenarios was rated as high. Performing the experiment including the tutorial and questionnaires, took participants between 1 and 1.5 hours.

5.2 Participant Task Interpretation

Every participant wrote down prior to the experiment how they believed the task to change as a result of the various contexts. Below, for every participant a short description is given of the participant's interpretation of the task: how did they view the various contexts to change the task, and how should the drone change it's behaviour to adapt.

Questionnaire Question	Mean (0-10)	Standard Deviation
Was the experiment clear?	8.33	1.25
How intuitive did you find the usage of constraints to teach the drone how to best perform the task?	7.67	0.47
Level of control experienced over the drone using constraints?	9.00	0.82
How much effort did it take to teach the drone for all 128 scenarios?	8.33	0.47

Table 5: Results of the questionnaire. Questions were rated between 0 (less) and 10 (more), filled by each participant after the experiment.

5.2.1 Participant 1

Participant 1 interpreted the task and various contexts in a straightforward fashion. The participant hypothesized that during nighttime and fog the drone's visibility is impaired, as such caution and slow flying is desired. Furthermore, in the case of an incoming friendly VIP, the drone should adapt by flying faster. Also, risks should be avoided as much as possible, by keeping a high distance to enemy radars, enemy tanks, villages with hostiles, and when possible avoiding flying over water.

5.2.2 Participant 2

The task interpretation of participant 2 coincided with participant 1 in that tanks and radars should always be avoided, as well as that the flight speed of the drone should increase when a VIP is inbound. However, less importance was placed on the time of day and weather contexts, instead focusing on the type of environment in combination with present radars, tanks, or villages. Mountainous terrain was hypothesized to heighten the possibility of crashing due to uneven terrain, as such requiring increased caution: a higher notification frequency and slower flying. Radars were avoided in all cases as it was hypothesized that detection by a radar results in it contacting a tank which intercepts the drone on the way back. Villages were also to always be avoided, as the noise of the drone would annoy locals, and spies might be present in the village. Instead, the lake in the centre was interpreted as being a small stream which posed little danger, as such making it desirable to always fly over the water.

Aside from this, it was mentioned that an inbound VIP and mountainous environment should result in a longer time limit, as it might crash otherwise because it has to take risks and take shortcuts in the dangerous terrain. Also, many enemies should results in an increased notification frequency.

5.2.3 Participant 3

The final participant used a combination of both participants 1 and 2's reasoning. Villages had to be be avoided at night, as to not wake and annoy locals. Fog and mountainous posed increased crash risk, and thus required extra caution (slower flying). VIP's resulted in a heightened state of alert, requiring an increased notification frequency and flying speed, as well as increased privileges (flying over villages at night). Different from participant 1 and 2, participant 3 hypothesized the tank and radar to be stationed on the other side of a mountain in a mountainous environment, making it possible to closely pass these. Furthermore, during foggy weather closely passing the tank was hypothesized as being possible, as the tank relies on visual contact for detection. This was not the case for radars, as these do not require visual contact for detection.

Water was mentioned as a (low priority) potential danger, and as such that is should be avoided when possible (e.g. when not night or foggy). Furthermore, when there is much danger present (multiple types of hostiles, weather, etc.), a higher notification frequency was desired.

		Participant 1		Participant 2		Participant 3	
		XGBoost	KNN	XGBoost	KNN	XGBoost	KNN
Drone flying speed	F1	0.77	0.87	0.89	0.92	0.97	0.97
	AUROC	0.82	0.89	0.87	0.90	0.97	0.97
Notification frequency	F1	0.91	0.86	1.00	1.00	0.94	0.92
	AUROC	0.92	0.88	1.00	1.00	0.93	0.9
Fly village	F1	1.00	1.00	1.00	1.00	1.00	0.98
	AUROC	1.00	1.00	-	-	1.00	0.95
Fly water	F1	0.97	0.91	1.00	1.00	0.67	0.86
	AUROC	0.97	0.91	-	-	0.75	0.88
Distance to tank	F1	1.00	1.00	1.00	1.00	1.00	0.97
Distance to radar	F1	1.00	1.00	1.00	1.00	0.97	0.88
Time limit	F1	0.94	0.97	0.97	0.97	0.94	0.84
Averages	F1	0.94	0.94	0.98	0.98	0.93	0.92
	AUROC	0.93	0.92	0.93	0.95	0.91	0.92

Table 6: Scores for the F1 and AUROC metrics of the XGBoost and KNN models for predicting the constraints (row labels most left) as filled in during the drone reconnaissance task. Results are shown for models trained on the data of each individual participant, the individual-intention condition.

5.3 Learning Human Intention

Table 6 shows the metric performances of each model on predicting the valid option for one individual constraint for the drone reconnaissance. Results are shown for the XGBoost and KNN models trained on data of an individual participant. F1-scores are shown for every constraint, while AUROC scores are only shown for binary constraint types. Exceptions are the four missing AUROC scores for models trained on participant 2 data, which is caused by the participant always having chosen the same option for the `fly_village` and `fly_water` constraints, making it impossible to calculate the AUROC score.

Generally seen, models achieve high metric scores (> 0.90) for all constraints on average, both XGBoost as well as KNN models, when trained on data of either of the three participants.

A summary of the experiment and model results are shown in Figures 8, 9 and 10 for respectively participant 1, 2 and 3. These figures describe what context variables each

participant found important for a constraint as they described in the questionnaire (see Section 5.2), compared to what context features the model found most important, and finally the best model performance.

5.3.1 Participant 1

Inspecting the metric scores for the models trained on data of participant 1, Table 6 shows that both models achieve similar scores on average over all constraints (both F1-score of 0.94).

Figure 8 shows that the `fly_village`, `min_distance_radar`, and `min_distance_tank` constraints can be perfectly predicted, which is not surprising as these constraints were based on a single context feature.

A more interesting constraint is the `fly_water` constraint which was only vaguely described by the participant with "avoid if possible". The feature importances show that the most importance features were primarily the contexts `hostiles_in_the_village`, which would block the right path, and the `tank` or `radar` feature, which would block the left path. Using these features, the model was able to predict this constraint quite precisely (F1 0.97).

For the `drone_flying_speed` and `notification_frequency`, the model used more features than the human described they used (5 instead of 3 features, and 5 instead of 1 feature). Both features did achieve high scores, with a F1 score of respectively 0.98 and 0.91.

5.3.2 Participant 2

For participant 2, the most interesting constraint seems to be the `drone_flying_speed`, where the model required 6 of the total 7 context features for prediction. In addition to using the area type, weather, and VIP inbound context features which were also described by the participant, the model also used the time of day, presence of tank and presence of radar context features. The `flying_speed` constraint had a slightly lower score compared to the other constraints (F1 0.92 for KNN).

The `time_limit` was close to perfectly predicted, which might be explained by the low number of important contexts.

Finally, the models were able to achieve a perfect score for 5 of the 7 constraints: `notification_frequency`, `fly_village`, `fly_water`, `min_distance_tank`, `min_distance_radar`. These results are largely unsurprising, as there was little to no learning required for these constraints. The exception being the `notification_frequency` constraint, where the participant also described taking the number of enemies into account, however this was not found in the model's feature importances.

Constraint	Context influences (questionnaire)	Features best model	Scores best model
Drone flying speed	Nighttime (<) Fog (<) VIP (>)	**KNN** VIP (0.42) Nighttime (0.18) Area type (0.12) Weather (0.12) Hostiles in village (0.10) Tank (0.06)	**KNN** F1: 0.98 AUROC: 0.95
Notification frequency	Fog (>)	**XGBoost** Weather (0.36) Hostiles in village (0.36) Radar (0.14) Tank (0.07) VIP (0.07)	**XGBoost** F1: 0.91 AUROC: 0.92
Fly village	Hostiles (x)	**XGBoost** Hostiles in village (1.00)	**XGBoost** F1: 1.00 AUROC: 1.00
Fly water	Avoid if possible (low priority)	**XGBoost** Hostiles in village (0.60) Tank (0.23) Radar (0.17)	**XGBoost** F1: 0.97 AUROC: 0.97
Distance to tank	Tank present (>)	**XGBoost** Tank (1.00)	**XGBoost** F1: 1.00
Distance to radar	Radar present (>)	**XGBoost** Radar (1.00)	**XGBoost** F1: 1.00
Time limit		**KNN** VIP (.078) Nighttime (0.09) Area type (0.04) Tank (0.04) Hostiles in village (0.04)	**KNN** F1: 0.97

Figure 8: **Participant 1**: Summary of the experiment and model results. In the context influences column, the >, <, x signs describe the relation to the constraint, meaning respectively: an increase of the constraint option (e.g. flying speed), a decrease, and a prohibition. Model features are described as the relative feature importance for the accuracy of the model.

Constraint	Context influences (questionnaire)	Features best model	Scores best model
Drone flying speed	Mountainous (<) Fog (<) VIP (>)	**KNN** Area type (0.57) VIP (0.12) Weather (0.10) Nighttime (0.10) Tank (0.08) Radar (0.02)	**KNN** F1: 0.92 AUROC: 0.90
Notification frequency	Mountainous (>) VIP (>) Many enemies (>)	**XGBoost** VIP (0.52) Area type (0.48)	**XGBoost** F1: 1.00 AUROC: 1.00
Fly village	Always prohibit	**XGBoost** -	**XGBoost** F1: 1.00 AUROC: -
Fly water	Always allow	**XGBoost** -	**XGBoost** F1: 1.00 AUROC: -
Distance to tank	Tank (>)	**XGBoost** Tank (1.0)	**XGBoost** F1: 1.00
Distance to radar	Radar (>)	**XGBoost** Radar (1.0)	**XGBoost** F1: 1.00
Time limit	VIP (<) VIP + Mountainous (>)	**XGBoost** VIP (0.61) Area type (0.39)	**XGBoost** F1: 1.00

Figure 9: **Participant 2**: Summary of the experiment and model results. In the context influences column, the >, <, x signs describe the relation to the constraint, meaning respectively: an increase of the constraint option (e.g. flying speed), a decrease, and a prohibition. Model features are described as the relative feature importance for the accuracy of the model.

Constraint	Context influences (questionnaire)	Model features	Scores
Drone flying speed	VIP (>) Fog (<)	**XGBoost** Weather (1.0)	**XGBoost** F1: 0.97 AUROC: 0.97
Notification frequency	VIP (>) Fog (>)	**XGBoost** Weather (0.41) Hostiles in village (0.25) VIP (0.09) Nighttime (0.09) Tank (0.09)	**XGBoost** F1: 0.94 AUROC: 0.93
Fly village	Hostiles in village (x) Nighttime (x) Fog (x)	**XGBoost** Hostiles in village (0.70) VIP (0.20) Nighttime (0.10)	**XGBoost** F1: 1.00 AUROC: 1.00
Fly water	Fog (x) Avoid if possible (low priority)	**KNN** Weather (0.44) Hostiles in village (0.25) Nighttime (0.19) Tank (0.06) VIP (0.06)	**KNN** F1: 0.86 AUROC: 0.88
Distance to tank	Tank (<) Nighttime (>) Mountainous (<) Fog (<)	**XGBoost** Tank (0.75) Area type (0.16) Weather (0.09)	**XGBoost** F1: 1.00
Distance to radar	Nighttime (>) Mountainous (<) Radar (<)	**XGBoost** Radar (0.67) Area type (0.33)	**XGBoost** F1: 1.00
Time limit	VIP (<)	**XGBoost** VIP (0.61) Hostiles in village (0.21) Weather (0.18)	**XGBoost** F1: 0.94

Figure 10: **Participant 3**: Summary of the experiment and model results. In the context influences column, the >, <, x signs describe the relation to the constraint, meaning respectively: an increase of the constraint option (e.g. flying speed), a decrease, and a prohibition. Model features are described as the relative feature importance for the accuracy of the model.

5.3.3 Participant 3

The summarized results of participant 3 in Figure 10 shows a number of interesting results, with mismatches between what was described in the questionnaire and what the model has learned. However, this mismatch did not prevent the model from achieving a high average F1-score (XGBoost 0.93, KNN 0.92). The only exception is the `fly_water` constraint where both constraints dip to scores below 0.90. Inspecting the participant note, `weather` is described as the most important influence for this constraint, followed by the vague description that it should be "avoided when possible". In addition to the contexts described by the participant, a number of other features were used as well.

The mismatches consist of three types: 1) The model not using features described by the participant, 2) the model using extra features, or 3) a combination of both.

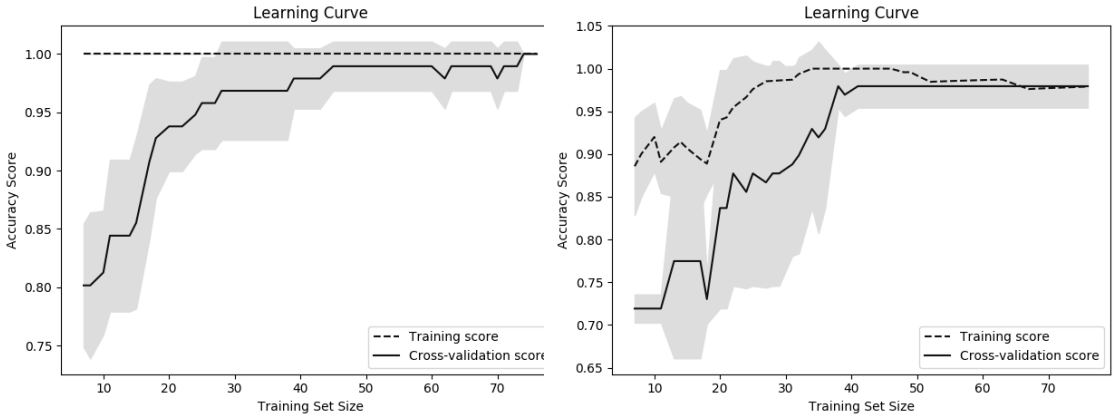
The `flying_speed` constraint is part of the first category: the model misses the `VIP` context feature, which was described by the participant as important. The `min_distance_tank` and `min_distance_radar` constraints also belong to this group, both missing the `nighttime` context in the model.

For the `fly_village` constraint a swap has been performed, where the `weather` context that was described as important by the participant was swapped for using the `VIP` context, which enables the model to achieve a perfect prediction score.

Similar to the `notification_frequency`, the `fly_water` and `time_limit` constraints are predicted by the models using a number of extra context features not described by the participant. The `fly_water` constraint in particular is interesting, as the model has learned that the "avoid when possible" description of the participant is related to the `hostiles_in_village`, `nighttime`, `tank`, and `VIP` context features. This corresponds to the participant's description, which described water as a potential danger in poor visibility (fog/nighttime), but was lower priority than other dangers.

5.4 Learning efficiency

To identify the learning efficiency of the models, for every model trained on an individual participant, the learning curves were plotted. Manual inspection showed that more than 90% of the models achieved a stable cross-validation score (during training) after training on just 50 datapoints. After these 50 datapoints the models improved only marginally. Two learning curves are shown in Figure 11.



(a) Learning curve for the KNN model trained on predicting the `tank_distance` constraint for participant 1.

(b) Learning curve for the XGBoost model trained on predicting the `notification_frequency` constraint for participant 2.

Figure 11: The learning curves of two separate models trained on predicting constraints of an individual participant, indicating the performance of the model for various dataset sizes.

5.5 Context Dependency

Table 7 shows the average scores for models trained on predicting the constraints with the individual-no-context condition. For this condition, models were trained with uniform contexts as input for all scenarios, such that the contexts had no predictive value. Table 9 in the appendix shows the results for every individual constraint as well.

Comparing the results of the individual-no-context condition (Table 7) to the results of the individual-intention in Table 6, it can be seen that the individual-no-context condition models score considerably lower than the context condition. An AUROC score of 0.50 indicates chance-level model performance (for the binary constraints). The F1-score is also considerably lower than the conditions with context.

An interesting result are the high F1-scores for participant 2. Examining the individual scores for the no-context condition (Table 9 in the appendix), shows the cause to be the `fly_village` and `fly_water` constraints, for which always the same constraint option was chosen. As such, these two constraints could be perfectly predicted even without context information, increasing the average F1-score.

5.6 Universal Human Intentions

In Table 8 the results are shown for training the KNN and XGBoost models on combinations of participant data.

		Participant 1		Participant 2		Participant 3	
		XGBoost	KNN	XGBoost	KNN	XGBoost	KNN
Averages	F1	0.44	0.62	0.69	0.69	0.49	0.49
	AUROC	0.50	0.50	0.50	0.50	0.50	0.50

Table 7: Scores of the XGBoost and KNN models for the individual-no-context condition. For this condition, models were trained to predict the constraints with no context variables. Thus leading to the model only being able to learn one option for the complete task. Results are shown for models trained on the data of each individual participant.

As can be seen from the average scores, performances are much lower compared to the models trained on an individual participant (Table 6).

For some specific constraint models, performance decreases even to worse than chance level (AUROC <0.5), such as the combinations of P1+P3 and P2+P3 for the `fly_village` constraint. Analyzing the intentions of these participants in Figures 8, 9 and 10, shows that all three participants have very different approaches to when to set these constraints. Participant 2 always chose the same constraint option no matter the context, whereas participant 1 and 3 have multiple, overlapping but also differing contexts which they found important. Comparing these figures, a similar analysis can be made for other poorly scoring constraints, such as the `time_limit` and `drone_flying_speed` constraint.

Figure 12 shows the learning curve for the XGBoost model, trained on data of all participants for predicting the `time_limit` constraint. As can be seen in the figure, the increase in the cross-validation score is almost non-existing as more data is fed to the model. Furthermore, the training score decreases as the model is trained on more data. This means the model is unable to learn a good fit for the dataset. Either caused by the model not being complex enough [83], or caused by the dataset not being completely consistent or logical, making it impossible to fit a perfect model on the data.

		P1 + P2		P1 + P3		P2 + P3		P1 + P2 + P3	
		XGBoost	KNN	XGBoost	KNN	XGBoost	KNN	XGBoost	KNN
Drone flying speed	F1	0.60	0.71	0.71	0.67	0.70	0.73	0.71	0.68
	AUROC	0.58	0.69	0.75	0.71	0.67	0.66	0.70	0.68
Notification frequency	F1	0.88	0.86	0.83	0.80	0.84	0.81	0.82	0.79
	AUROC	0.67	0.65	0.76	0.73	0.70	0.67	0.73	0.68
Fly village	F1	0.60	0.38	0.75	0.48	0.08	0.29	0.74	0.63
	AUROC	0.80	0.59	0.5	0.42	0.49	0.48	0.79	0.71
Fly water	F1	0.90	0.87	0.17	0.07	0.65	0.55	0.77	0.69
	AUROC	0.50	0.54	0.52	0.45	0.47	0.45	0.70	0.57
Distance to tank	F1	1.00	1.00	0.84	0.80	0.84	0.80	0.89	0.82
Distance to radar	F1	1.00	1.00	0.88	0.80	0.88	0.80	0.91	0.81
Time limit	F1	0.61	0.58	0.58	0.48	0.77	0.81	0.65	0.61
Averages	F1	0.80	0.77	0.68	0.59	0.68	0.69	0.78	0.72
	AUROC	0.64	0.62	0.63	0.58	0.58	0.56	0.73	0.66

Table 8: Scores for the F1 and AUROC metrics of the XGBoost and KNN models for predicting the constraints (row labels most left) as filled in during the drone reconnaissance task. Results are shown for models trained on a combination of participant data (the universal-intentions condition), indicated at the column labels (e.g. P1 = participant 1).

Figure 12: The learning curve for the XGBoost model, trained on data of all participants for predicting the `time_limit` constraint.

6 Discussion

Having AI systems do what we want them to do has been a longstanding challenge in Artificial Intelligence, kindled in recent years by the discussion around autonomous weapon systems, meaningful human control, and AI safety. Tasking is a direct form of agent directability, studied in this thesis, where the difficulty is in being able to task an agent efficiently with low effort, while the agent performs the task in a way that the human intended.

In this thesis a method has been explored leveraging rules and learning, as to combine the advantages of both the data-driven and rule-based approaches. It has been shown that it is possible to have humans specify their intentions for a task as constraints in various contexts, the task and environment being simulated in a 2D environment, subsequently with a model learning this data as to predict personalized context-dependent task constraints. The model as such learns an approximation to the human intention for a task.

6.1 Representing Human Intentions

The goal of this thesis was to answer the research question: How can an agent learn context-dependent task constraints provided by the human task instructor, as to perform the task as the human intended? To answer this question, a set of smaller subquestions were asked.

First was studied how the agent can identify the human intention during agent tasking (SQ1). The method explored in this thesis was to provide the agent with information on the task and context, in combination with the human intention encoded (by the human) as task constraints. The goal was then for the agent to learn to predict these constraints based on the task and context. By using constraints which relate to the agent's behaviour, setting prohibitions, obligations, permissions, settings, etcetera, the human can specify their intention for the agent's behaviour with the constraints. In this way, the task constraints can be used by the agent as an approximation to the human's intention (SQ 2).

Building a dataset from which an agent can learn the human intention for a task (SQ 5) was done by setting up an experiment with a simulator, simulating a task in various contexts, and having the participants specify their intentions as task constraints. The context of the task changed for every trial by tweaking 7 context variables, such as the weather or time of day.

Participants reported it being clear what they had to do during the experiment.

Regarding the constraint-based approach, participants reported feeling in control over the agents behaviour through the use of constraints during the experiment, as well as it being an intuitive method for influencing the agent's behaviour.

This finding is in line with prior research that also underpin the intuitiveness of constraints for communicating desired agent behaviour, and the control over the agent when using constraints or closely related rule-based approaches such as work agreements and policies [51, 47, 35].

A possible alternative explanation for participants reporting as feeling in control can be attributed to the experiment being specified very loosely, where the participants did not see the resulting agent behaviour based on their specified constraints. As such, they might have imagined the agent to perfectly follow their intentions, which in reality might not have been the case when an agent actually executes the task according to the imposed constraints. This approach was chosen as otherwise an agent would have had to be build which can incorporate the constraints in its behaviour, which in combination with time restrictions was not feasible for this study.

In total, each participant had to specify their intention for 128 scenarios during the experiment, which the participants reported as requiring much effort .

Gathering enough data in a human-friendly way is a major challenge in the field of machine learning, especially for deep learning, where large amounts of data are required to train an accurate model. For some tasks which require expensive oversight (see Section 2.1.4), this becomes a difficult issue. As can be seen from this study, even with a model requiring only 128 datapoints to become accurate, data acquisition already becomes an issue, especially when scaling the experiment to more complex tasks and environments. Active learning and interactive learning are promising methods that tackle this issue by taking a more active learning approach where the student (the agent) can query the teacher (the human) at each point during the learning process and can thus ask for specific information which is unclear to accelerate the learning process [19, 84].

6.2 Learning Human Intentions

Next, the question was asked how a model could learn these context-dependent constraints for a task. This was achieved by tackling the problem as a classification task. Using the experiment data, machine learning models were trained to predict for an individual constraint which constraint option the human most likely would have chosen for the scenario. This was done using the context features as input.

The results show that this personalized approach, training a model on data of a single participant, was highly successfully, with an average F1 score over all participants of 0.95. On multiple occasions, models were able to perfectly predict the constraints. Furthermore, this F1-score does not take into account if errors were made by the model or are the result of mistakes by the human. A machine learning model tries to fit the data and learn underlying rules. The model is unable to predict inconsistent answers of the human (not following their own definition of what would be desirable in a specific scenario), and as such, every mistake made by the human reduces the maximum model accuracy with a set percentage. A single inconsistent answer in the test set (32 datapoints) for the dataset used in this thesis reduces the maximum model accuracy attainable with 3.1%. Meaning, the achieved score of 0.95 may possibly be the maximum attainable score for a non-overfitting machine learning model.

An assumption for this thesis was that human intentions are context-dependent. This was tested by providing the models with uniform context input features during training, making the context uninformative. The results clearly show the context-dependency, where

the no-context models achieve an average F1-score of only 0.60, significantly lower than the personalized approach with context features.

An interesting concept for agent tasking is that of universal human intentions, where multiple people want the agent to complete the task in an identical or at least similar manner. If these universal human intentions exist, the creation of a dataset for a model which can learn human task intentions would be much easier, as data acquisition can be spread over multiple people easing the effort required per participant, or have all participants perform the complete experiment and gathering more data to train the models. In this thesis, models were trained on all possible combinations of the 3 participants. The results showed that the human participants from this pilot study were largely misaligned in their intentions. Average F1-scores vary between 0.59 and 0.80 for various participant combinations, sometimes even being outperformed by the no-context models. The results do show participants having aligned intentions for some (very) simple constraints. However, the results also show that universal human intentions can work counter-productive if human intentions are not closely aligned. A number of models have F1-scores of just 0.08 and an AUROC score of sub-0.50 due to clashing intentions of participants, making these models worse than random models.

A possible explanation for the lack of universal human intentions for the experiment conducted in this thesis, is that there are no set consequences which are enforced for the contexts and constraints. This is partially because of the abstract nature of the simulation, and partially because of the experimental setup. As the simulator provides an abstract representation of the environment, the exact details can be filled in by the participants themselves. For instance, one participant interpreted the lake in the middle of the area as a small (mountain) stream, while another participant interpreted it as a large lake which would pose considerably more difficulty and danger for the drone when crossing. Also, the experiment only included the initial scenario and the drone did not actually execute the task using the human imposed constraints. As such, the exact range of the tank, the exact length of a short time limit, or other details and consequences of the context variables and constraints were left to the imagination of the participant. This was done on purpose, to get a large variety of human intentions and see if the model still works, but such a loosely defined experiment is less suitable for identifying universal human intentions.

The issue of learning a task from multiple people is also a known open issue in the machine learning community, for instance with Learning from Demonstration (LfD) methods [85]. An example LfD approach is inverse reinforcement learning, where the model tries to learn a reward function from expert demonstrations [60]. The issue of clashing intentions (and thus the constraints as in this thesis) of multiple people is circumvented when using inverse reinforcement learning, as the model does not copy the exact behaviour, but only the factors which they find important and encode that in a reward function. In this manner, two people who find the same things important but achieve them using a different approach, can still be used together to train the same model [86]. Unfortunately, these methods suffer from other downsides, as explained in Section 2.3.

For this problem domain, various desiderata have been identified for a model learning

the human intention for agent tasking. These being, high model performance, efficient learning, robustness, confidence estimation, simplicity, transparency, and effective learning. See Section 4.2.1 for a more in-depth discussion on these requirements.

These model requirements are based on a variety of related studies, such as interactive task learning and AI safety [19, 11].

6.3 Limitations

A limitation of the executed experiment is that the number of participants was low (n=3), as such making it difficult to generalize the results. Furthermore, one might argue that the number of context variables and possible constraints is also fairly small. These factors are mainly due to the nature of this study, a pilot study which identifies the feasibility of the chosen approach. Nevertheless, with this limited number of contexts and constraints, already a rich variety of human intentions can be encoded, which can be seen from all participants having a different intention for how the agent should solve the task. A limitation of the used learning approach where each model is trained for predicting individual constraints, is that interdependencies between constraints cannot be learned, as well as that continuous contexts or constraints cannot be learned.

A more conceptual issue is the assumption that the context of a task in the real-world can be represented sufficiently using a finite (small) number of context variables. Rule-based approaches rely on programming for the capability of recognizing these context variables. However, the real-world can be unpredictable, complex and unstructured, leading to a very large number of context variables which might influence the human's intention for the task [18]. Making a robust AI-system which does what you want it to do while dealing with unexpected circumstances in an complex environment is hypothesized to be an AI-complete problem [87]. Similar difficulties can be found for deep learning approaches, where scaling up the system to a real-world environment leads to unreliable behaviour, and shows the brittleness of current AI-systems [17]. As such, truly solving agent tasking might be an AI-complete problem as well.

7 Conclusion

An agent that can learn context-dependent task constraints provided by a human enables the efficient taskability of data-driven methods, while also providing the directability and control of rule-based tasking methods. In this thesis has been shown that such an approach is feasible: a trained model can accurately predict the context-dependent constraints for a task in various contexts, as an approximation to the human intention. Furthermore, context has been shown to have a large influence on how a human wants an agent to complete a task. Although the conclusiveness of the results is lower due to the magnitude of the experiment, the results show this to be a promising approach for establishing meaningful human control over agents.

7.1 Future Research

Future work can be roughly divided into two categories. The first category deals with improving the agent-tasking approach proposed in this thesis: learning rule-based agent behaviour restrictions. The second category improves the implementation and testing of the approach in practice.

7.1.1 Context-Constraint-Task Learning Improvements

Active Learning Active learning could be used in two ways to drastically improve the usability of the taken agent-tasking approach. First, using (inter)active learning, the model would be able to deal with failures caused by unpredictable, complex or unknown environments, by asking the human for help. For instance, the human could explicitly communicate their intention for the task, or when other knowledge is missing teach new context features or constraints on the fly. Although this is less efficient for the user, it does guarantee a safe and reliable fallback method for when the model fails. Secondly, active learning could also be used for continuous learning, training the model after every new scenario. Doing so, the model would become useful for the human at a much earlier stage in the learning process and is able to for instance provide the human with suggestions.

Confidence Estimate Closely related is the need for a confidence estimation capability of the model. Confidence estimates of constraint predictions could make the model more transparent towards the human, and make humans able to take a more informed choice. Furthermore, the confidence estimate can also be used by the agent itself to pinpoint gaps in its knowledge, asking the human for extra data on these topics.

Preferences For this thesis constraints were used, as they are an intuitive and easy method for restricting the agent's behaviour. However, an issue with constraints is that they are a rather rough way to describe real-life problems. Switching to a tasking-method which allows the user to specify preferences, such as work agreements or soft constraints, may be a more

natural method compared to strict requirements.

Action-level planning Finally, an interesting alternative approach would be to change the agent's learning process from the plan level to the action level. Instead of the human specifying their intention for the agent on the plan level (execute complete plan with these restrictions), the human can specify at specific points during the task which action the agent may take, effectively specifying the human intention on the action level.

7.1.2 Implementation Improvements

Experts on Real-World Task The first major improvement which can be made, is to take a real-world task, scale it back to a simulation, and let a number of experts specify their intentions for how it should be performed. For instance, a firefighting scenario performed by a number of trained firefighters. Doing so, the learning effect of participants during the experiment can be minimized. Furthermore, identically trained experts might be a good test to see if universal human intentions do exist in practice or not. In addition, it can be tested whether it is possible to represent such a real-world task using contexts and constraints realistically, evaluated by domain experts. Also, the number of participants in the experiment should preferably be above 50, as to have enough statistical power to backup the results.

After training on the task, it would be very interesting to see the capability of the model to transfer it's learning to a new similar task.

Apply to Agent As the goal of the human intention learning model is to improve taskable agents, it is recommended to combine the model with an agent that plans the task taking the human-provided constraints into account, and then executes it. Showing the result to the human makes them able to better estimate the agent's capabilities and the result of constraints and contexts on the agent's behaviour.

8 References

[1] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[2] P. Langley, J. E. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, 2009.

[3] G. J. Schaub Jr and J. W. Kristoffersen, "In, on, or out of the loop?: Denmark and autonomous weapon systems," 2017.

[4] D. S. Board, "Defense science board summer study on autonomy," 2016.

[5] R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen, "Search and rescue robotics," in *Springer handbook of robotics*, pp. 1151–1173, Springer, 2008.

[6] L. E. Parker and J. V. Draper, "Robotics applications in maintenance and repair," *Handbook of industrial robotics*, vol. 1378, 1998.

[7] N. Bostrom, "Ethical issues in advanced artificial intelligence," *Science Fiction and Philosophy: From Time Travel to Superintelligence*, pp. 277–284, 2003.

[8] N. Bostrom, *Superintelligence*. Dunod, 2017.

[9] T. M. VII, "The first level of super mario bros. is easy with lexicographic," 2013.

[10] S. Russell, D. Dewey, and M. Tegmark, "Research priorities for robust and beneficial artificial intelligence," *Ai Magazine*, vol. 36, no. 4, pp. 105–114, 2015.

[11] H. M. Roff and R. Moyes, "Meaningful human control, artificial intelligence and autonomous weapons," in *Briefing Paper Prepared for the Informal Meeting of Experts on Lethal Au-Tonomous Weapons Systems, UN Convention on Certain Conventional Weapons*, 2016.

[12] M. Horowitz and P. Scharre, *Meaningful human control in weapon systems: a primer*. Center for a New American Security, 2015.

[13] F. Santoni de Sio and J. Van den Hoven, "Meaningful human control over autonomous systems: A philosophical account," *Frontiers in Robotics and AI*, vol. 5, p. 15, 2018.

[14] J. van Diggelen, "Dr jurriaan van diggelen (tno) - meaningful human control over ai-based systems."

[15] K. Christoffersen and D. D. Woods, "How to make automated systems team players," in *Advances in human performance and cognitive engineering research*, pp. 1–12, Emerald Group Publishing Limited, 2002.

[16] J. M. Bradshaw, V. Dignum, C. Jonker, and M. Sierhuis, "Human-agent-robot teamwork," *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 8–13, 2012.

[17] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018.

[18] P. Werkhoven, L. Kester, and M. Neerincx, "Telling autonomous systems what to do," in *Proceedings of the 36th European Conference on Cognitive Ergonomics*, p. 2, ACM, 2018.

[19] J. E. Laird, K. Gluck, J. Anderson, K. D. Forbus, O. C. Jenkins, C. Lebiere, D. Salvucci, M. Scheutz, A. Thomaz, G. Trafton, *et al.*, "Interactive task learning," *IEEE Intelligent Systems*, vol. 32, no. 4, pp. 6–21, 2017.

[20] J. Morley, "Meaningful human control in weapons systems: A primer," *Arms Control Today*, vol. 45, no. 4, p. 7, 2015.

[21] N. J. Smelser, P. B. Baltes, *et al.*, *International encyclopedia of the social & behavioral sciences*, vol. 11. Elsevier Amsterdam, 2001.

[22] J. M. Bradshaw, P. J. Feltovich, H. Jung, S. Kulkarni, W. Taysom, and A. Uszok, "Dimensions of adjustable autonomy and mixed-initiative interaction," in *International Workshop on Computational Autonomy*, pp. 17–39, Springer, 2003.

[23] A. HLEG, "Ethics guidelines for trustworthy ai," 2019.

[24] A. Asilomar, "Principles.(2017)," in *Principles developed in conjunction with the 2017 Asilomar conference [Benevolent AI 2017]*, 2018.

[25] X. Qiang, "The road to digital unfreedom: President xi's surveillance state," *Journal of Democracy*, vol. 30, no. 1, pp. 53–67, 2019.

[26] J. Altmann, P. Asaro, N. Sharkey, and R. Sparrow, "Armed military robots," *Ethics and Information Technology*, vol. 15, no. 2, pp. 73–76, 2013.

[27] T. G. Dietterich and E. Horvitz, "Rise of concerns about ai: Reflections and directions.," *Commun. ACM*, vol. 58, no. 10, pp. 38–40, 2015.

[28] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[29] J. Clark and D. Amodei, "Faulty reward functions in the wild, 2016," *URL https://blog. openai. com/faulty-reward-functions*.

[30] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2067–2069, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[31] J. Lehman, J. Clune, D. Misevic, C. Adami, L. Altenberg, J. Beaulieu, P. J. Bentley, S. Bernard, G. Beslon, D. M. Bryson, *et al.*, "The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities," *arXiv preprint arXiv:1803.03453*, 2018.

[32] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE transactions on evolutionary computation*, vol. 9, no. 6, pp. 653–668, 2005.

[33] J. van Diggelen, R. Looije, J. van der Waa, and M. Neerincx, "Human robot team development: An operational and technical perspective," in *International Conference on Applied Human Factors and Ergonomics*, pp. 293–302, Springer, 2017.

[34] R. Parasuraman, M. Barnes, K. Cosenzo, and S. Mulgund, "Adaptive automation for human-robot teaming in future command and control systems," tech. rep., Army research lab aberdeen proving ground md human research and engineering …, 2007.

[35] T. Mioch, M. M. Peeters, and M. A. Nccrincx, "Improving adaptive human-robot cooperation through work agreements," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 1105–1110, IEEE, 2018.

[36] G. Klien, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich, "Ten challenges for making automation a" team player" in joint human-agent activity," *IEEE Intelligent Systems*, vol. 19, no. 6, pp. 91–95, 2004.

[37] S. Weinschenk and D. T. Barker, *Designing Effective Speech Interfaces*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[38] S. J. Lackey, D. J. Barber, and S. G. Martinez, "Recommended considerations for human-robot interaction communication requirements," in *International Conference on Human-Computer Interaction*, pp. 663–674, Springer, 2014.

[39] J. Y. Chai, M. Cakmak, and C. Sidner, "Teaching robots new tasks through natural interaction," in *Interactive Task Learning: Agents, Robots, and Humans Acquiring New Tasks through Natural Interactions, Strüngmann Forum Reports, J. Lupp, series editor*, vol. 26, 2017.

[40] J. M. Bradshaw, P. J. Feltovich, M. J. Johnson, L. Bunch, M. R. Breedy, T. Eskridge, H. Jung, J. Lott, and A. Uszok, "Coordination in human-agent-robot teamwork," in *2008 International Symposium on Collaborative Technologies and Systems*, pp. 467–476, IEEE, 2008.

[41] A. K. Dey, "Understanding and using context," *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[42] M. Bratman, *Intention, plans, and practical reason*, vol. 10. Harvard University Press Cambridge, MA, 1987.

[43] A. S. Rao, M. P. Georgeff, *et al.*, "Bdi agents: from theory to practice.," in *ICMAS*, vol. 95, pp. 312–319, 1995.

[44] B. Logan, "Future directions in agent programming," *ALP Issue*, vol. 29, no. 4, 2017.

[45] D. S. Lange and R. S. Gutzwiller, "Human-autonomy teaming patterns in the command and control of teams of autonomous systems," in *International Conference on Engineering Psychology and Cognitive Ergonomics*, pp. 179–188, Springer, 2016.

[46] R. S. Gutzwiller, S. H. Espinosa, C. Kenny, and D. S. Lange, "A design pattern for working agreements in human-autonomy teaming," in *International Conference on Applied Human Factors and Ergonomics*, pp. 12–24, Springer, 2017.

[47] J. M. Bradshaw, P. Feltovich, and M. Johnson, "Human-agent interaction," *Handbook of Human-Machine Interaction*, pp. 283–302, 2011.

[48] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott, "Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," in *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, pp. 93–96, IEEE, 2003.

[49] E. C. Freuder, "In pursuit of the holy grail," *Constraints*, vol. 2, no. 1, pp. 57–61, 1997.

[50] R. Barták, "Constraint programming: In pursuit of the holy grail," in *Proceedings of the Week of Doctoral Students (WDS99)*, vol. 4, pp. 555–564, MatFyzPress Prague, 1999.

[51] E. C. Freuder, "Progress towards the holy grail," *Constraints*, vol. 23, no. 2, pp. 158–171, 2018.

[52] F. Rossi and A. Sperduti, "Acquiring both constraint and solution preferences in interactive constraint systems," *Constraints*, vol. 9, no. 4, pp. 311–332, 2004.

[53] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[54] C. Bessiere, R. Coletta, B. O'Sullivan, M. Paulin, *et al.*, "Query-driven constraint acquisition.," in *IJCAI*, vol. 7, pp. 50–55, 2007.

[55] C. Bessiere, A. Daoudi, E. Hebrard, G. Katsirelos, N. Lazaar, Y. Mechqrane, N. Narodytska, C.-G. Quimper, and T. Walsh, "New approaches to constraint acquisition," in *Data mining and constraint programming*, pp. 51–76, Springer, 2016.

[56] J. Charnley, S. Colton, and I. Miguel, "Automatic generation of implied constraints," in *ECAI*, vol. 141, pp. 73–77, 2006.

[57] J. R. Kirk and J. E. Laird, "Learning task formulations through situated interactive instruction," in *Proceedings of the Second Annual Conference on Advances in Cognitive Systems (ACS)*, vol. 219, p. 236, 2013.

[58] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31, JMLR. org, 2017.

[59] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, 2010.

[60] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, ACM, 2004.

[61] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

[62] B. van der Vecht, J. van Diggelen, M. Peeters, J. Barnhoorn, and J. van der Waa, "Sail: a social artificial intelligence layer for human-machine teaming," in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pp. 262–274, Springer, 2018.

[63] J. H. Moor, "The nature, importance, and difficulty of machine ethics," *IEEE intelligent systems*, vol. 21, no. 4, pp. 18–21, 2006.

[64] G. Irving and A. Askell, "Ai safety needs social scientists," *Distill*, vol. 4, no. 2, p. e14, 2019.

[65] H. Nakashima, H. Aghajan, and J. C. Augusto, *Handbook of ambient intelligence and smart environments*. Springer Science & Business Media, 2009.

[66] M. E. Bratman, "What is intention," *Intentions in communication*, pp. 15–31, 1990.

[67] D. Kahneman, S. P. Slovic, P. Slovic, and A. Tversky, *Judgment under uncertainty: Heuristics and biases*. Cambridge university press, 1982.

[68] A. Tversky and E. Shafir, "The disjunction effect in choice under uncertainty," *Psychological science*, vol. 3, no. 5, pp. 305–310, 1992.

[69] M. Johnson, C. Jonker, B. Van Riemsdijk, P. J. Feltovich, and J. M. Bradshaw, "Joint activity testbed: Blocks world for teams (bw4t)," in *International Workshop on Engineering Societies in the Agents World*, pp. 254–256, Springer, 2009.

[70] T. Balch, "Overview for teambots 2.0." `https://www.cs.cmu.edu/~trb/TeamBots/`, 1998.

[71] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "Mason: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 2005.

[72] J. Linietsky and A. Manzur, "Godot game engine." `https://godotengine.org/`, 2014.

[73] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International conference on complex systems*, vol. 21, pp. 16–21, Boston, MA, 2004.

[74] J. Boyd, *A discourse on winning and losing.* Air University Press, Curtis E. LeMay Center for Doctrine Development and …, 2018.

[75] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," *Information processing letters*, vol. 24, no. 6, pp. 377–380, 1987.

[76] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, Montreal, Canada, 1995.

[77] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[78] Y. Sasaki *et al.*, "The truth of the f-measure," *Teach Tutor mater*, vol. 1, no. 5, pp. 1–5, 2007.

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[80] S. J. Delany, P. Cunningham, D. Doyle, and A. Zamolotskikh, "Generating estimates of classification confidence for a case-based spam filter," in *International conference on case-based reasoning*, pp. 177–190, Springer, 2005.

[81] J. van der Waa, J. van Diggelen, M. A. Neerincx, and S. Raaijmakers, "Icm: An intuitive model independent and accurate certainty measure for machine learning.," in *ICAART (2)*, pp. 314–321, 2018.

[82] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, ACM, 2016.

[83] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[84] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty.," in *Robotics: Science and Systems*, vol. 3, pp. 321–328, 2007.

[85] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[86] P. S. Castro, S. Li, and D. Zhang, "Inverse reinforcement learning with multiple ranked experts," *arXiv preprint arXiv:1907.13411*, 2019.

[87] R. V. Yampolskiy, "Ai-complete, ai-hard, or ai-easy–classification of problems in ai," in *The 23rd Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, OH, USA*, 2012.

9 Appendix

9.1 Terminology

Agent An agent can generally be described as an entity which is autonomous, reactive to its environment, pro-active towards achieving some goal, and has some social ability for communicating with humans or other agents [1].

Meaningful human control Entails that a human has the ability to make informed choices in sufficient time to influence the system, as to achieve a desired effect or to prevent undesired immediate or future effects on the environment [14].

Directability The ability of a person to influence or control the behaviour of an agent [15].

Tasking The act of a human providing one or multiple agents with a task to perform, which the agent has to perform as intended by the human.

Task A task is an activity or piece of work which has to be done. For this thesis, a task is defined as a specification of a desired outcome, independent of the process followed to achieve them. As such, a tasked agent is left with freedom on *how* to complete the task.

Human Intention For agent tasking, the human intention defines the way in which the human wants the agent to complete its task, as to be in line with what the human finds important (e.g. ethics, law, safety, task execution speed, etc), avoiding unintended behaviour and consequences. The human intention for a task is defined in this thesis as restrictions on the agent's procedure or policy followed to achieve the task. As an approximation of human intention, constraints are used in this thesis.

Taskable agent An agent which has the ability to carry out different tasks, in response to some task command from a human or other agent. The greater the diversity and number of tasks it can perform based on the external commands, the greater its taskability [2]. In addition, there are various desired characteristics for a task-executing agent, such as being adaptive and task effective [19].

Plan Knowledge representation which encodes a method for achieving the task through a set of actions in a specific order or sequence [43].

(Task) Constraints In this thesis, constraints are defined as strict restrictions on the behaviour of the agent, such as prohibitions, obligations, or permissions, constraining the possible methods of executing and completing a task. Constraints are as such similar to policies [47], or in the constraint programming literature corresponding to hard constraints (in contrast to soft constraints or preferences) [52].

Context Any information that can be used to characterise the situation of an entity [41].

Scenario A task that is to be executed in a specific context.

MATRXS The Man-Agent Teaming Rapid Experimentation Simulator (MATRXS), developed as part of this thesis in collaboration with TNO for easy simulation of human-machine interaction experiments, used in this thesis for simulating agent tasking with various constraints, scenarios, and agents.

9.2 Context-dependency Complete Results

		Participant 1		Participant 2		Participant 3	
		XGBoost	KNN	XGBoost	KNN	XGBoost	KNN
Drone flying speed	F1	0.00	0.55	0.69	0.69	0.64	0.64
	AUROC	0.50	0.50	0.50	0.50	0.50	0.50
Notification frequency	F1	0.88	0.88	0.92	0.92	0.69	0.69
	AUROC	0.50	0.50	0.50	0.50	0.50	0.50
Fly village	F1	0.00	0.75	1.00	1.00	0.81	0.81
	AUROC	0.50	0.50	-	-	0.50	0.50
Fly water	F1	0.67	0.67	1.00	1.00	0.00	0.00
	AUROC	0.50	0.50	-	-	0.50	0.50
Distance to tank	F1	0.41	0.41	0.41	0.41	0.41	0.41
Distance to radar	F1	0.47	0.47	0.47	0.47	0.47	0.47
Time limit	F1	0.66	0.66	0.38	0.38	0.41	0.41
Averages	F1	0.44	0.62	0.69	0.69	0.49	0.49
	AUROC	0.50	0.50	0.50	0.50	0.50	0.50

Table 9: Scores for the F1 and AUROC metrics of the XGBoost and KNN models for predicting the constraints (row labels most left) as filled in during the drone reconnaissance task. For this condition, models had were trained to predict the constraints with no contextvariables. Thus leading to the model only being able to learn one option for the completetask. Results are shown for models trained on the data of each individual participant