POLITECNICO
MILANO 1863

POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY

LEARNING BEHAVIORS TO OPTIMIZE THE PLAYER'S EXPERIENCE IN ROBOGAMES

Doctoral Dissertation of:
Ewerton Lopes Silva de Oliveira

Supervisor:
Prof. Andrea Bonarini
Co-supervisors:
Prof. Tiago Pereira do Nascimento

Tutor:
Prof. Francesco Amigoni

Chair of the Doctoral Program:
Prof. Andrea Bonarini

2018 – Cycle XXX

Abstract

Write me

Sommario

Scrivimi

Contents

List of Figures

List of Tables

Chapter 1

Introduction

What seems to be a natural evolution for game playing experience is to bring the elimination of screens and devices in order to present the users with the possibility to physically interact with autonomous agents in their homes without the need to produce an entire virtual reality. This pretty new style of games has been recently defined as Physically Interactive Robogame (PIRG) and has as the main objective the exploitation of the real world (in both its dynamical unstructured and structured aspects) as environment and one or more real, physical, autonomous robots as a game opponents or companions [62].

Like commercial virtual games, the main aspect of PIRGs is to produce a sense of entertainment and pleasure that can be "consumed" by a large number of users[1]. Furthermore, an important aspect of autonomous robots and systems during the game should be, as commonly expected, an exhibition of rational behavior and, in this sense, they must be capable enough to play the role of opponents or teammates effectively, since by practical means people tend do not play with or against a dull entity [62].

Naturally, to help to come up with better agents its easy to think about extracting knowledge from co-existing agents in the sense of implementing some mechanism for modeling them, both to the extent of the quality of low-level actions they choose as well as interaction patterns with the system. This effort is justified in the context that being able to recognize other's intention may substantially improve one's capacity of taking better decision when acting upon the environment. At least in the human's perspective, this ability is critical since interpersonal interactions presuppose the understanding of motivations, high-level plans, and estimation of future events [101].

[1] In this work, since the player is an user for the gaming application both words "player(s)" and "user(s)" will be used interchangeably.

When it comes to extract useful information from agent's behavior, one can see at least two main different, yet related, approaches: *a*) Modeling for competitive advantage and *b*) Modeling for experience optimization. In the former, techniques for evaluating pay-offs from interaction patterns, such that provided from *game theory*, play an intense role not only to what concerns virtual interactions (with virtual agents) but real-world events involving humans as agents (e.g. trading, patrolling, competition) or physical robot entities. In order to get some competitive advantage against an adversary with private strategies and conflicting goals it is necessary to adapt to the dynamics of the environment caused by the game play [75]. In essence, this means that it is vital to pay attention to any information from the opponent behavior that might help to optimize the decision making process and find appropriated countermeasure actions.

The focus on modeling behavior for experience optimization, is much related to the idea of extracting useful features from a user in order to adjust parameters that are correlated with his experience in a platform or product, all for the sake of offering a better product or helping the user to achieve some particular goals. In a gaming scenario, in turn, this notion is commonly applied when designers attempt to define a mechanism capable of adjusting the difficulty or general appearance of the game all in the expectation of rising the player entertainment. Very traditionally the sense of game difficulty is designed to increase along the course of the experience, which can either happen in a linear fashion or through steps represented by the levels or phases, where a player is forced to select the difficulty level through a set of discrete option (easy, medium, hard, very hard). However, given the observation that very often this "static" way of setting up a difficulty curve is not accurate enough and it may not account for the difference between players or even the different rates of leaning of each of them, in principle, it turns out useful and natural to think about coming up with modeling techniques that may help to empower the player experience.

Such models, however, are greatly impacted by the type of scenario they are applied to. Normally, in a game scenario a computer-controlled agent may receive a noise-free sensory data which is obviously not possible to occur in a real-world scenario, specially in robotic applications such of that of PIRG. The general suspicion about a full spread of such solutions, mainly in virtual game development, is that they ultimately takes the control away from the design and put it basically in the code, which has obvious drawbacks, ranging from high-demand for computational resources and storage to general game behavior [40].

In summary, it would be important to have a PIRG autonomous robot that

could be able to automatically adjust its behavior such that it may be likely to match the user's skill and by doing so maintain the user engaged. Also, it is important to, by aiming those objectives, make the robots appear more intelligent.

In this thesis work, after providing a panorama of approaches that take inspiration from *artificial intelligence* (AI) and *machine learning* (ML) techniques in order to tackle the problem of modeling co-existing agent's behavior and activity (including humans) in games and robots, we propose models and report of results showing effort in addressing the design of better agents for PIRG.

The scope of this document is heavily centered on model proposals that can latently model player activities and general behavior. Next section details our objects and research organization.

1.1 Research questions, hypothesis and objectives

Since, in any kind of PIRG, autonomous robot are supposed to be perceived as smart entities, the key point for investigation deals with finding good answers for two main questions:

- How to discover players types and quantify player behavior in PIRG.
- To what extent does adaptation impact reported fun?

From this, the objective focus on the exploitation of Machine Learning (ML) techniques to the design of better PIRG robotic agents, which should lead to more engaging playmates, possibly capable of performing behavior/strategy adjustment. The hypothesis is that ML would help to decrease predictability in robot behavior and introduce game dynamics capable of considerably empower the user's engagement, making so that the agents can be well accepted as game companions. Specifically, the general goals were:

- To perceive and report the impact caused by the use of ML-based techniques in the design of PIRG from the view of human-robot interaction.
- To study ways of implementing user's behavior modeling in PIRG robots by reasoning about data coming from player tracking.
- To explore ways of strategy adjustment using information about past interactions and experiences (player's typical behaviors, preferences, etc.).

It is supposed that autonomous and learning systems that encompass perception, action and communication in a unified and principled way via ML-based techniques lay at the core of a new frontier for robotics, and PIRGs in particular. The objectives also aim to keep control on technical constraints by exploring the use of cheap sensors, and algorithms requiring little power ("green algorithms") to be executed in real time and non-structured environments, as a way to enable the spread of PIRG in the society, making them reach the market in large scale.

1.2 Thesis outline

The thesis is divided into the following chapters.

- *Chapter 2:* Many approaches for tailoring game experience to players had been proposed over the years. The chapter presents an overview of such literature, placing PIRG as a relatively new area of research.
- *Chapter 3:* The game environment and robot platform are at the core of a PIRG application. In this chapter we detail our selected environment and adopted robotic platform. Other mathematical background are provided.
- *Chapter 4:* One step towards allowing effective player modeling is the implementation of an activity recognition system. In this chapter we describe the efforts on exploiting a simple input data transformation for the recognition of motion primitives in acceleration pattern akin to archetypal activities in the game scenario.
- *Chapter 5:* Player behavior modeling is the backbone of adaptive behavior strategy for playing robots. The chapter presents new idea for latent player behavior modeling.
- *Chapter 6:* Adaption is by no means a trivial task specially in the context of PIRG. In this chapter we detail a system to actively selected appropriate difficulty settings for the human player.
- *Chapter 7:* Engagement is believed to be related to several factors and one of such is the level of information about the opponent actions. In this chapter we present a study case of the use of deceptive motion during play. Results indicate that on our particular game scenario the human capacity to correctly identify the robot deceptive intentions is blurred possibly by the intensive cognitive load demanded by the activity.

- *Chapter 8:* The design of PIRG is an intensive process. In this chapter we provide some key-issues regarding how one may develop successful engaging experiences.
- *Chapter 9:* Concludes the thesis and details further direction for our research.

1.3 Paper contributions

- Ewerton L. S. Oliveira et al. "Learning and Mining Player Motion Profiles in Physically Interactive Robogames". In: *Future Internet* 10.3 (2018). ISSN: 1999-5903. DOI: 10.3390/fi10030022. URL: http://www.mdpi.com/1999-5903/10/3/22
- Ewerton Oliveira et al. "Modeling Player Activity in a Physical Interactive Robot Game Scenario". In: *Proceedings of the 5th International Conference on Human Agent Interaction*. HAI '17. New York, NY, USA: ACM, 2017, pp. 411–414. ISBN: 978-1-4503-5113-3. DOI: 10.1145/3125739.3132608. URL: http://doi.acm.org/10.1145/3125739.3132608 (visited on 11/10/2017)
- Ewerton Oliveira et al. "Activity Recognition in a Physical Interactive RoboGame". In: *2017 Joint {IEEE} International Conference on Development and Learning and Epigenetic Robotics, ICDL-EpiRob 2017, Lisbon, Portugal, September 18-21, 2017*. Lisbon, 2017, pp. 1–6

Chapter 2

State Of The Art

2.1 Physically Interactive RoboGames

Recently, video games companies inserted into a mass market of entertainment, have aimed on establishing a new paradigm that involve the players actively moving in front of the screen, picking object around them, actually interacting with the game in a three-dimensional space. This often happens by constructing an immersive virtual reality game where players are plunged in an artificial world reproduced by *ad-hoc* intelligent systems devices [131]. This scenario, although enabling impressive game experience often requires to wear special devices that may limit the quality of the movement possibilities [62].

Enabled by the increase maturity of fields like social robotics, artificial intelligence and machine learning, [62] pushes forward the idea of a designing a new level of game experience: that of Physically Interactive RoboGames (PIRG). They provide a definition to this kind of ambient by which I adopt here also as the meaning for the term *robogame*:

"*A Physically Interactive RoboGame consists of a number of autonomous agents (including software, hardware, and physical agents) of which at least one is an autonomous robot, and one is a person. These agents interact with each other in a possibly variable and unknown environment, by following some game rules, so that the human players can have fun* [62]."

The authors also emphasize the existence of similar attempts for presenting robots as games where, in most cases, acted more or less as mobile pets (e.g., [32, 90]). In that case, interaction is often seen as limited to almost static positions, not exploiting rich movement, nor high autonomy; the credibility of these toys to really engage lively people, such as kids, cannot be high. A notable exception, according to them, is Kiro [121], a robot able

to successfully play table soccer even with experienced users. In the game, the movement is limited to the control of the table bars, and the game is the robotic version of a classical bar game. Note the reader that the interaction provided by Kiro is relatively simple, in a very structured situation, but despite of that it matches exactly the user's expectations [62].

Robogames can be one of the next robotic products for the mass technological market, thus demanding a large exploration of new methodologies and application, specially to what concerns methods for enabling high autonomy, intelligence and adaptive behavior. In this direction, a company called Anki started to make use of artificial intelligence techniques for the task of bringing consumer robotics into physically gaming experience. During the keynote event at Apple's Worldwide Developers Conference in 2013, the company revealed its first product: Anki Drive, a racing game featuring toy-size robotic cars controlled by a mobile app that serves as an AI engine. In 2015, the product was considered one of the best toys available in the market, which somehow express the interest of the market for this kind of real-world smart robotic game experience.

In the game set, the cars are sensing their positions on a carpet-like track and continually exchanging data via Bluetooth with the mobile phone. The app computes possible actions and decides what each car should do based on its objective. Thus, because is the app that defines how each car behaves and creates different gameplay scenarios, the cars can have different "personalities" and get customized features, like different "weapon systems". Despite of the app ability for orchestrating the behaviors of the cars and posit a sense of autonomous driving, players are still limit to the use of their mobile to actually drive the cars, racing against each other or taking on the AI (facing other cars).

Also over the few years, a set of PIRGs have been developed and tested at the Artificial Intelligence and Robotics Laboratory (Politecnico di Milano, Italy) although focusing on a more close physical interaction with players, as defined in [62]. As an example, the Jedi Trainer 3.0 was developed with the goal of resembling a scene of the first Star Wars saga movie "Episode IV - A New Hope". The game was designed to use an autonomous drone, a Parrot quadricopter, flying around the human player (Jedi trainee) and making sound recalling the shot of a laser blast. The drone always aims at shooting at the "Jedi" (human player) chest that in turn physically interact with it using a light-saber-looking object trying to parry the quadricopter attack as best as he can. The Jedi Trainer has been successfully tested in different unstructured environments, as reported in [15, 62].

Another example is the RoboTower. It is inspired by the videogame Rock

of Ages. In the game, a 30 cm high robot has to bring down a set of towers in three minutes. Human players can interact with the robot by delaying its movement through the use of cards. Those card are selected from a deck they have in hands and put in front of the robot that can read them when it passes over. Each card represents either an action that the robot has to execute (go back, turn around) or a deficit for its sensors (go blind), or a stop for an amount of time. A red tower correspond to the player's home and when ruined he loses; other towers are production plants that are used to recharge the delaying cards, and make them again playable after a given time proportional to the number of active plants (cf. Fig **??**).

As it can be easily deducted, the new application field of PIRG and the complexity it requires prompts a new endeavor of Human-Robot Interaction (HRI) which is defined as the study of how humans can interact with robots, "and how best to design and implement robot systems capable of accomplishing interactive tasks in human environments" [31]. Undoubtedly, robogames appears as a tool to investigate HRI issues as its design process has not to do only with the technical issues of a robotic application, but it has also to consider other important aspects, including playability and usability of the game [62].

Up to this point in time, it becomes more and more evident the necessity of investing on the research and implementation of intelligent algorithms as a way to improve PIRG design and help to popularize the idea of having such products on the market. The real necessity is that of investigating how to develop complex cognitive abilities in autonomous robots by the use of machine learning (ML) techniques. One example of such ability would be that of intention detection for strategy adjustment.

Also, it is important to give preference to the exploration of mobile robot bases with cheap sensors and algorithms requiring little power to be executed in real time ("green algorithms") in non-structured environments since these are interesting constraints currently addressed in robogames, and in the whole Robotics community. Of course, reducing price can hugely enable the spread of robots in the society and make PIRG easily reach the market.

In the next sections, it will be identified several important aspects from methodologies aiming to construct user models all in the purpose of getting useful insights to help in the design of better PIRGs. The intuition I explore is that of user modeling techniques should strongly help in the discussed problem. Restating it again, exploiting complex cognitive abilities in robogames may increase the player's engagement since any apparently rational behavior of the robot may help to accept it as a robotic companion (or opponent) "while random actions or too complex strategies for the cogni-

tive or perception levels of the players are perceived as nor purposeful" [62]. Due to the inherent complexity involved in the scenario, it is not possible to only rely on "hard-code" abilities. The problem calls for the massive use of ML-based techniques to make sense of the obtained interaction information.

2.2 Modeling for competitive advantage

There is a set of different lines of research that examine several different methods and goals by which an autonomous systems, virtual or robotic, can learn to recognize activities. Here, I focus on systems that exhibit some concerns about taking advantage from knowledge extracted from physical interaction in robotic competitive environment and virtual scenarios. Since the adversarial nature, those methods are often referred to as approaches that use *opponent modeling*. Specifically, this section summarizes prominent aspects of popular opponent modeling papers envisioning to point out related possibilities for PIRG design. I begin by stating the basic work-flow that is often done when constructing such approaches.

Commonly, three basic steps are followed: *feature extraction, model construction and design of changes* (cf. fig. 2.1). Feature extraction relates to the idea of choosing which set of raw sensory inputs gives most information for capturing a target behavior. After deciding upon those, a model is constructed in favor of any successive user classification attempt. Intuitively, the goal here is to design a way to get models that encode different types of behavior/strategy co-existing agents (opponent) might have, thus enabling recognition. The last step is the decision of how the system should react to different users. This is undoubtedly the most difficult phase since, among other things, adjustments should also take into consideration the effect of modeling errors. Perhaps, the concentrated reader could realize this work pipeline correlates to approaches normally used in data mining or statistical/machine learning methodologies.

It is not a secret that in order to be able to increase the likelihood of winning a game one should also consider the extraction of knowledge from opponents in the environment. Opponent modeling *per se* is seen as the attempt to predict and identify behaviors of co-existing adversarial agents and propose appropriate countermeasure actions toward maximizing a given utility [30].

In the Robot World Cup Initiative (RoboCup) a body of research has been done in the last decade trying to address the problem of opponent modeling. The RoboCup is a multidisciplinary initiative for building a team of robots capable of playing soccer. In this scenario, opponent modeling concepts are

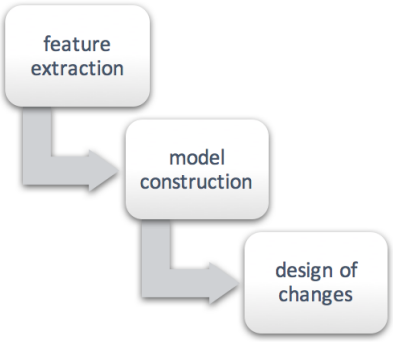


Figure 2.1: Basic work-flow for designing systems able to implement any kind of modeling of co-existing agents for competitive advantage.

seen as an important requirement for building a competitive team and also as a growing research topic not only on this domain but also in the general multi-agent system area [75].

Since in most cases game history data is available, in either categories, researchers have extensively used data mining and machine learning techniques in order to construct off-line model of opponents which could them be used as classes in a kind of "prediction phase" during actual game play. Often, this prediction phase basically means: try to match the current observed opponent behavior to one the existing models for then try to adjust the internal behavior parameters accordingly. This matches the basic work-flow scheme in figure 2.1.

Some important contribution to the problem has been done by [82–85]. In [84], the authors explored the domain of RoboCup by studying how to improve their agent team by constructing a probabilistic model representing predicted opponents' locations given the recent history of ball's movement and initial team members' locations. As common in this scenario, their approach focused in the use of a "coach" agent, enabled with a number of predefined models and a centralized view of the world, whose main role is that of communicating a plan to the rest of the team. The big idea is to taking into consideration the models' match with the observed data in order to predict opponent general behavior.

The main assumption made by the author was that of opponent behavior being generated from a sample drawn from the predefined models given to the coach. Naturally, given the predefined models, it is still a challenge to decide which model best describes the opponent at run-time. As in the common supervised learning paradigm, they assumed the opponents will behave similarly to how they performed in the past, and use that information to de-

velop a plan. In all of the models used, the distribution of each players final position was represented by a 2-dimensional Gaussian with equal variance in all directions. On successive works [82, 85] the authors managed to improve the online model selection through a kind of Naive Bayes approach focusing on plan representation and execution expressing spatial-temporal relations.

An interesting observation made by the authors, also pointed out in [75], is that during the plan execution it was not possible to take advantage from single unpredictable opportunities that may emerge. For instance, the agents would follow the plan strictly even if there is a clear chance of being successful against the opponent team by taking an immediate off-plan action. It is important to observe how this issue may direct affect the efficiency of autonomous agent involved in a PIRG. I understand that it is extremely necessary for the agent to keep a closer loop with the environment, paying attention to every observation that may emerge as opposed to always blindly follow a prescribed solution. Naturally, this pave the road for solutions that are able to represent some degree of belief regarding plan execution and revision. A possible solution to this issue was also pointed out by the authors. They basically suggested to store alternative plans and intelligently add monitors for these plans as in [110] so that they could make the plan execution opportunistic [75, 82, 85].

In [41] the goal was to use of statistical dependency tests for the identification of significant sequences from which to relate states to chains of events. Their underlying assumption was that observed team behavior can be transformed into a sequence of ordered atomic behaviors. In the follow-up work [19], sub-sequences inside sequences of behaviors are analyzed by using a frequency-based method. The big aspect presented on their work is that the model of an agent behavior is represented by a distribution of relevant sub-sequences. The behavior classification procedure is done using a modified Chi-square Test. Further improvements descibed in [42].

[97, 98] studied the application of feature-based models for representing opponent behavior. In [97], the gist of his approach comprises the identification of few features that could be observed by raw sensor data during game play, which diverged from [83] whose method stored every observation on the models. Those features were classified by using a Bayesian method, then a knowledge base is used in order to respond to what has been classified, i.e, select the appropriate strategy. One example of such features were the use of text-based description for in-game situations, like: “The opponent often does long passes along the left wing to the forwards”. The methodology investigated in the paper turned out to be further studied to the extent of the use of Case-based Reasoning in [98]. In this follow-up research, Steffens was

able to identify some gain in classification accuracy showing that similarity-based opponent modeling can benefit from domain knowledge [75]. Case-base Reasoning, however, is likely to lead to high computational costs given the large number of cases do be retained in a highly dynamic environment [1, 75].

In [44] their technique translates observations into a time-series of recognized atomic behaviors. For instance, given a stream of raw observation about the team members' position and orientation plus the position of the ball, their approach would recognize soccer-playing behaviors (passes, dribbles, etc..). Unlike other efforts, they did not care about how the low-level behaviors combine together for generating high-level strategies.

[56] also formulates the problem of opponent modeling as a classification task. In tacking the problem they proposed the decomposition of the learning task into procedures: the learning of the action name (passing the ball, kicking), and learning the parameters of that action. A follow-up work [54] studied the subject further by implementing machine learning to create modules that is able to infer the opponent's actions by means of observation. Their method, was called "Opponent Modeling Based on Observation" (OMBO) and was designed for two make improvements in two pathways: First, tacking the creation of a generic module (Action Labeling Module) that is able to label the last action (and its parameters) performed by any robosoccer opponent. This happens via the observation of another agent .The justification for this method is that agents generally do not have access to input/output pairs of events generated by a given agent. So, the approach must rely on sensor inputs only. Second, there is the construction of a model of the other agent based on the acquired data from the first module. This is what they called "Model Builder Module". In [55] they further discussed the applicability of the OMBO method giving a proof-of-concept of when learning a model of two different goalies. The idea was that their team striker gets as close to the goal as possible, and shoots when the goalie is predicted to move, i.e, by anticipation of the opponent movement. Figure **??** presents the holistic view of the method found on [54, 55]. The reader is invited to see how it relates do figure 2.1.

Using data mining in order to define models of opponents in an off-line manner is the ideal scenario of an agent learning a model of other agents' behavior via direct observation of their past actions. However, that is only somewhat reliable when agents have many repeated interactions with one another or when the assumption of similar behavior between agents holds, otherwise it is somewhat hard to obtain any good generalization [99]. Given the characteristics of the domain (physical interaction between agents), one may

realize that the notion of proximity is quite important[1]. Often, authors try to somehow exploit proximity aspects as well as the spatial-temporal relations with certain property of the environment, for instance, the domain an agent exert over a field region (e.g, [81]). Obviously, proximity is not enough, but when combined with other aspects, like timing aspects, it is possible to get good estimates about the behavior of agents in their attitude with respect to in-game tasks. This trend is going to be present on the majority of papers focusing behavioral modeling.

When tackling team formation, one may observe that a large number of machine learning techniques has been tested in order to improve countermeasure actions for a specific team. Again, most of the methodologies used log files for the purpose of off-line learning of important characteristics. Neural Networks, for instance, has been extensively used for identifying the position of opponent team members with respect to given preset formations. After identifying a given formation, a plan could be transmitted by the coach agent to the rest of the team all for the purpose of performing the appropriated counteraction [29, 66, 77, 111].

For a in-depth overview of opponent modeling in RoboCup the reader is invited to refer to [75]. In the work the authors classify methodologies into two categories which basically comprise *a*) "formation modeling", as it stands for the task of modeling team plays under the umbrella of collective behavior, and *b*) "individual behavior modeling", which is related to the idea of modeling a single opponent agent.

2.2.1 Commercial interactive environment

Virtual games have enabled a large amount of research effort into the design of player modeling methodologies. This is due to the increasing need for making Non-player Characters (NPCs) believable, i.e, make them resemble human behavior by implementing similar (to human) deductive reasoning process for action selection. In summary, the need of such ability in virtual games is at least due to a few main facts: *a*) the growth as a commercial product; *b*) the increasing complexity; as well as *c*) the need for developing games that can exhibit a major level of intelligence and adaptive behavior (personalization) [7].

An increase collection of papers has been published trying to address the problem of predicting the player (opponent) behavior (actions) in different levels and in different contexts. Focusing on opponent modeling, [38] presented an overview of efforts for commercial games. In this work, they

[1] proximity is a notion that basically relates to how close/distant interacting entities. It this turns out to be an ubiquitous notion even in simulated physical interaction as that showed in games.

emphasize types and roles of opponent models, such as "speculation", "tutoring and training" as well as "mimicking characters". In "speculation", the idea is that some kind of heuristic (or utility function) – such as minimax in zero-sum games like chess or checkers – is used in order to assess the quality of available opponent's actions during game-play. In simple terms, this relates to the idea of using knowledge of the opponent's preferences or skills in order to drive the game into positions/states that are considered to be *less favorable* to the opponent. This is the same core idea behind the approach described in [61], where they defined the concept of *opponent weakness* (as a quantifying measure) together with a method for learning a model of this concept. The key-point in [61] was the care about the potential harm of modeling error and the incorporation of their concept as an extra feature in the decision process, not as the core itself.

Another example of such modeling type is the research done by [64] where the authors proposed to rank available actions for the player (at each turn of the connect four game) as to find a way to balance the behavior of the computer-controlled agent to that of the human player. The aim was to estimate the player's expertise, when looking at the history of moves performed, and by assessing their quality (rank) based on a minimax approach. Thus, at each turn, they could make their agent select only actions that have similar rank to that of the player[2].

Tutoring and Training is seen in [38] as a type of opponent modeling that has to do with the assistance of a human player. For instance, a model of the human opponent may be construct as to teach him how to achieve certain in-game goals in a personalized manner. This is ideal for *serious game* environments where the player is confronted with a simulation of real-world events while trying to solving a potential real-world problem. Even by knowing that serious games can be entertaining, in this scenario the main purpose is essentially to train or educate users, thus, coming up with models that helps the game steer behavior towards those aspects are definitely important. In this context, [36] tried to identify player's goals in the game Crystal Island – a non-linear educational game about microbiology – by using Markov Logic Networks[3]. Goal recognition is assumed to be an important piece for player modeling and generally tries to identify the user's goals from a set of low-level observation (abduction)[4].

[2]Despite the fact [64] is mentioned here as an "speculation" type of opponent modeling – since there's a kind of simulation procedure for evaluating the player's quality of movement – one may also see the work as an effort for tailoring the game for optimizing gaming experience, i.e., a kind of difficult adjustment approach (cf. section 2.4).

[3]a probabilistic technique comprising a set of weighted first-order logic *formulae* that enables uncertain inference over "traditional" binary logic.

[4]Also related to goal recognition are techniques like plan and activity recognition, both well-known problems

The study in [36] follows previous work on goal recognition (such as [65]), but under a much broader view: that of having individual goals not independent from one another (which turns goal recognition into a classification problem) and that of having ambiguous causality effect between actions and goals. In terms of features for player actions, they targeted three properties: *action type* (e.g, moving to a place), *location* (game place where the action was taken), *narrative state* (player's progress in the game narrative). Once again, model parameters are learned from corpus of data collected from the environment. Their results were compared with two baselines based on *unigram* (a model that predicts goal based on the current player action) and *bigram* models (makes prediction based on previous action as well) obtaining a 82% improvement over them.

In [38], the "mimicking characters" type is said to correspond to the observation that the virtual game is designed to be, above all, fun and entertaining, as opposed to play as strong as possible all the time. This amounts to the well-known observation that when performing a companion role, for example, the agent must do what is possible to behave in accordance to the expectation of the player, otherwise the human may lose interest in the game. Thus, this requires a model of the human in order to be effective. Also, in a PIRG scenario, this is an important aspect that would enable advancements in the interaction as a whole since the behavior of an autonomous agent should be dynamic enough to adjust to the individual experience. The central point is that of not designing static behaviors that are likely to be exhaustively exploited by the player. This type hugely overlap with the ideas of player modeling for creating a balanced game play, i.e., adjusting the game for the player individual experience. This subject is brought back on section 2.4.

Often, games and simulations provide access to full-observability of in-game events and player's actions. When this is somehow not true, there may be a (often large) corpus of data from usage history that is available for information extraction. This, naturally favors the use of off-the-shelf data-mining algorithms that are able to be put together easily and tested multiple times. However, on the design of PIRG, specially on a brand-new project, one may suffer from the lack of data for constructing off-line models of players. To get around this problem there exist at least two main alternatives: *a*) setup a data collection procedure, for example, by designing a first version of the game, implementing it and then collecting data in order to refine the game design; or *b*) invest on online procedures that are able to extract features from the player and come up with models of player behavior on run-time,

in general AI [36].

this later, of course, is undoubtedly much harder to face.

In PIRG, it may be possible to design methods that account for the evaluation of the current state in order to know which are the corresponding chances of winning/losing at the moment allowing for planning ahead ways to overcome/support in case of need. However, it is worth to salient the general constraints perceived in the domain, such as those from computing complexity, mobility, perception. As part of the domain definition it has been suggested the necessity for designing PIRGs via the use of low cost platforms which commonly have limitations regarding computing power, energy consumption and others. Indeed those constraints must be taken into account if there is the clear goal of making PIRGS reach the market [62].

Competitive advantage and/or experience optimization (section 2.3) are not the unique types of approaches. There were also attempts for classifying opponent behavior to the extent of helping in the design of the game itself. An example of this is the work of [28] where it is possible to see the implementation of fuzzy cluster analysis and Hidden Markov Models (HMM) for finding player styles. The approach works as player behavior[5] classification method that consists of three components: Interaction Registry, cluster analysis and HMM. The first, serves as a data storage for actions maintaining a cumulative score value for them. The idea behind this is that actions that are not frequent are going to have an exponentially decreasing value in importance. On the other hand, frequent used actions have high importance score. Those importance values for actions then constitute a primary source of information that is going to be used by the cluster analysis as a way to group, i.e. classify, player's style. HMM is subsequent used during game-play aiming to classify new players, as they play, and helping to spot appropriate and dependent game adjustments. The author claimed their method would be able to be used across different games (given their definition of player behavior) which can help as a possible generic design-tool.

Using the Rush 2008 American football simulator [52] introduced methods for performing predictions about the players' physical movements when learning team policies. When focusing on recognition of team play they investigated the use of support vector machines, an optimal margin classifier commonly used in supervised learning problems. They trained SVMs for a multi-class objective by using a collection of simulated games under controlled conditions, so they got instances of every possible combination of offense and defense plays from a number of team starting formation configurations. The output of the play recognizer was defined as the system's best guess (at the current time step) about the opponent's choice of defensive

[5] defined as a sequence of game actions.

play. Thus, they could use this information to select the most appropriate offense.

Also in [52], the authors focused on proposing methods for discovering how to effectively subgroup agents together so to accomplish a given formation task, similar to [100]. Here again they based their method on an analysis of game data from successful team plays. The idea was to implement a supervised learning mechanism in order to identify important group of players w.r.t. each play. It is interesting to mention the three general types of cues that were used for the purpose of subgroup extraction: *spatial* – i.e, the constant relationships between team members over a period of time; *temporal* – co-occurrence of related actions; *coordination* – dependencies between members' actions. Those cues turn out to be basics features from most types of approaches in the discussed scenario. Undoubtedly, a larger number of papers has been focusing on player modeling for improving playing experience as to maximize the chances of maintaining the player engaged. This is the topic of discussion of the next section.

2.3 Modeling for experience optimization

In virtual games player modeling is best defined as the study of the use of artificial and computational intelligence techniques for the construction of models of players in dimensions regarding behavior, cognition and affective state as well as beyond-game high level aspects spanning personality and cultural background [129]. The idea behind such models is related to the goal of enabling a game to adjust the capability of its components and attributes towards an individual player satisfaction [38]. This idea has largely increased in importance over the last years. As one of the central reasons for that is the increasing complexity of modern games fostered by the enhancement in computing power and graphics, as well as the commercial strategy of personalizing gaming experience [38, 103]. This section discusses the dimensions mentioned in the definition, namely: *cognition*, *behavior* and *affection*, describing their main characteristics and supporting work.

2.3.1 Focus on cognition and theoretical models of behavior

One way researchers have addressed player modeling in games is by relying on theoretical frameworks mainly supported by psychology and neuroscience factors [129]. By doing so, cognition aspects – broadly seen as a set of all mental abilities and processes related to knowledge, such as: attention, memory, evaluation, reasoning, problem solving and decision making, learning and so forth [**wiki:cognition**] – are often investigated.

Researches focused on theoretical framework of behavior are described in [129] as being model-based. In this sense, they follow the *modus operandi* of humanities and social sciences which work, according to the authors, by coming up with a hypothetical model in advance that is investigated in relation to what extend it fits the observations. When concerning the development of engaging experiences in entertainment systems, [122] states that a comprehensive review of the literature regarding the elicitation of qualitative features and criteria derived from experimental psychological studies reveal a tendency of overlapping with the exploitation of theories of *fun*, such as those discussed in [50, 53, 60] envisioning to drive game design, as well as *flow* [24] – which is traditionally used as a way of evaluating player enjoyment [23, 102] (Fig. 2.2) – and concepts like *immersion* [21]. Most acceptably, according to theses theories and concepts, player engagement is most related to factors such as challenge, curiosity and fantasy.

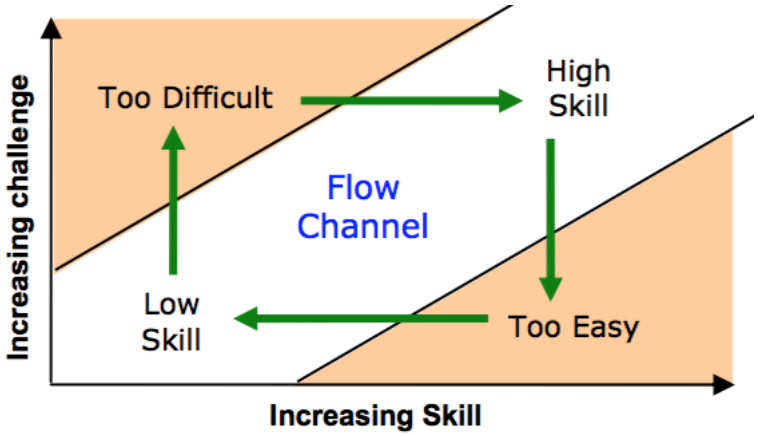


Figure 2.2: A flow diagram from [40]. The figure shows the relationship between skill and level of challenge derived from the theory of flow [24]. The point is to keep those two dimensions in a kind of positive correlation, meaning an increase that roughly approximates the diagonal.

Models aligned with cognition theories are seen as a valuable opportunity for understanding what the user is thinking when playing, which has direct relation with the ultimate desire of making the game respond intelligently to the player. Supported by [14], the fundamental advantage of a direct focus on modeling principles embodied in a theory of cognition is that it provides dynamic view at the individual player level, since it is possible to make statements about attention, learning, decision strategies, biases, an so on, unraveling indicators of the mental underpinning of observable behavior.

The design of better NPC's, for instance, is one of the central areas positively affected by the focus on cognition [33]. Intuitively, once it is possible to measure some cognitive-related characteristics in the player (e.g., the attention level with respect to some in game resource) it is possible to use them to steer the NPC behavior towards the improvement of a desired AI capability – Adaptive difficulty adjustment is one of those capabilities. Indeed game difficulty is seen as having a direct link between challenge and fun, acting as source of satisfaction [50, 128] (cf. section 2.4).

Cognitive models of players were also investigated with respect to psychological and cognitive neuroscience motivated question trough assessing physiological and/or psycho-physiological states during play. For instance, brain imaging techniques have been used to understand brain activity patterns related to aggressive thought stimulated via violent game content in [120]. The work of [11] used Electroencephalography (EEG) combined with psychometric measures, as a first attempt to investigate neurophysiological underpinnings of spatial presence[6] triggered in different virtual roller coaster scenarios.

Optimistically, as seen in [14], the major point in cognition or related framework-based approaches for player modeling is to discover which model inputs[7] place, for instance, unrealistic demands on a specific cognitive function, such as attention. Successful results on this, might heavily equip researches and game developers with valuable guidance for narrowing the range of necessary resource-consumption when aiming for desired results.

2.3.2 Focus on in-game actions

A classical trend in player modeling is that of understand how primitive in-game activities are responsible for comprising the player's general behavior. Originally, the first attempts for modeling players in this context used classical zero-sum board games (like chess, checker and go) as test-beds mainly by implementing search/heuristic methods for find best moves in the game-tree. According to [8], since tree-search techniques use evaluation functions in order to assess the pay-off of a particular move, they may be consider player models.

In-game actions are building blocks for any standard behavioral modeling approach. Most commonly, primitive (or low-level) actions are the unique way available through which it is possible to identify player's preferences and decision-making style as well. Additionally, this modeling dimension

[6] spatial presence is considered as a sense of being physically situated within a spatial environment portrayed by a medium, such as television and virtual reality.

[7] in the sense of a parametric representation of cognition aspects.

is transferable across different game genres. A sequence of low-level actions – such as moving to one direction to another – is generally related with a classifiable behavior, such as "attacking", "defending", "fleeing", etc. In the RoboCup soccer league, for instance, a sequence of actions comprising targeted movements towards the opponent goalie may be commonly (and generally) classified as an "attack". Similarly, a sequence of actions intending to systematically prevent the opponent team from advancing on the field may be characterize as a "defensive" behavior. From this example, it is quite natural to realize that actions encode behaviors by perhaps acting as a kind of representation language for them. The important bit is that by keeping track of actions performed it is possible to find patterns that can be reasonably grouped together under the concept of a specific behavior[8].

Although classifying behaviors in this way may be relatively easy for simple full-observable environments, involving a few possible deterministic actions and abstract behaviors, this classification process is limited in modern complex games since they normally have a large state-action space. Also, behaviors commonly lack the existence of a clear crossing point between them what makes it challenging for the construction of player models. Giving the size of the state-space, researchers have also to deal with the uncertainty involved in the fact that different action sequence may converge to a same behavior while having a close relation with others. The notion that action sequences may be noise, is important at the same proportion that it is also a challenge that has to be taken into account when using low-level actions for modeling behavior. Furthermore, while computational constraints (such as memory demand) are the big key points for some case, for instance, in search methods applied to turn based games, the dynamic characteristics in other types of games is the key instead. In PIRGs, the inherently dynamism of the environment demands a special treatment on those aspects.

Besides all computational constraints involved and discussed on the previous paragraph, one might also see fit the necessity of having different levels for behaviors. For instance, one might be interested on identifying when the player is attacking mindless or when the player is using a full-fledged plan for performing the attack. In this context, what is the boundary characteristic that separates those two behaviors? Again, using the RoboCup soccer league as an example, *is the player engaged in a proper "attack" or just keeping the ball in the attack field perhaps for the purpose of spending time?*[9].

The work of [8] presents an extensive discussion about the use of actions

[8] as discussed, actions were the raw information in most modeling approach for competitive advantage in section 2.2.

[9] Note that a "proper" attack may have more structured actions, perhaps comprising well-know patterns as opposed to a behavior of just "spending time", which in this case may use different sequence of actions.

for player modeling envisioning game experience optimization. In the author's understanding, actions models seem to be an attractive possibility of a model since once being able to predict accurately the future actions to be taken by the player, acting accordingly would be a relatively easy task. However, they agree that the use of such models are limited unless when applied to relatively uncomplicated games. The justification for that is also related to the complexity of state-action space in state-of-art games. The authors, also deem three more subdivision of for the topic, described on Table 2.1.

Table 2.1: The proposed classification by [8] for dimensions used in player modeling techniques based on virtual in-game actions.

Type of model	Brief description
Tactic-based	relates to the automatic identification of short-term behavior as composed from actions targeting the achievement of a specific local goal. A common well-exploited concept is *formation of game characters* which can be defined as a disposition of certain game agents.
Strategic-based	concerns global-term game behavior assessed via tactics sequences. This behavior may span over the entire game, some iterations of it or even across distinct genres of games.
Profiling-based	concerns psychological motivation for tactics and strategies.

The divisions presented in Table 2.1 are, in fact, not mutually exclusive and one can see them in a loose hierarchical manner. In other words, tactical-based models depend on action-based ones and, in the same way, strategy-based models may comprise a behavior inferred with the help of the previous two, etc. The main advantage, of modeling tactics over actions, however, is that the state-space complexity decreases as a result of higher information abstraction. But, since they are interrelated with each other aiming to achieve an overarching goal, modeling tactics alone, according to the authors, is not enough for effective modeling player behavior since strategic play assumptions, as a high-level motivation for tactics, is commonly not incorporated, therefore, those models cannot generalize well over the underlying intentions behind observed tactics.

Additional to the use of actions in the modeling context discussed in section 2.2, they are also the raw material exploited on designing games that can adjust difficulty automatically in order to keep the player's engagement at a high. Specifically, along-side procedural content generation – the concept of generating game content on the fly – automatic difficult adjustment (DDA) has taken a large piece of the virtual game community.

In essence, the idea is pretty simple: start by measuring the player's skill or level of challenge (difficulty) at a given moment and based on that steer

the game behavior towards a state much likely to be compatible (in skills or challenge level) with that of the player. Following this recipe, Andrade et al. [3, 4] aimed at investigating the usage of Q-learning and a challenge function in a 2D fighting game scenario. In their work the authors came up with a challenge function strictly based on the difference between the NPC and player's health. From this, a game is assumed to be balanced if the function fluctuates around zero. By using a regular Q-learning they get to choose different best actions so to force the algorithm to choose the ones that most likely, according to the challenge function, matches the perceived player skill.

[40], in turn, addressed DDA by considering the notion of inventory analysis, that is, the processes of analyzing what resource are immediate available to the player w.r.t. the current challenge level of the game. During the game process, they observe certain player's characteristics, for example the level of damage the player takes, in order to generate an indicative for the necessity of system intervention. When needed, their system would adjust supply and demand or resources so to control overall game difficulty. Ideally, the authors target the reduction of necessary intervention so to make those as seamless as possible. The system aims at keeping the player at the flow channel [24] by encouraging certain events to take place or not. For example, the system basically tries to predict when the player is repeatedly putting himself on a state whee current means can no longer global or local goals. When this happens, the system intervene helping the player progress.

Not surprising, the notion of having a "challenger function" that helps to guide adaptation is pretty much ubiquitous in the adapting game literature. Furthermore, any system relying on DDA should also be concerned about the timing implication of such technique, i.e., the implications regarding the good moment to perform an attempt towards adjusting the game. For beginners, this is because a high frequency of adaptation is more likely to give the appearance of instability and so cause the game to be perceived as random. This instability is often referred to as the "rubber-band" effect as the IA appears to wobble around its skill level if the (human) player is too good and, conversely, overrun the player when he plays equivalent to the AI. Here though, appears clear the difficulty involving designing challenger functions. In the experiments done by [40], the authors noticed that an reactive approach, i.e. adjustment of onset elements, may run the risk of disrupting the player's sense of disbelief contributing to make the interpretation of the game harder and also make it "schizophrenic".

In Hagelbäck et al. [37], the authors went further on pooling people's opinion for the sake of knowing if playing an even game is more entertain-

ing over being superior all the time. On their case study 60 people participated and they took the opportunity to conducted experiments using static and dynamic agents for their game test-bed. The dynamic agent type was able to react to in-game player behavior (for instance, to the player losing game units). The results gotten from their research suggested the dynamic agents were most entertainment to play, at the same time the static ones presented themselves as too easy or too difficult. From here, I see once more the supporting evidence for justifying any effort in constructing efficient DDA-based systems for the purpose of keeping interactions. Despite the fact there are intuitive justifications, experimental results cover a fraction of the driving force behind such effort.

[96] is one of the most popular example of player modeling in virtual games. In their method, called "dynamic scripting" and defined as an unsupervised online learning technique for games, they maintain several rule-bases, one for each class of computer-controlled agents. Rules in the bases are manually designed using domain-specific knowledge and every time a new agent of a particular class is generated, the rules that comprise the agent script are extracted from the corresponding rule-base. In this approach, the probability that a rule is selected for a script is proportional to the weight value that is associated with the rule. Also, the rulebase adapts by changing the weight values to reflect the success or failure rate of the associated rules in scripts.

It did not take long until researchers realize using a challenge function would be able to pave the way for black-box optimization algorithm. Much of the time the literature covers the attempt on using evolutionary techniques as the main technique for DDA. Among those work, it is worth noticing [69] from which the authors use evolutionary techniques to evolve agents so to balance an estimated challenge rating for the player skill. The take-away idea is that of making agents with a minimal difference in skill (w.r.t the player) to survive more and, on the other hand, lower the fitness of the ones whose difference is larger.

[25] focused on the use of co-evolutionary algorithms (CEAs) proposing some methods and strategies for online evolution in an action (real-time) game. In this game, the NPC is evolved towards improving difficulty levels of gameplay. The author present four different methods: one that uses game specific information; one that merges offline-evolved data with online evolution; an two others that focus on using online data only and using offline and online data together. Considerable drawbacks in the approach are the slow rate of learning and the one-direct way of evolution, always toward the optimizing behavior.

[88] took a different approach by using Behavior Trees (BT), a kind of hierarchical finite state machine for controlling NPCs, in order to dynamically adjust the difficulty in a 2D fighting game. They basically tested two different approaches. First, the traditional approach of defining predefined behaviors (encoded on BTs) and finding a way to switch between models (behavior) at run-time. For this, they developed an algorithm for selecting BTs based on perceived conditions. The second method, however, adjusts difficulty by changing BT properties themselves at run-time, such as the probability associated with children of selector nodes. This last method, was the one that obtained high capability in balancing the game. As claimed by the authors, their method served as a proof of concept for a small sized scenario (simple 2D fighting game) leaving the possibility for further investigation in larger game scenarios. Here, pops out an opportunity for testing on a PIRG context, perhaps by following the current trend on applying such technique in Robotics [63, 73, 87].

In summary, approaches addressing DDA must prior to the definition of the challenge function identify important game characteristics affecting game difficulty, i.e., challenge level. This is where the knowledge engineering resides. The overall philosophy is the adjustment of the game behavior to the perceived player's game skill, which should be done as seamless as possible. A good survey about adaptivity challenges in games and simulations is that of [57]. The reader may refer to it in order to see a deeper treat on the identification and discussion of main challenges associated with the domain and also promising directions for research.

2.3.3 Focus on affection aspects

Also, some trend has emerged aiming on assessing the player's affective state (mainly emotion) as a mean of measuring the quality of interaction provided by the game. Here, the goal is related to the task of inferring internal traits of the player, such as personality and preference, during game-play [109]. Methodologies that take this approach are more towards a research domain known as *affective user modeling*, where the key point is that of assessing, heavily by using affection models, the inner state of the player regarding motivations for actions either based on action selection or through physiological data [109].

Being able to identify emotional profiles in this domain is useful given that they target a direct characterization of enjoyment level. For instance, after realizing that the player is compatible with an extroverted and highly responsible profile, the game AI may engage the user in rapidly and repeat-

edly shifts of events, like from calm situation to a long and intense chain of events, or even an steeper increase on motion abilities for NPC agents when proximity aspects, for instance, are important [8].

Working on this, [106] proposed a framework to estimate player enjoyment preference from physiological signals in a car racing game. They collected 5 physiological signals from players using the propComp Infiniti device[10], such as: Blood Volume Pulse (BVP) and Electrocardiogram (ECG), both sampled at 2048Hz, as well as Galvanic Skin Response (GSR), Respiration (RESP) and Temperature (TEMP), all of the three sampled at a rate of 256Hz. From those basic signals, they were able to extracted features such as: Heart rate (from ECG and BVP); magnitude and duration of GSR; expiration/inspiration time, apnea in/out time, respiration interval (all three from RESP); upper/lower envelope of BVP. Their data analysis, using Linear Discriminant Analysis (LDA), showed correlation between reported enjoyment and the described features, motivating further application on the use of biological signals without making any assumption on players' in-game activity.

Yannakakis and Hallam have published lots of papers on the matter of capturing and modeling affective state of entertainment, targeting children physiological state during physical game play. In [124, 128], children's heart rate, blood volume pulse and skin conductance were analyzed in the Playware prototype playground [58]. Their findings suggested that a higher average and maximum heart rate, a steeper blood volume appear to correlate with higher levels of reported entertainment in children of 8-10 years-old. Essencially, the main effort is related to modeling entertainment based on selecting a minimal subset of individual features that are able to construct the quantitative user model for predicting the children's reported entertainment preferences. In doing so, they tested Large Margin algorithms (based on the SVM principle) and Evolving Artificial Neural Networks. Successive works investigated feature selection and extend the approach for computer games [**yannakakis_entertainment_2008B**, 123, 125, 127].

There are indeed plenty of papers proposing the use of physiological method in game research. A widespread understanding is that measures from those methods are known for providing sensitive ways to assess the game experience, but they are hard to deal with since very often they require controlled experiments. In order to know more about this trend of research, the reader may refer to [49] whose work presents a review focused on psychophysiological methods for game research.

[10] `http://thoughttechnology.com/index.php/procomp-infiniti-308.html` accessed on March, 24, 2016.

2.3.4 Existing review papers and taxonomies for player modeling in games

Taxonomies and reviews for player modeling has been also proposed recently. In [94], under the emphasis that "player modeling" is a lose concept[11], the authors introduced a broad taxonomy for the purpose of distinguishing between the major existing player modeling applications and techniques. Four facets were suggested: the scope of application, the purpose of use, the domain of modeled details, and the source of a model's derivation or motivation. The expectation involved is that the taxonomy would allow the identification of relevant player modeling methods for particular problems and clarify roles that a player model can take. As pointed out in the previous section, the work of [8] focus on player behavioral modeling via in-game measurement of the human player distinguishing four types of player models: actions models, tactical models, strategic models and player profiling. Through their examination, they noticed that those models are increasingly resource-intensive to construct, but they also have a increasing tendency to generalize better.

[59] also proposed and discussed a taxonomy for player modeling research gathering and organizing information from several different sources. They, in turn, tried to characterize the most important topics in the area expanding the discussion from [38] about the most common techniques, further presenting a new set of techniques. Additionally, they did an analysis of player modeling research possibilities in several game genres and also listed suitable game platforms for experimentation, discussing main characteristics.

Targeting a holistic view of player modeling with the aim of providing a high level taxonomy and discussion of key components of a player model, [129] cluster the field into either *model-based* approaches – those that are built on a theoretical framework – or *model-free* ones – those that refer to the construction of a model between player input and a player state representation by mainly relying on the modus operandi of exact science, through computational techniques related to AI and ML. Additionally, they present a discussion about inputs and outputs to computational methods described as well as applications and current key challenges the field faces which are correlated to the inputs and outputs to the mentioned computational models.

Focusing on the point of personalized game experience in general, [7] provides a motivation for the promotion of methodologies in the topic as well

[11] According to them, it can equally apply to everything from a predictive model of player actions resulting from machine learning to a designer's description of a player's expected reactions in response to some piece of game content.

Table 2.2: Summary of existing popular survey-like papers concerning player modeling. • stands for proposal of taxonomy, ⋆ stands for overview of literature and ▼, in turn, correspond to survey or review.

Paper	Type	Brief description
[94]	•	Build a broadly applicable taxonomy that can describe player modeling techniques across all games , both digital and non-digital, and in all games genres.
[8]	•	An overview of methods by detailing four distinct approaches for modeling behaviour of players, namely: modeling player actions, modeling player tactics, modeling player strategies and player profiling.
[129]	•,⋆	A holistic view of player modeling, a high level taxonomy and discussion of key components as well as a description of challenges currently faced in the topic.
[7]	⋆	Motivation concerns for the topic, an broader overview, and adaptive components for personalised games
[59]	▼,•	Presents a survey of the field, discussing the main concepts and proposing a general taxonomy.
[45]	▼	Highlight most relevant trends and directions of research for the task of designing personalisation in games.

as an extensive overview of scientific literature. To the former objective, they describe psychological foundations, the effect of satisfaction, the advantages to game development and requirements for achieving ambitions. To the extent of the overview, they go pretty much in the same room of what has been exposed in [8], however providing an intensive discussion about *components of personalized games*, namely: space adaptation, mission/task adaptation, character adaptation, game mechanics adaptation, narrative adaptation, music/sound adaptation, player matching and difficulty scaling. An additional discussion about the relationship between personalized gaming and procedural content generation as well as the generalization to other domains (such as ambient games, human-computer interaction) is also mentioned.

Similarly, the work of [45] touches the subject of surveying the most relevant trends and directions of research in personalisation of computer games, in the extent that it is a true multi-disciplinary problem requiring contribution from areas as diverse as artificial and computational intelligence, game studies, psychology, game development and human computer interaction. Their survey correspond to five ways that, according them, players are often distinguishable from each other: by preference, personality, experience, performance and in-game behavior. Their discussion also aims to identify key research avenues that require further exploration.

Table 2.2 characterize the most important points in the mentioned papers.

In summary, in this section I attempted to point out existing review that are useful to get more in-depth details about the discussed topic.

2.4 A brief discussion on aspects of behavior and activity modeling for PIRG design

As discussed previously, a desired goal for PIRG design is to have autonomous robots able to adjust behavior towards optimizing the player experience. In the direction of achieving this result, one may see some sub-tasks that are fundamental to the problem, i.e., one must *a*) deal with the maximization of the player's expectation about the agent rationale – allowing believable companion/adversarial roles to be effectively implemented; *b*) deal with timing requirements for the design of interaction – since asynchronous actions may hugely impact engagement; *c*) feature selection; and *d*) modeling framework – to the extent of selecting which one of the focuses presented on section 2.3.

[15] presented key aspects on timing issues he observed during the design of several robogames. Regarding those aspects related to the structure of the game:

- **Duration of the game**: This is a key-point since a game should reach its end in a time that guarantees to keep the players involved and to make them enjoy it. This greatly depends on the activity to be performed. In the PIRG case, the user's physical activity, i.e the workload, is the core of the interaction and given that it must be taken into account when defining the game duration so that the player can come to the end with a proper fatigue requirement. The reader is invited to think about the effect on engagement when this point is not well designed.
- **Duration of an in-game task**: In a game, a set of tasks should be achieved. Naturally, the duration of such tasks has to be bounded by the duration of the game, or a game match. So, the duration of in-game tasks can be appropriately defined in order to provide some pressure on the players such that it may be likely to engage them, rise their interest, since an appropriate level of pressure and anxiety is related to challenge. The take-away idea here is that limiting the time available for an in-game task is a key-point to make it challenging.
- **Duration of non-interactive activities**: In some games, there are activities that must be performed by a single player and so must have the right amount of time dedicated to them, i.e, a right period of time without any interaction from others (e.g, a solution of a problem, a recovering procedure). If those activities have to be done by human players,

enough time has to be left for their accomplishment, but not too much time that could make them less challenging. On the other hand, if those activities must be performed by the robot(s), the human player should somehow be involved at the same time in some other activity, or the time dedicated to them should be short enough in order to lower the likelihood of making the human player bored, but long enough to be credible w.r.t. the storyboard of the game.

Some timing aspects related to the performance of both the human players and robot were also mentioned in [15].

- **Reaction time**: The time each player needs to react to an external event can be a constraint to be considered in game design. The human player's reactions in a physical interaction can be instinctive, thus requiring few hundreds of milliseconds to be activated, or may require some cognitive activity (e.g., reasoning, recognition), whose duration may span also some seconds. In PIRGs, since they are often designed for a lively interaction, the cognitive load is usually relatively small, and a time around one-two seconds for a cognitive response from the human player in a challenging situation is often considered as appropriate. The reaction time of the robot player mainly depends on the time to recognize a situation, which is related to the time required to elaborate signals, which in turn depends on the complexity of the data to be analyzed and on the available computational power. Since PIRGs are targeted in principle at a mass market comparable to the one of video games, the robots should be low cost, with simple sensors, and the available computational power might be as low as the one provided by cheap processing systems, like Arduino or Raspberry pi, up to that of an external laptop, tablet or smart phone. This might be a time constraint to be considered in game design, possibly justifying the related delays in the story.

- **Actuation time**: Also actuation time may concern both the human and the robot player. For people, they might be constrained by some devices to dedicate time to perform an action (e.g., to perform a gesture with a WIIMote device). This might be desired to put some challenge in the game, and also to reduce the power of the human player w.r.t. that of the robot, so to make the game more even. For the robot, the actuation time might be a constraint given by the selected mechanical implementation, or might even be desired to reduce the power of the robot. For instance, if a robot could run fast enough to reach a target before a human player,

it might be the case to reduce its speed so that the player can compete with it with some possibility to win.

Some other timing aspects are related to the establishment of a relationship.

- **Opponent behavior detection time**: Playing with artificial entities is engaging if the human player forgets the status of the opponent and attribute to it some human-like abilities. In particular, human players would like to play with entities that show some intelligence and intentionality. A way to achieve this status is to understand why the entity is performing an action, and, in general, what is its behavior, and what it aimed at. This may require an amount of cognitive activity proportional to the complexity of the behavior. In the mentioned experiments turned out that a random behavior is perceived as not interesting: the player believes that it is not worth to spend time with a silly entity. A too complex behavior is perceived as a random one, mostly because in PIRGs there is not much time to reason in a cool way on all the aspects of the perceived actions. The good behavior is one that requires a short time to be detected: not too short to consider the robot as "too simple minded" to play with, but also not too long to dismiss the cognitive activity of trying to understand it while the player is confronting the robot.

- **Credibility**: Each action should have a motivation, and should be credible w.r.t. the perceived motivation. Timing of the action should be consistent with this. For instance, if the robot seems to take a decision about what to do, the consequent action should last until there is a good motivation to change it. For instance, if a autonomous robot would change its movement direction randomly, there would be no apparent reason to motivate the change, and the robot would be perceived as silly. This, of course, depends on the PIRG. In Jedi Trainer 3.0, for example, a random decision about the direction to take is consistent with what the robot is doing: trying to find a gap in the trainee guard.

- **Activity pace**: Each player is assumed to do actions with a purpose for the game. Since they are interacting, the activity pace should be similar: a different pace, a different time between the selection of subsequent actions, would be perceived as if one would be favored w.r.t. the other one. Uneven games, in one sense or the other, are usually not appreciated.

- **Timing perception**: In interaction, timing is a subjective perception, and it can be modified by the interaction mood, or media, or by external devices. If there is an exchange, its pace can be modified by a "modeling and lead" strategy. If there is a time limit to perform a task, the perception of its urgency might be increased again by taking a faster pace in all movement changes, or also by simply giving relevance to the time-to-end, e.g., by adding rhythmic lights, sounds, or clocks.

2.5 Conclusions

In this document, I have aimed on the presentation of relevant aspects and popular papers in the field of player modeling both to the extent of maximizing competitive advantage (section 2.2) and experience optimization (section 2.3). Despite the fact it is not supposed to be an exhaustive review of literature (for which the reader may refer to works pointed-out on section 2.3.4), it was enough for enabling the spot of interesting aspects one must consider when trying to achieve similar goals on PIRGs. The motivation for this document was that of providing some insights – based on related literature aspects – for the design of adaptive behavior in PIRGs, which more broadly represents the student's effort on going through the literature in order to ease the selection of a first approach towards tackling this objective.

Chapter 3

Foundations

3.1 Game environment

During work research we have proposed a new PIRG where the "playground" consisted of a rectangular area of 4m×2m where, on each corner, tubes (henceforth called "towers") were placed. Each tower is equipped with a button (which sits on the tower's cap) and four LED that can be progressively turned on, one by one. Each LED requires the button to be pressed for 2.5 seconds, meaning that the tower takes about 10 seconds of button push in order to light up all of the four LEDs.

The LEDs are supposed to display the progress of the human player in capturing a specific tower. For a given tower, after turning on all LEDs, it is said that the player has captured it; Figure **??** presents the towers that were used in the game. Button pressing time is cumulative and can be distributed on different moments – this means that the player will not lose his progress if he stops pressing the button before the tower is completely captured.

In order to win, the human player must be able to secure all the existing towers without letting a single one be knocked down by the robot. If, at anytime, a tower falls (because of the robot or player) the game ends and the human player is defeated.

The robot is able to move across the entire playground just as the human player and it is only constrained by the fact that an already captured tower, or one whose button is currently being pressed by the player, cannot be teared down. The player can also block the robot path by staying in front of it, causing the later to likely change target tower. Notice that, while the player is trying to capture a given tower, the robot can try to tear down another one.

By relying on the lasers scans the robot can perceive its environment, locate itself and the human player during the game, being able to perform

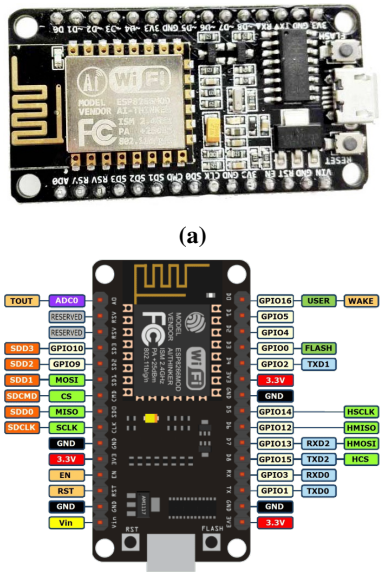
obstacle avoidance when appropriated.

This game is designed so that the robot, given its agility and maximum speed (1 m/sec) could always win, but it should show the appropriate and believable behavior to keep the player engaged and interested.

During the game, the robot must be able to track the human player movements while navigating the path through the various target towers mainly to collect information on the activity level of its opponent (chapter **??**). Details on the platform is given in section 3.3.

3.2 Towers

Each tower is powered individually, being capable of transmitting its status to the robot at a constant rate. The circuit uses the NodeMCU V3 ESP8266 ESP-12E WiFi module[1] whose connection is done via a private network. The communication between towers and the robot is support through TCP protocol using the rosserial_server[2] package. Figure 3.1 presents the board together with its pinout.



(a)

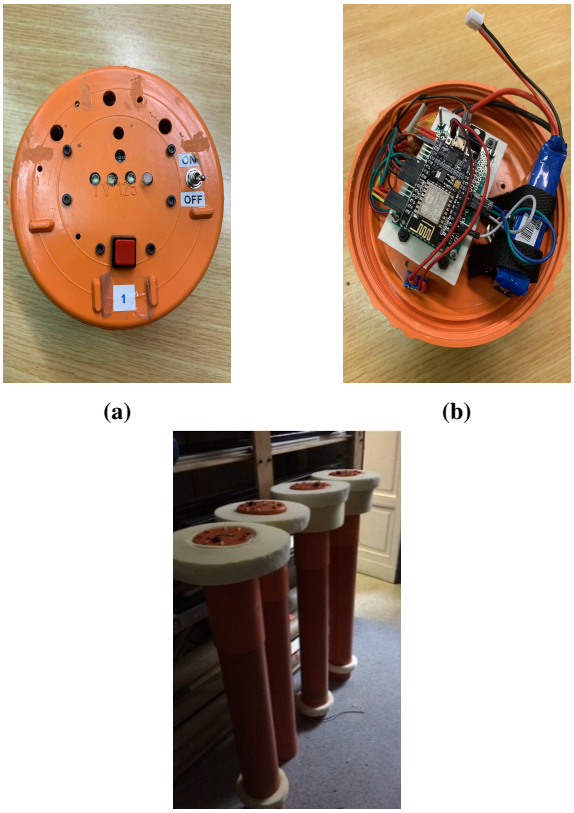
(b)

Figure 3.1: a) The NodeMCU V3 ESP8266 ESP-12E WiFi module used for data transmission. b) module pinout.

[1] `https://einstronic.com/wp-content/uploads/2017/06/NodeMCU-ESP8266-ESP-12E-Catalogue.pdf` accessed on December 17th, 2018.

[2] `http://wiki.ros.org/rosserial_server` accessed on December 17th, 2018.

A 7.4V LiPo battery is used on each tower as can been seen on figure 3.2. The nominal voltage for the boards is 5V, which is supplied through a voltage regulator. An angle switch allows the detection of fallen towers. A schematic of such event is provided in figure 3.3.



(a) **(b)**

(c)

Figure 3.2: a) Tower cap containing the button and LEDs; b) Tower cap circuit; c) The four towers used in the game (height 110cm).

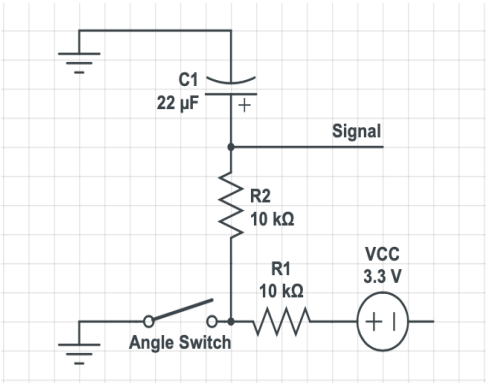


Figure 3.3: The circuit for detecting when a tower has fallen. An angle switch detects the inclination and the low-pass filter smooths out noise from vibrations such as those caused by the player touching the tower. The signal wire is attached to a pin on the NodeMCU V3 ESP8266 ESP-12E WiFi Module, which allows communication with the robot's onboard computer.

3.3 The robotic platform

We have designed an holonomic robot as experimental platform. This robot, called Triskar, is free to move in any direction at a speed comparable to that of people in indoor environments (up to 1.4 m/sec). In all prototypes, the base consisted of a metallic, triangular-shaped base where motors, batteries, computer and necessary electronics were embedded. Triskar has simultaneously and independently controlled rotational and translational motion capabilities all thanks to three omni-directional wheels actuated by a motor each.

During the research, we had designed several versions where the first two had versions an overall hight of 85 cm. The first three of which had a kinect sensor on top allowing for player tracking. Given the need to improve robustness, we have redefined the base by making structural changes toward vibration control and stability, which increased the overall high to 1 meter (3rd version). We also added new sensors such as planar laser scans needed to obtain reliable obstacle avoidance (3rd and 4th version). Figure 3.4 depicts our prototype evolution.

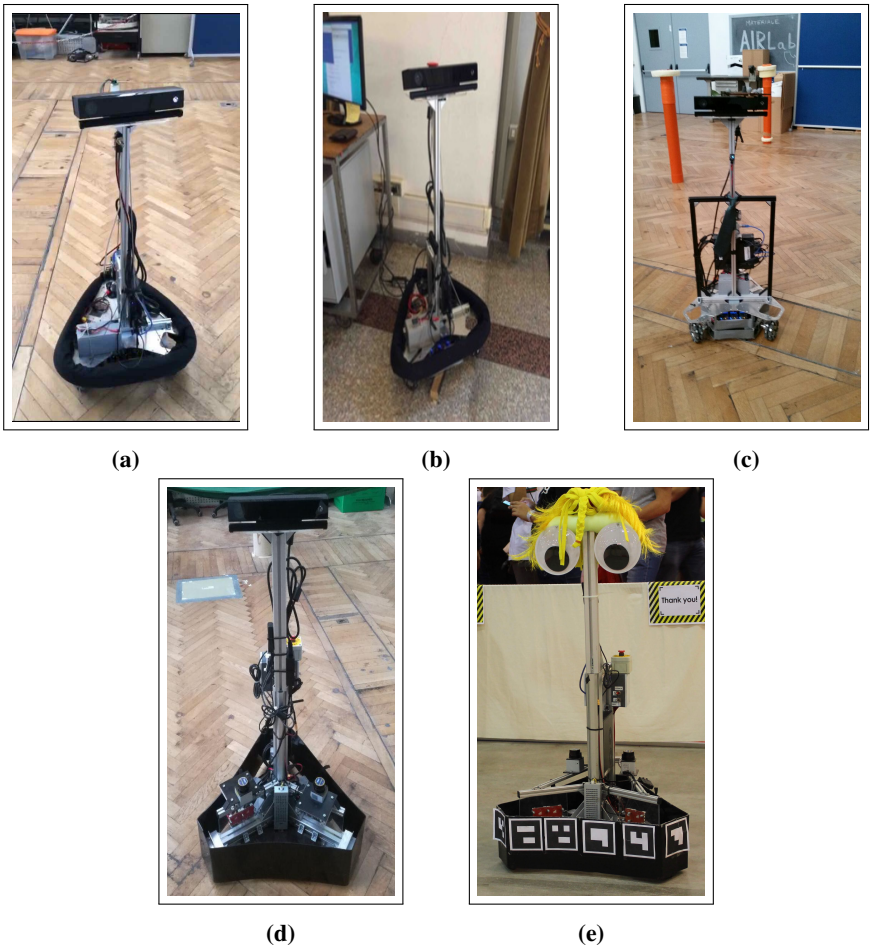


Figure 3.4: Prototype evolution. a) first version; b) second version differing on a button placed above the kinect; c) third version improving instability. d) fourth version including lasers on the base; e) current version during the Maker Faire 2018 in Rome (the European edition) from October 12th to 14th of 2018. A better placement of lasers has made the use of the Kinect unnecessary.

3.3.1 Sensors

Microsoft Kinect®

The Microsoft Kinect® sensor is a peripheral device that functions much like a webcam. However, in addition to providing an RGB image with its 1080p color camera, it also provides a depth map, meaning that for every pixel of the depth image provided by the sensor, Kinect provides the distance from the sensor. This makes the sensor suitable for a variety of computer vision

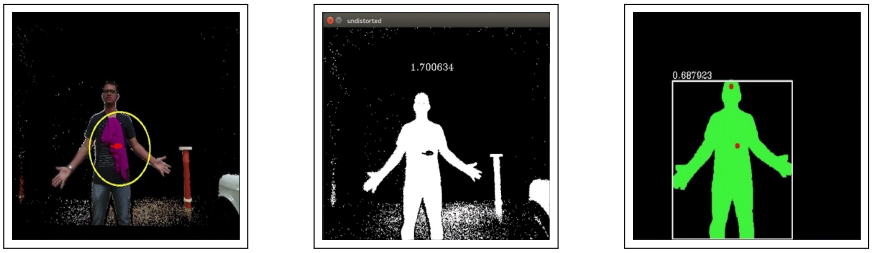
problems like background removal, blob detection, and people tracking.

Figure 3.5: Kinect® for Xbox ONE.

sensor dimensions	24.9 cm × 6.6 cm × 6.7 cm
sensor weight	approximately 3.1 lbs (1.4 kg)
sensor FOV	70° x 60°
depth sensing resolution	512 x 424
max - min depth	4.5m - 0.4m
working frequency	30 hz

Table 3.1: Kinect® sensor features.

We have used libfreenect2, a library for managing the Kinect device on Linux. We have then created a custom tracking node capable of estimating the player's position in two phases: First, color blob detection. Second, segmentation on the depth frame using *region growing* algorithm for the purpose of detecting the player's body. Two features can be detected from this procedure: distance (relative to the robot) and contraction index. The first is calculated from the mean value of the depth pixels around the center of the blob. The latter, in turn, is computed based on the subtraction of the occupied area with respect to the dimension of the bounding box that encompasses the segmented silhouette (see Figure 3.6).



(a) (b) (c)

Figure 3.6: Example of Frames processed by our algorithm. a) Point cloud showing the center of the detected (light-purple) color blob. b) Depth frame. The number showed above the user correspond to his estimated distance relative to the robot. c) Segmentation results. The number above is the contraction index defined in the interval [0,1].

Laser scanners

The robot was equipped with two laser scanners Hokuyo URG-04LX, shown in Figure 3.7. These laser sensors perceive the range of obstacles on a plan with a field of view of $240°$ and a resolution of $0.36°$, the maximum detectable distance is $5.6m$ and they can be connected to the computer by means of a USB interface, being operated with a nominal voltage of 5V.

The Hokuyo URG-04LX consists of a compact stacked structure with a spindle motor and the actual scanner on top of it. The motor rotates a small transmission mirror that deflects the vertical laser beam coming from the top of the sensor into horizontal direction. This allows the laser beam to scan a planar area around the sensor with an opening angle of $240°$. A second mirror below, the reception mirror, deviates the horizontal laser beam captured by a lens into vertical direction again.

Figure 3.7: An Hokuyo URG-04LX laser scanner.

A full scan is performed every 100 ms. The two laser scanners were mounted on each side of the lower chassis allowing for a $360°$ coverage around the robot.

3.3.2 Motors

The three motors are MAXON 118798 DC motor RE36 GB 70W KL 2WE, whose characteristics are reported in Figure 3.8 and Table 3.2.

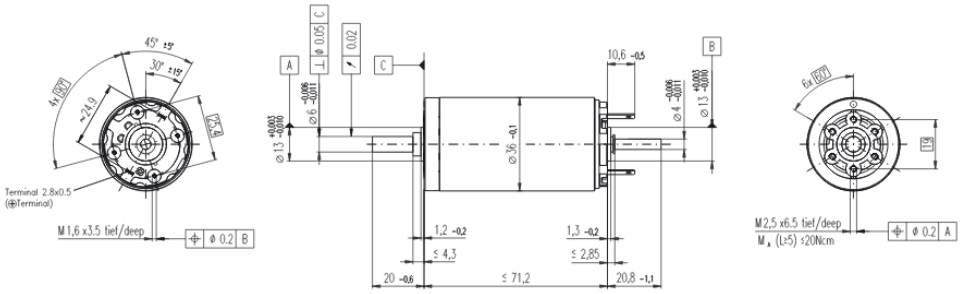


Figure 3.8: Robot motors (3 units)

Assigned power rating	70 W
Nominal voltage	24 V
No load speed	70 x 60
Stall torque	783 mNm
No load current	105 mA
Terminal resistance	1.11 ohm
Max. permissible speed	8200 rpm
Max. efficiency	85%
Torque constant	36.4 mNm/A
Speed constant	263 rpm/V
Mechanical time constant	6 ms
Rotor inertia	67.7 gcm^2
Terminal Inductance	0.2 mH
reduction ratio	[14 : 1] (166158 planetary gear GP32A 2.25NM)

Table 3.2: MAXON 118798 DC motor parameters.

Encoders

On each motor a 110513 tacho ENCODER HEDS 5540 500IMP 3K is mounted to get the speed of the motor, whose characteristics are reported

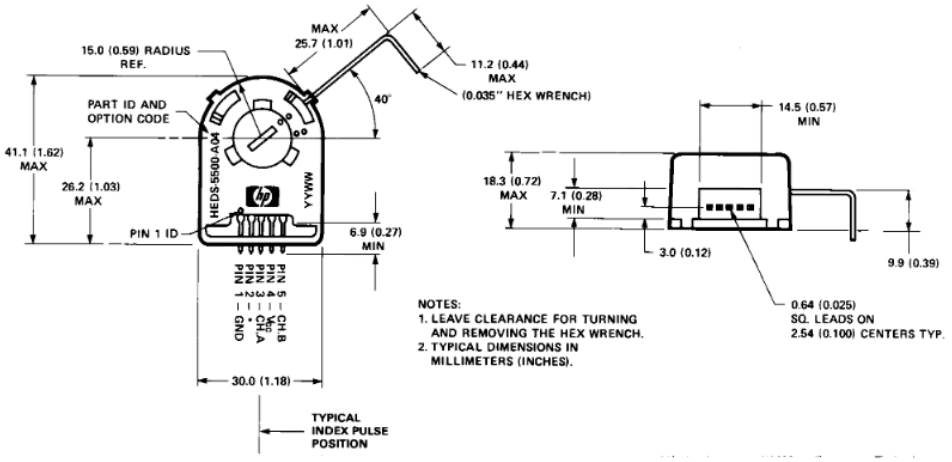


Figure 3.9: Motor encoders deployed for motion sensing and control.

in Figure 3.9. Encoders contain a single Light Emitting Diode (LED) as its light source. The light is collimated into a parallel beam by means of a single lens located directly over the LED. Opposite the emitter is the integrated detector circuit. This IC consists of multiple sets of photodetectors and the signal processing circuitry necessary to produce the digital waveforms. The code-wheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and bars on the code-wheel. The photodiodes detect these interruptions and send signals to the signal processing circuitry that produces the final outputs that is an index pulse P_O which is generated once for each full rotation of the code-wheel.

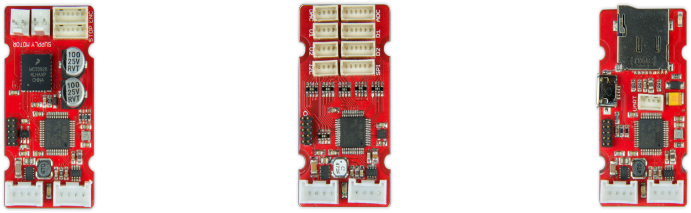
3.3.3 Onboard computer

For computing a Shuttle XPC Slim DH270 was used. The device has a $190 \times 165 \times 43$ mm steel case weighting 1.3 kg. The armature presents two holes for Kensington Locks and numerous threaded holes (M3) at both sides allowing for an easy placement. The operating system used was the *Ubuntu 16.04.3 LTS (64bit)*.

Figure 3.10: The on-board pc, Shuttle XPC Slim DH270: barebone Intel Kaby-Lake. Processor Intel Core i7; RAM memory of 16 GB DDR4.

3.3.4 The control boards

The low-level motors actuation and their interface between the ROS system have been realized with the Nova Core modules based on STM32-chip [68], which implement ready to use components to fulfill robot prototyping requirements with plug & play approach. The provided modules allow to control different type of motors that can be modeled as a second order system, where the input is the voltage applied to the motor armature and output variable is the motor angular speed. Futher details about the deployed boards are presented in figure 3.11.



(a) **(b)** **(c)**

Figure 3.11: a) UDC board (1 per each motor) capable of driving motors up to 70 W, with torque, speed, and position closed loop control. General attributes: 5-28V supply; 3A max (5A peak); current sense; encoder input; limit switch input; 25 x 45 mm in size. b) IO board (1 per each motor): Integrate existing hardware into the real-time Nova Core bus with analog and digital signals. General attributes: 8 digital GPIO; 4 analog inputs; 2 analog outputs; 2 UART; 1 I2C; 1 SPI; 25 x 45 mm in size. c) USB board (used for data collection) Interface the real-time Nova Core bus with a computer and logs data to microSD memory. General attributes: USB connector; UART connector; microSD card slot; rosserial support; 25 x 45 mm in size.

3.3.5 Kinematics

Kinematics is a branch of classical mechanics that describes the motion of points, bodies, and systems of bodies without considering the mass of each or the forces that caused the motion. The configuration of a rigid mobile robot is commonly described by six variables, its three-dimensional Cartesian coordinates and its three Euler angles (roll, pitch, yaw) relative to an external coordinate frame [105]. If the robot is considered as moving on a planar surface, this reduces to two Cartesian coordinates and an orientation angle. The kinematic model of an omni-directional base consists of an equation of motion of the robot in function of wheels velocity without considering the forces acting on the system. In this section, a kinematic model for a 3-wheeled omni-directional robot will be derived, while the 3-wheeled omni-directional robot used for this thesis work will be described in deep in chapter **??**.

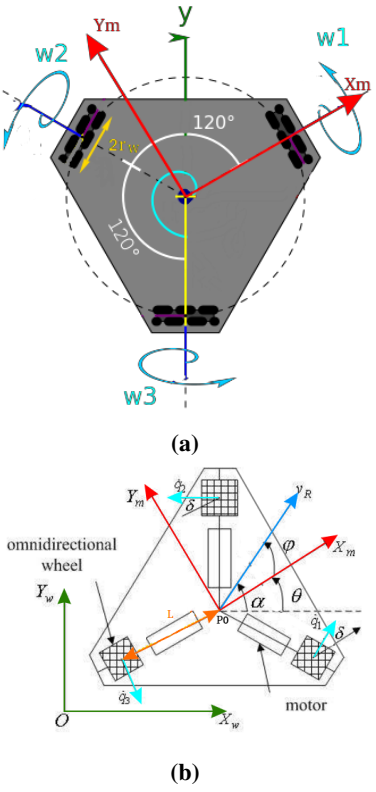


Figure 3.12: Kinematics diagram of the base of an omnidirectional robot. a) omni-directional base wheels displacement angle. b) omni-directional base reference frames and velocities.

Considering the figure (3.12b), each wheel of the robot is driven by a DC motor and its center has the same distance L to the robot center of mass P_0. we define the fixed world coordinate system [X_R, Y_R] and a mobile robot fixed frame [x_R^m, y_R^m] that is parallel to the floor and whose origin locates at P_0.
The robot's orientation is denoted by angle θ, which is the direction angle of the axis X_m in the world coordinate system (positive in the counterclockwise direction) and δ refers to the wheel orientation in the robot coordinate system and it is equal to 30 degrees in our considered example.
α and ϕ denote the direction of the robot translation velocity v_R observed in the world and robot coordinate systems, respectively.
We consider $\mathbf{v} = [\dot{x}_R^m, \dot{y}_R^m, \omega]^T$ the robot velocities observed in the robot coordinate system;
$\dot{\mathbf{q}} = [\dot{q}_1, \dot{q}_2, \dot{q}_3]^T$ is the vector of wheel velocities equal to the i-th wheel radius r_ω multiplied by the wheel angular velocity.

We introduce the transformation matrix from the robot coordinate system to the world coordinate system:

$$ {}^{w}R_m(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.1} $$

We can transform from robot to world coordinates system as:

$$ \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_R^m \\ \dot{y}_R^m \\ \omega \end{bmatrix} \tag{3.2} $$

P_0 denotes the position of the center of mass with respect to the world frame as:

$$ P_0 = \begin{bmatrix} x_R \\ y_R \end{bmatrix} \tag{3.3} $$

The position $[x_i \quad y_i]^T$ of each wheel can be given with respect to the center of mass of the robot, for i=1,2,3:

$$ P_i = \begin{bmatrix} x_{Ri} \\ y_{Ri} \end{bmatrix} = {}^{w}\, R_m(\theta) \cdot L \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{3.4} $$

Again, considering that the wheels present a displacement of 120° between each other we can deduce the following three vectors:

$$\begin{cases} P_1 =^w R_m(0) \cdot L \begin{bmatrix} 1 \\ 0 \end{bmatrix} = L \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ P_2 =^w R_m(\frac{2\pi}{3}) \cdot L \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{L}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix} \\ P_3 =^w R_m(\frac{4\pi}{3}) \cdot L \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -\frac{L}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix} \end{cases} \tag{3.5}$$

We now define the normal unit vectors of each wheel, representing the translational direction, as follows:

$$D_i = \frac{1}{L} R(\frac{\pi}{2}) P_i \qquad i = 1, 2, 3 \tag{3.6}$$

$$\begin{cases} D_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ D_2 = -\frac{1}{2} \begin{bmatrix} \sqrt{3} \\ 1 \end{bmatrix} \\ D_3 = \frac{1}{2} \begin{bmatrix} \sqrt{3} \\ -1 \end{bmatrix} \end{cases} \tag{3.7}$$

Then the translational velocity q_i as depicted in figure 3.12b can be written in the robot reference frame as follows:

$$\begin{cases} \dot{q_1} = \cos(\delta)\dot{x_R^m} + \sin(\delta)\dot{y_R^m} + L\omega \\ \dot{q_2} = -\cos(\delta)\dot{x_R^m} + \sin(\delta)\dot{y_R^m} + L\omega \\ \dot{q_3} = -\dot{y_R^m} + L\omega \end{cases} \tag{3.8}$$

The kinematic model with respect to the robot coordinate system is given by:

$$\begin{bmatrix} \dot{x_R^m} \\ \dot{y_R^m} \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\delta) & \sin(\delta) & L \\ -\cos(\delta) & \sin(\delta) & L \\ 0 & -1 & L \end{bmatrix}^{-1} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} \tag{3.9}$$

$$\begin{bmatrix} \dot{x_R^m} \\ \dot{y_R^m} \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix}$$

If we now consider the equation 3.2 the kinematic model with respect to the world coordinate system is described as:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{2}{3}\cos(\theta+\delta) & -\frac{2}{3}\cos(\theta-\delta) & \frac{2}{3}\sin(\theta) \\ \frac{2}{3}\sin(\theta+\delta) & -\frac{2}{3}\sin(\theta-\delta) & \frac{2}{3}\cos(\theta) \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} \tag{3.10}$$

where $\dot{\mathbf{x}} = [\dot{x}_R, \dot{y}_R, \dot{\theta}]^T$ is the robot velocity vector with respect to the world coordinate system;

It is important to notice that the transformation matrix in model 3.9 is full rank, which denotes that the translation and rotation of the robot are decoupled, and guarantees the separate control of these two movements.

Low level actuation (such as velocity control of the wheels) is embedded in the robot boards and, after considering figure 3.12a and figure 3.12b, we can define a system of equations that describes the angular velocity of each wheel.
If we define $[\omega R_1; \omega R_2; \omega R_3]^T$ as the wheel angular velocities vector, we have:

$$\begin{cases} \omega_{R1} = \dot{x}_R^m \cos(\delta) + \dot{y}_R^m \sin(\delta) + \omega L \\ \omega_{R2} = -\dot{x}_R^m \cos(\delta) + \dot{y}_R^m \sin(\delta) + \omega L \\ \omega_{R3} = -\dot{y}_R^m + \omega L \end{cases} \tag{3.11}$$

This can be written in matrix form as:

$$\begin{bmatrix} \omega_{R1} \\ \omega_{R2} \\ \omega_{R3} \end{bmatrix} = \begin{bmatrix} \cos(\delta) & \sin(\delta) & L \\ -\cos(\delta) & \sin(\delta) & L \\ 0 & -1 & L \end{bmatrix} \begin{bmatrix} \dot{x}_R^m \\ \dot{y}_R^m \\ \omega \end{bmatrix} \tag{3.12}$$

As previously anticipated we can also say:

$$\begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = r_\omega \begin{bmatrix} \omega_{R1} \\ \omega_{R2} \\ \omega_{R3} \end{bmatrix} \tag{3.13}$$

It is also known that a further relation between motor velocity and wheel velocity exists and is given by the equation:

$$\omega_{Ri} = \frac{r_\omega}{\eta N} \omega_{mi} \tag{3.14}$$

for i=1,2,3 where r_ω is the wheel radius, N is the coupling factor and η is the wheel/motor coupling efficiency factor.

The direct kinematic for an holonomic robot can be finally written as:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{2}{3}\cos(\theta+\delta) & -\frac{2}{3}\cos(\theta-\delta) & \frac{2}{3}\sin(\theta) \\ \frac{2}{3}\sin(\theta+\delta) & -\frac{2}{3}\sin(\theta-\delta) & \frac{2}{3}\cos(\theta) \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \frac{r_\omega}{\eta N} \begin{bmatrix} \omega_{m1} \\ \omega_{m2} \\ \omega_{m3} \end{bmatrix} \tag{3.15}$$

Inverse Kinematic: If we reverse and re-arrange equation 3.15 we obtain:

$$B = \begin{bmatrix} \frac{2}{3}\cos(\theta+\delta) & -\frac{2}{3}\cos(\theta-\delta) & \frac{2}{3}\sin(\theta) \\ \frac{2}{3}\sin(\theta+\delta) & -\frac{2}{3}\sin(\theta-\delta) & \frac{2}{3}\cos(\theta) \\ \frac{1}{3L} & \frac{1}{3L} & \frac{1}{3L} \end{bmatrix} \tag{3.16}$$

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} = B\frac{r_\omega}{\eta N} \begin{bmatrix} \omega_{m1} \\ \omega_{m2} \\ \omega_{m3} \end{bmatrix} \tag{3.17}$$

$$\begin{bmatrix} \omega_{m1} \\ \omega_{m2} \\ \omega_{m3} \end{bmatrix} = \frac{\eta N}{r_\omega} B^{-1} \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta} \end{bmatrix} \tag{3.18}$$

Equation 3.18 represents the inverse kinematic model for the considered holonomic robot that bounds the robot velocities in the inertial reference frame to the actual motors velocities.

Chapter 4

Exploring Activity Recognition in Robogames

On a PIRG, one is often interested in modeling the human player behavior and, from such modeling, program the system to respond properly. Our research began by exploring activity recognition and how that could be incorporated on the design of better PIRG's.

In the literature, some approaches for activity recognition are based on the analysis of data coming from camera devices. Others, instead, focus on data coming from wearable devices, such as: cellphones, watches and Inertial Measurement Unit (IMU) – a specific-purpose device for measuring accelerations and rotation angles. Across years, the type of analysis carried out has pretty much shifted from basic exploratory analysis, e.g., summary statistics, to complex data-based models, such as those from the ML community. The advantage of this is in the fact that ML models often grant better results by the characterization of the hidden structures present in data.

One may spot at least two groups into which to classify works in the area: those performed on an off-line manner and those performed on an on-line one. The first focus on the exploitation of methods for the case were data are first collected, tagged and them processed. The philosophy behind on-line methods, as the name suggests, rely on methodologies that can process the data on the go, providing real-time information about the estimated activity. In all of the vast literature in this field, ML-based approaches rely on some kind of transformation to the data in order to increase recognition results.

In the following, we present how we exploit a simple transformation of the input space targeting the creation of a flexible, generic, and well defined framework for activity recognition in a PIRG scenario. To the best of our knowledge, our proposal is the first one to incorporate activity recognition

on a Physically Interactive Robogame involving a human and a mobile robot in a adversarial game setting.

4.1 Introduction

Detailed measurement and classification of an individual's physical activity is fundamental in order to understand the relationship between physical activity and health but, also, to achieve an enhanced level of interaction between humans and robots. In particular, when considering Physically Interactive Robogame [62] – where the objective is the exploitation of both the real world as environment, and of one or more real, physical, autonomous robots as game opponents or companions of human players – activity recognition plays a fundamental role to adapt the robot strategy to support the player's entertainment during the game.

Here we propose a model which aims at classifying player's activity in a PIRG game using a 3-axis custom accelerometer positioned on the player's chest. We define a set of high level activity classes that are automatically classified relying on a supervised machine learning framework. Our methodology consists of transforming the raw input space into one that is able to capture variance of the signal to emphasize the recognition of the target activities.

Our main contribution is on the fact that we are able to obtain reasonable results in accuracy by applying a simple transformation. The achieved results are comparable, in terms of accuracy, with sliding window approaches, which makes our method feasible for real applications, but at lower cost regarding data processing. We test out methodology on activity recognition during a real robogame scenario (see section 3.1).

The chapter is organized as follows: we first present some related works about activity recognition and classification; In section 4.2.1 we explain how we collected data, while in section 4.3.1 we provide the description of the activity model. The results are finally discussed in section 4.4.

4.2 Related Works

A number of recent studies have investigated activity recognition using one or more accelerometer placed in different parts of the body. In [78], a well-cited paper in the activity recognition community, the authors have used a triaxial accelerometer worn near the pelvic region in order to classify eight different daily-life activities: *standing*, *walking*, *running*, *climbing up stairs*, *climbing down stairs*, *sit-ups*, *vacuuming*, and *brushing teeth*.

The windows size used had 256 data-points with 128 samples overlapping (50% overlap) between consecutive windows. At a sampling frequency of 50Hz, each window represents data for 5.12 seconds. The ammount of overlap used is justified by the work of [10], where they had demonstrated the feasibility of such overlap.

Considering a game environment, [43] investigated sensor placement and modality for activity recognition within the context of children's playground activities. By mean of parallel sensing, performed using a set of smartphones, activity dependent data have been generated. The obtained set of data was then used to train decision tree classifiers. This study shows once again that phones placed closer to the core of the body generate better models than phones placed on the extremities.

Similarly, in [2] a stochastic approximation framework for intensity independent activity recognition based on clustering techniques is proposed. The aim is to enhance and automate the calculation of Metabolic Equivalent of Task (MET) and also to improve an exergaming (video games that are also a form of exercise) platform consisting of two main components: an accelerometer-embedded belt and an Role-Playing Game (RPG) video game called *FreedroidRPG* that was used as an incentive for the participant to perform physical activity throughout the day. The study shows the ability of the used stochastic approximation framework to extrapolate unknown intensity levels from a few known ones that can be used to enhance activity recognition.

Several studies in literature also focused on the comparison between multi-sensor versus single-sensor activity detection and also on the optimal body placement of such sensors.

The work of [34] compared two distinct types of wearable systems: single-sensor wearable systems adopting complex algorithms and multi-sensor systems employing light-weight algorithms. The impact of the sampling rate on the recognition accuracy was then investigated using four classifiers. The experimental results illustrated that the recognition accuracy was steady at 50-Hz and above, and the single sensor system was more sensitive to the sampling rate than the multi-sensor system.

The work of [107] is, in turn, focused on making a comparison between the activity recognition rates of an activity classifier trained on acceleration signal collected on the wrist and hip. During the experiments 52 children and adolescents completed 12 activity trials that were categorized into 7 activity classes: *lying down*, *sitting*, *standing*, *walking*, *running*, *basketball*, and *dancing*. As result, the hip model exhibited great classification accuracy for *sitting*, *standing*, *walking*, and *running*; acceptable classification accu-

racy for *lying down* and *basketball*; and modest accuracy for *dance*. The wrist model, in turn, exhibited excellent classification accuracy for *sitting*, *standing*, and *walking*; acceptable classification accuracy for *basketball*; and modest accuracy for *running*, *lying down* and *dance*.

The present work, follows the line of reasoning present in popular ones, such as [10, 78], where the methodology relies on the use of supervised learning methods, powered by the extraction of discriminative features from a sliding window. However, the use of such method in our application scenario rises same special issues: since the annotated data come from real game interaction between a human and a mobile robot, the activities are not limited in any aspect.

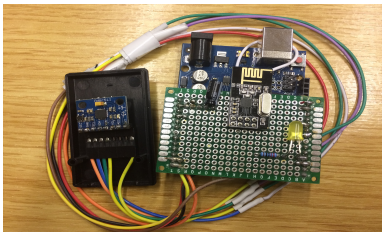
Additionally, when using a traditional sliding window method, special attention has to be made on the choice of window size. This is because while a too narrow window will produce very accurate representations of the current state, it will also be heavily affected by noise. On the other hand, a too wide window results in more stable, yet similarly inaccurate results due to the effects of change in the underlying data [12].

Equally concerning, fixed-size windows are likely to ignore differences in the duration of activities. This happens very often in our scenarios where different in-game situations would make the players react differently. For instance, the time the player spends running is a product of multiple factors, including personal motivations and robot state given in-game situation. This would demand adaptive strategies for using sliding windows, as suggested by [67]. However, in this work we follow a different route, by proposing to perform activity recognition by first transforming the data stream input space (acceleration raw data) into a different space that would capture the "turbulence" of the signal underlying each activity of interest. The motivation is that a running activity, for example, would be categorized by a different amount of turbulence compared to other activities.

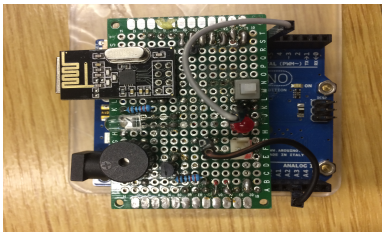
4.2.1 Data Collection

When playing the game (see section 3.1), we ask the human player to wear a colored robe (see figure 4.2) in order to allow for blob detection and tracking, leading to feature extraction. Those visual features, however, were out of the scope of this experiment, since we describe only results relying on the accelerometer data. For this last, we have used a custom accelerometer board attached to the player's chest in order to capture detailed player motion information. The device is based on the InvenSense MPU-6050 3-axis accelerometer board and an Arduino Uno micro-controller. The circuit

also contains a Nrf24l01 radio-frequency module that allows the accelerometer data to be sent to the on-board computer. Figure 4.1 shows our custom accelerometer device.



(a)



(b)

Figure 4.1: a) the accelerometer transmitter circuit and b) the receiver circuit used onboard. Both circuit are based on Arduino boards.

The choice of the accelerometer position was conditioned by the need to minimize the influence of noise from irrelevant player motion.

Figure 4.2: Human player (in magenta) during the game. The playground configuration consisted of 3 target towers.

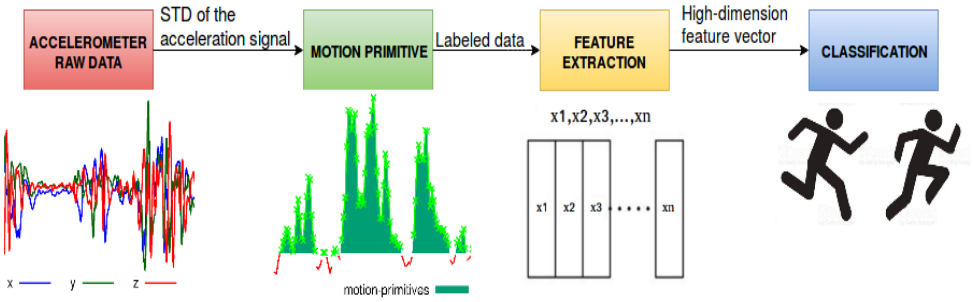


Figure 4.3: Overview of the activity recognition system.

For example, hand-waving when pressing buttons are not supposed to be meaningful to identify player activities – recall the reader in our game the full state of towers is transmitted to the robot via wireless communication. Therefore, having the device placed on the wrist, or other highly movable body-part (as the feet or head) would capture useless acceleration information contributing to a decrease in classification accuracy for the mentioned target motions. Stick with a more conservative region, such as the chests, in order to capture the essential player motion is reasonable.

In terms of collected data, for this work, we considered 29 matches involving 15 male participants of different ages. The age distribution consisted of children (7-10) and adults (26-40). Matches had a minimum time duration of about 40 seconds and a maximum of about 1 minute and 10 seconds.

The collected data correspond to acceleration values along x, y, and z axis with a sampling frequency of 50Hz, which is five times larger than the frequency considered to be sufficient for detecting daily activities from accelerometer data (10Hz) [6, 48, 78].

4.3 Method

4.3.1 Activity analysis

In our game, by using an accelerometer attached to the player, we were mainly interested on identifying activities that would help to describe the player interaction level. For the scope of this work, we aimed at identifying recurrent physical activities that would be useful for achieving that goal. From the collected data (see section 4.2.1), we were able to identify a few high-level activity types, listed below:

- **running:** describes a running activity. For this experiment, multiple styles of running are not considered. For instance, “fast” or “slow”

running are considered as the same.

- **walking/dodging:** represents the walking and dodging activity. The latter refers to a sudden quick movement to avoid the robot or to call its attention.

- **locally_moving:** a player generic motion that is too small to fall into other categories, but not so small to be characterized as inactivity. Robot path-blocking motions also fall into this type of motion.

- **inactive:** motion that are too low in intensity to be characterized as one of the above activities. A crisp threshold is used to delimit this category.

By inspecting the data, we observed that a given player activity would occur in "bursts", being followed by a short period of inactivity (or rest), i.e., a short period where the player is not really moving, or the expressed acceleration motion is too small to be related to any fundamental activity of interest.

Resting periods are a common characteristic present in any physical game, both related to the organic human need for resting after an intense physical activity or even during strategic moments of pause. Examples of that can be seen when the player is pushing a button on a tower, or is trying to block the robot's path or is even waiting still for a specific robot position on the environment.

Naturally, on such moments of inactivity, the changes in acceleration are usually small, which produces a relative flatness of the signal (e.g., secs 9-12 and 15-18 on figure 4.4) and allows for the delimitation of an activity begin/end time.

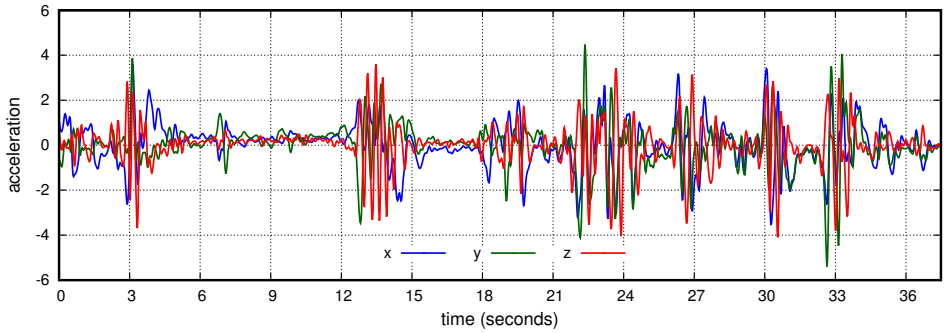


Figure 4.4: Graph of acceleration in x, y and z axis for a game that lasted about 40 seconds.

One way to describe the activity information carried on by the acceleration patterns is by considering the amount of signal "turbulence". As a measure of such turbulence, we rely on the information present on the signal variance.

For this, we process the incoming data stream by computing the standard deviation of the signal inside a sliding window. This transforms the original input space into a new space where activity information are much easily seen: resulting in the generation of a continuous graph, where pulses refer to a given player's physical activity (see figure 4.5).

Working this way turns out to be simpler than to perform data annotation by, for instance, using a predefined sliding windows size. In our case, activities, such as "running", do not have fixed time duration, which makes the applicability of fixed sliding windows methods not well suitable [67].

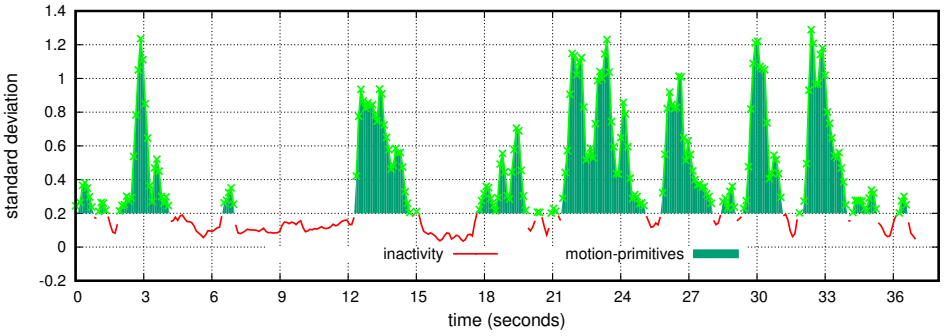
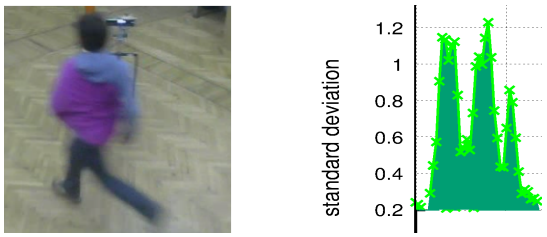


Figure 4.5: Standard deviation of the acceleration in Figure 4.4, computed using a sliding window of half a second. The red line portions represent variance values inside the inactivity zone (below a threshold of 0.2). Green areas are referenced as "motion primitives".

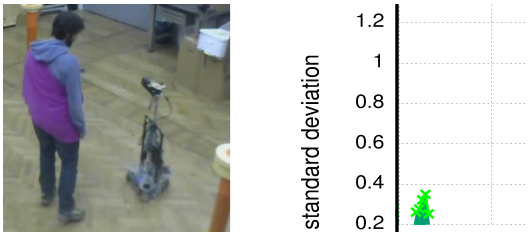
After performing the mentioned data transformation, we have empirically identified a crisp threshold that would catch irrelevant signal values, thus, we say that any value below that threshold is related to player's inactivity.

The inactivity threshold is game-dependent and sensitive to the position of the accelerometer as well. On our experiments, the selection of the inactivity threshold is done manually, via inspection of video recorded during the game.



(a) (b)

Figure 4.6: a) Player performing a running activity; b) Associated running motion primitive.



(a) (b)

Figure 4.7: a) Player performing local movements; b) Associated motion primitive.

Since any motion information below the threshold is used to determine the "inactivity" of the player, we use machine learning-based classification only on each data interval above the threshold. This interval is called a "motion primitive" (Figure 4.5). The motion primitives are the data aggregations to which we associate a class label in the annotation procedure.

Following the classification need, we have extracted the following descriptors from the motion primitives (recall, they refer to standard deviation of the raw values):

- **mean**: the mean values.
- **activity_time**: the time duration in seconds.
- **max_peaks**: the max peak.
- **number_of_peaks**: the number of peaks.
- **mean_of_peaks**: the mean of all peaks.
- **max-min**: difference between max and min.
- **std**: standard deviation.

- **mad**: median absolute deviation.
- **sma**: signal magnitude area.
- **energy**: the signal energy.
- **iqr**: interquartile range.
- **mean_over_max**: mean of peaks divided by the max peak.
- **maxInd**: index of the frequency component with largest magnitude.
- **meanFreq**: weighted average of the frequency components to obtain a mean frequency.
- **skewness**: skewness of the motion primitive.
- **kurtosis**: kurtosis of the motion primitive.
- **freq-skewness**: skewness of the frequency domain signal.
- **freq-kurtosis**: kurtosis of the frequency domain signal.
- **pse**: Power spectral entropy.
- **rms**: Return the root mean square.

Note that, the motion primitives carry sufficient information to distinguish between target physical activities. For example, "running" is related to a motion primitive that has a longer duration and a higher amplitude value (see figure 4.6) as opposed to the duration and amplitude values of a player local motion (figure 4.7). Following the same argument, a "walking" activity has higher amplitude values compared with a local movement. A holistic view of the classification approach is detailed in Figure 4.3.

4.3.2 Classification setup

As pointed out, for the automatic classification we first build the standard deviation graph from the raw accelerometer data, and then we manually label motion primitives.

On our experiments, the std graph was generated considering a sliding window 500msec long, resulting on a dataset composed by 367 motion primitives with 34% labeled as "locally_moving"; 25% as "walking/dodging" and 41% as "running".

Empirically, this time length turned out to be descriptive enough to produce variance intervals that made it possible to distinguish activities. Naturally, the windows size has an impact on the total number of motion primitives generated per game match, as well as on the inactive threshold value. With a windows size of half a second, however, it was possible to capture immediate transition between activities, given that such transition would manifest on higher spikes on variance at the beginning of an activity. For that reason, we kept the mentioned size.

As argued above, the "inactive" type is not considered in the recognition task since it is directly classified by the inactive threshold. From video log inspection, we observed that most of the useless motion would occurs below a threshold of 0.2.

Before training classifiers, we performed feature selection by evaluating the importance of the extracted features (see section 4.3.1) using random forest method, composed by 300 decision trees.

4.4 Results and discussion

We tested different classifiers using 10-fold cross validation in order to have a more descriptive accuracy information. Following common practice, the train-test dataset ratio was defined as 80% and 20% respectively.

Table 4.1: Cross-validation accuracy results for several classification methods using the 5 most significant features.

Method	Accuracy
SVM (Linear Kernel)	0.80 (+/- 0.08)
Random Forest	0.81 (+/- 0.06)
Gaussian Naive Bayes	0.80 (+/- 0.11)
Ensemble (Hard voting)	0.82 (+/- 0.10)
AdaBoost	0.65 (+/- 0.40)

Table 4.2: Classification report for the chosen ensemble method (Random Forest)

	precision	recall	f1-score	support
LM	0.88	0.91	0.89	23
WD	0.79	0.60	0.68	25
R	0.77	0.92	0.84	26
avg/total	0.81	0.81	0.80	74

Table 4.1 presents 10-fold cross validation results using different classifiers on the five most important features, that is: *rms*, *fft_energy*, *sma*, *max_peak* and *mean*.

The ensemble in Table 4.1 is defined as a majority (Hard) voting approach by the combination of the SVM, Gaussian Naive Bayes and Random Forest. The Adaboost method, in turn, takes a combination of 100 weak classifiers (Decision Trees). All methods were trained by using Python Scikit-learn machine learning library.

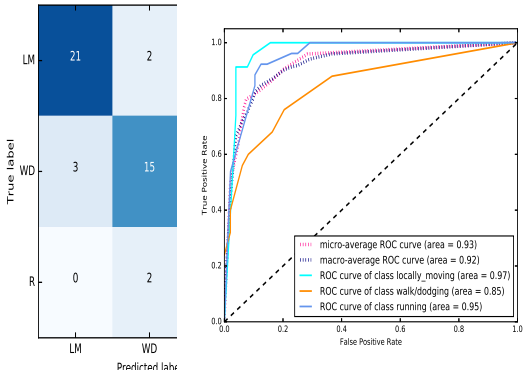
Despite the effort, with a confidence interval of 95% we see that SVM, Random Forest, Gaussian Naive Bayes as well as their ensemble have a similar accuracy result. By considering the variance in their result, we see that Random Forest gives the most stable result.

Given the 10-fold cross validation results, we decided to use as final method the Random Forest ensemble classifier (10 decision trees). Detailed results for the method are shown in table 4.2 and the corresponding confusion matrix and ROC in Figure 4.8. The majority of mistakes in the classification correspond to the difficulty in separating "walking/dodging" from a "running" activity.

The justification for this may be on the fact that occasionally the player walks in a fast way what causes an increase in similarity with proper running. This is acceptable given that even for humans it is not straight forward to decide about the boundaries conditions of two different but related activities.

Despite the fact that our method relies on the computation of a fixed slide window when transforming the input data, we see that our method also allows for a small improvement in the annotation procedure. Consider, for example, that we do not have to worry about the effects of overlapping windows. Things like the choice of windowing functions, which are used to mitigate the effects of overlap, are not needed. Also, it allows for a more intuitive way of perceiving what underlying activity has occurred.

We have already conducted experiments on using our method online, and the results have been satisfactory. One drawback, however, is on the possibility of having two different activities associated to the same motion primitive. This is likely to occur when the player rapidly shifts between two activities. For exampl, this happens when the player stops running for a fraction of a second and then immediately starts walking. In that case we see a small decrease in the signal variance that may not be enough to characterize an inactivity period. Note that a value below the inactive threshold is the event that separates motion primitives.



(a) **(b)**

Figure 4.8: a) Confusion matrix for the trained Random Forest ensemble method and the associated. b) ROC curve.

4.5 Conclusion

In this work, we have investigated the recognition of high-level human player activity in a Physically Interactive RobotGame (PIRG) scenario, where a human player faces a mobile robot. We have used the variance in player motion as data instances and primary source of information from where to extract features and then train a machine learning model.

Physically Interactive RoboGames are a rather new style of game and provide a specific setting from which to study human-robot interaction in situations framed by rules.

In order to design better PIRGs where, for instance, robots can adapt their strategies to support the player entertainment, a player activity model is desired, for which activity recognition is fundamental. We are currently working on the real-time applicability of this method as for the support of methodologies that are capable of quantifying the player engagement by, for instance, analyzing the rate of change in the number of activities performed by the player during the game.

Chapter 5

Player behavior modeling in Physically Interactive Robogame

In this chapter we present some ideas for player modeling in PIRG, beginning from the activity recognition system described in the previous chapter.

5.1 A simple activity model

In the possession of an activity system we aimed at the definition of an approach to quantify user behavior that can account for the combination of the player physical effort and interaction level. The proposed model was based on the human activity recognition (see chapter 4) and a description of interaction (proximity and body contraction index) relative to the robot co-player.

5.2 Relational Activity Model

As in [28], we defined as behavior a sequence of discrete time-bounded actions performed by the player. Since we are dealing with a physically interactive scenario, these time-bounded actions have strong relation with the amount of physical activity spent, hereby considered as the human "activity effort". The relationship with the robot, on the other hand, is considered the "relative interaction".

In the proposed model, the two dimensions (activity and interaction) are combined into a simple overall scoring system capable of describing quantitatively the player's active participation in the PIRG. The proposed model is detailed in the next.

Activity

Equation 5.1 is used to compute the general amount of activity $\alpha(m)$, given the classification of primitive motions from chapter 4:

$$\alpha(m) = \sum_{i \in \mathcal{A}} \omega_i \varphi_i(m); \qquad m \in \mathcal{M} \tag{5.1}$$

, where $\mathcal{A}$ stands for the set of activities {"running", "locally moving", "walking/dodging"}; ω stands for the activity weight; φ for the stochastic prediction value for the motion primitive m.

We consider equation 5.1 as a measure of the "effort" of the player. The weight ω states how much each physical activity contributes to the overall quantification and is a hyper-parameter.

5.2.1 Interaction

In order to model the interaction we take into account the spatial relationship between the player and the robot defined as "proximity" and a measure of body contraction named "contraction index" (CI).

Proximity is a measure defined in the interval [0, 1], computed from the data provided by the Microsoft Kinect®, normalized given the sensor specifications (see section 3.3.1). CI is also in [0 , 1] and it is calculated using a technique related to the bounding region, i.e., the minimum rectangle surrounding the body: the algorithm compares the area covered by this rectangle with the area currently covered by the player's silhouette [22].

A high CI is associated with an intense interaction because a wide openness of the body is interpreted as a higher focus level of the player w.r.t the robot and to the game; for this reason it will boost the final outcome. Low CI, in turn, is interpreted as a lack of focus or, in general, limited motivation of the player, which penalizes the interaction.

5.2.2 Combined model

The model to describe the player's relational activity is a combination of the activity level and the degree of interaction, named ϵ; it summarizes the player physical activity during the game, weighted by his interaction with the robotic opponent as follows:

$$\epsilon = \kappa \alpha(m), \tag{5.2}$$

where $\kappa = (1 - \rho)$ and uses the degree of interaction ρ so that when the output of the fuzzy system is zero (meaning no interaction) the activity level

is only the activity α. In other cases, κ is the multiplying factor that boosts the importance of the activity by putting it in relation with the amount of interaction. Notice, however, that ρ is an overall degree of interaction, i.e., the combination of CI and proximity. The exact procedure for the combination of the interaction signals is not bound to a specific method. In this work, we have used a fuzzy logic system to obtain the desired combination, described in a linguistic way. In principle, the combination of the signals depends on the specifics of the PIRG environment.

5.2.3 Evaluation

Data elaboration

We considered 29 matches involving 15 male participants of different ages. The age distribution consisted of children (7-10) and adults (26-40). This sample matches the typical user distribution for this kind of games, and arose in uncontrolled way from the presentation of the game to general public (including also differently aged people and females) in an exhibition context. The variety of behaviors collected in these trials was high and supported our hypothesis about the need of an adapting robot. In these matches, the robot was remotely driven by a human operator. Matches had a minimum time duration of about 40 secs and a maximum of about 1 minute and 10 secs.

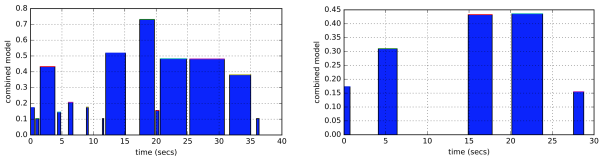
The collected data are the acceleration values along x, y, and z axis with a sampling frequency of 50Hz, which is five times larger than the frequency considered to be sufficient for detecting daily activities from accelerometer data (10Hz) [48].

In terms of dataset support, 34% of the motion primitives correspond to locally moving (LM); 25% to walking/dodging (WD) and 41% running (R).

We use a sliding window half a second long, with 10% overlap, for the STD graph generation. In our experiments, the produced spikes in variance have a huge discriminative power w.r.t. activities demanding sudden changes in behavior, e. g., a "running" motion. We also used a positive half sine wave as windowing function in order to soften the transition between windows. As mentioned, the "inactive" type is not considered in the recognition task since it is directly classified by the inactive threshold line.

Model results

The model output for two different players can be seen on figure 5.1. In figure **??**, the player is perceived as more active than the one in fig. **??**. The ratio between activity and inactivity exposed by the output matches the observed data and empirically confirms the model viability.



(a) **(b)**

Figure 5.1: a) Model results for a single match that lasted 45 secs. b) Model output for a less active player. The graph refers to 30 initial secs of activity.

The model quantitatively describe, in a simple way, the player's relational activity by the combination of the amount of physical activity and interaction relative to the robot. This can be used as a metric expected to enable the robot to take into account the player's attitude in order to adapt its own activity to match it. Using this model, for instance, an active player could be viewed as one having a larger cumulative sum of ϵ values up to a point in time.

However, this model is limited in several ways, one of such is the manual definition of weights for the activities (ω). Manually selecting those weight becomes difficult as the number of activities grows. We have then worked on a definition of a more robust player model relying on latent representation. This is describe next.

5.3 Latent player modeling

5.3.1 Introduction

We attempted to mitigate the limitations in the previous model by first mapping the stream of data into an image representation, namely a Gramian Angular Field (GAF) [118], which is subsequently processed by a simple autoencoder [35] architecture for unsupervised feature extraction. After the definition of a dictionary of primitives on the feature space, we apply latent Dirichlet allocation [13], a generative model mostly used for document retrieval and classification, to uncover coherent groups of motion patterns that can be used to categorize players.

Our decision to use Latent Dirichlet Allocation (LDA) takes inspiration from recent approaches where each game session is represented as a "document", and chunks of generated stream data are encoded into discrete "words" [95]. This arrangement is based on the assumption that different players generate different streams of input words, thus different documents, depending on their own motivation or play style.

The possibility to apply LDA to this problem is further reinforced by

noticing the apparent difficulty in objectively separating types of motion patterns, since a player in his/her playing activity may show one or more different motion styles. More explicitly, a player may, for instance, become bored or frustrated with the game and consequently show a low motion profile w.r.t. the one that the same player might have shown at other moments during the same game session; for instance, when he/she was excited about it. Here, we consider that a high motion profile, i.e., high turbulence in acceleration signal, is typical for a highly motivated player, since by common sense, a non-motivated player tends to stay relatively still or show a very low acceleration signal.

The claim that motion style conveys engagement or even emotion [5, 89, 108] types has been supported by several papers. As an example, systematic approaches for describing movement, such as Laban Movement Analysis (LMA) [51], are often used in deriving low-dimensional representation of movement, facilitating affective motion modeling [20]. This same type of analysis has been applied when investigating motor elements that characterize movements whose execution enhances basic emotions, such as: anger, fear, happiness and sadness [89]. In game situations, despite the limited capacity for entertainment detection and modeling in Exergames, LMA had been reported as useful in fostering emotion recognition in states like: concentration, meditation, excitement and frustration [130].

Here, we hypothesize that, on average, a player will display his/her game-intrinsic motion style and convey some information about his/her interest in/motivation to play. Discovering which coherent groups of motion style exist in a dataset allows us to estimate to what extent an unknown player relates in terms of his/her own motion profile to known groups. This may ultimately help in designing PIRGs that are able to adapt accordingly in order to offer a better user experience for the players.

Following this, in this manuscript, a player representation is addressed as how much of each discovered motion types to which we are confident to say a given player is mostly related. We call this representation a mixture proportion of motion types (similar to topics in the LDA perspective).

In summary, we expect to see that similar player-generated data are grouped together in a coherent collection, allowing the distinction of the existing motion types. The results obtained are validated with human supervision, suggesting the applicability of our method to future mechanisms for designing auto-adapting PIRGs agents.

5.3.2 Mathematical Preliminaries

This section presents the basic concepts and principles of the Gramian angular field, autoencoders and latent Dirichlet allocation, our targeted probabilistic graphical model.

Gramian Angular Field Images

The results obtained in this work rely on the transformation of the time series data generated by the player motion into images that can then be processed and understood in terms of emergent patterns that can, in turn, be discovered by state-of-the-art machine learning algorithms. The type of image we have chosen is Gramian Angular Summation/Difference Field (GASF/-GADF). Such images had been proposed in the field of time series classification [118], where the authors evaluated the efficacy of representing time series in a polar coordinate system instead of the typical Cartesian coordinates. For a deep understanding of the method, we suggest reading the original paper, where the authors did not only present the idea of GASF/GADF, but also reported experimental results comparing them with other time series imagery techniques. Here, we present an overview of the technique in order to familiarize the reader with such a method and to point out its role in our proposal.

In general, in order to obtain a GASF/GADF representation for a time series $\mathcal{X} = \{x_1, x_2, ...x_n\}$ of size $n \in \mathbb{N}_{>0}$, rescaling is first performed in order to put the time series in the interval $[\,-1, 1]$ or $[\,0, 1]$. The second step is to represent the rescaled time series $\widetilde{\mathcal{X}}$ in polar coordinates, where each sample value is now mapped into the angular cosine and the time stamp information is represented as the radius. The formula for this step is as follows:

$$\begin{cases} \phi = \arccos(\widetilde{x}_i) & -1 \leq \widetilde{x}_i \leq 1, \widetilde{x}_i \in \widetilde{\mathcal{X}} \\ r = \dfrac{t_i}{N} & t_i \in \mathbb{N}_{>0}, \end{cases} \tag{5.3}$$

where t_i is the time stamp and N plays the role of a normalizer for the span of the polar coordinate system. Some properties of this transformation are known:

- Property #1: The encoding map of Equation (5.3) is bijective as $\cos(\phi)$ is monotonic when $\phi \in [\,0, \pi]$. Given a time series, the proposed map produces a unique result in the polar coordinate system together with a unique inverse map.

- Property #2: As opposed to Cartesian coordinates, polar coordinates preserve absolute temporal relations.
- Property #3: Rescaled data in different intervals have different angular bounds. That is, $[0, 1]$ corresponds to the cosine function in $[0, \frac{\pi}{2}]$, while cosine values in the interval $[-1, 1]$ fall into the angular bounds $[0, \pi]$. These intervals are known to produce different information granularity in the Gramian angular field, especially for classification tasks [118].

The third step in completing the transformation is to exploit the angular perspective by considering the trigonometric sum/difference between each point. This allows for the identification of the temporal correlation within different time intervals. The two types of GAFs, namely GASF and GADF, are obtained as follows:

$$\begin{aligned} GASF &= [\cos(\phi_i + \phi j] \\ &= \widetilde{\mathcal{X}}' \cdot \widetilde{\mathcal{X}} - \sqrt{I - \widetilde{\mathcal{X}}^2}' \cdot \sqrt{I - \widetilde{\mathcal{X}}^2} \end{aligned} \tag{5.4}$$

$$\begin{aligned} GADF &= [\sin(\phi_i + \phi j] \\ &= \sqrt{I - \widetilde{\mathcal{X}}^2}' \cdot \widetilde{\mathcal{X}} - \widetilde{\mathcal{X}}' \cdot \sqrt{I - \widetilde{\mathcal{X}}^2} \end{aligned} \tag{5.5}$$

I is the unit row vector. After translating the time series from the Cartesian to polar coordinate system, each time step in it is taken as a 1D metric space. By defining the inner product as $< x, y >= x \cdot y - \sqrt{1 - x^2} \cdot \sqrt{1 - y^2}$ and $< x, y >= \sqrt{1 - x^2} \cdot y - x \cdot \sqrt{1 - y^2}$, respectively, the GAFs become quasi-Gramian matrices, since the defined $< x, y >$ do not satisfy the property of linearity in inner-product space.

Still, according to [118], the GAFs have several advantages:

- Advantage #1: They provide a way to preserve temporal information. The temporal correlation is present because $G_{(i,j||i-j|=k)}$ represents the relative correlation by the superposition/difference of directions with respect to time interval k. The main diagonal $G_{i,i}$ is the special case when $k = 0$, which contains the original value/angular information.
- Advantage #2: From the main diagonal, it is possible to reconstruct the time series back into the original Cartesian space. This is useful especially in a classification task, where we can convert the time series back to numerical Cartesian space from the high level features learned by, for instance, a deep neural network.

However, the size of the Gramian matrix G is $n \times n$, for n being the length of the raw time series. The authors propose a solution for this issue by attempting to reduce the size of the produced GAFs using, for instance, Piecewise Aggregation Approximation (PAA) [47] to smooth the time series while preserving the trends. An example of the conversion from time series to GASF image is shown in Figure 5.2.

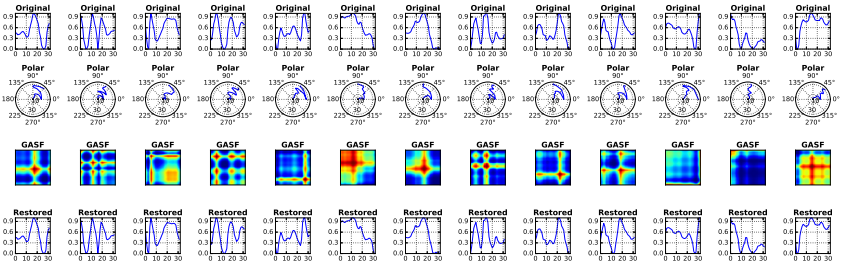


Figure 5.2: Conversion of time series into a Gramian Angular Summation Field (GASF) image and then back into its original space.

Latent Dirichlet Allocation

LDA is one of the most simple and influential topic models. A topic model is a formal statistical relationship between a group of observed and latent (unknown) random variables, which specifies a probabilistic procedure to generate topics. The latter, in turn, is defined as a probability distribution over a collection of words [79].

The graphical model shown in Figure **??** presents the structure of the LDA model [13]. Here, as in [95], the *K*-th probability distribution over input words β (called topics) is representative of the K groups (types) of player motion styles in which we are interested. In turn, each document d is a game play session for the game. We had to deal with the deterioration of model accuracy when considering the whole game section as a single document. Our solution for the problem was to segment each one of those into subsets based on duration in seconds. This approach raises several issues that are going to be properly addressed in a future work, but still allow one to obtain interesting results.

In these terms, a document may present multiple types β_k with mixture proportions θ_d This mixture of proportions defines a player at each instant of time. The choice of the N_D words comes from $\overline{\beta}_d$, which acts as the weighted average of the game play types for a given document. This average is defined as follows [13, 95]:

$$\overline{\beta}_d = \sum_{n=1}^{K} \theta_d(k) \cdot \beta_k \tag{5.6}$$

The *n*-th word $w_{d,n}$ of document *d* is, in our case, the GASF encoding obtained, as we will explain later. $z_{d,n} \in \{1 : K\}$ refers to the random type from which a word is drawn. Following the standard LDA model, α and η are hyperparameters that control the sparsity in the mixture proportions and word distributions [95].

Autoencoder

In many problems, we may have to deal with highly dimensional data that represent the situation of the world as captured by the sensors. Given that the raw data dimensionality could be practically intractable, the common procedure in these cases is to reduce the dimensionality possibly keeping only the important features of the input data.

In image processing, for instance, understanding which pieces of an image represent a significant aspect for the task ahead is not a trivial issue. Significance may be related to specific parts of the image and may depend on many aspects. A common technique to face this issue is autoencoder. Autoencoder is an unsupervised learning technique that learns a representation of the input data, usually derived by means of non-linear transformations of the input [35]. A typical implementation of autoencoder consists of a feedforward neural network that first encodes the input into a code, the representation and then reconstructs the input from it by means of a decoder (see Figure 5.3).

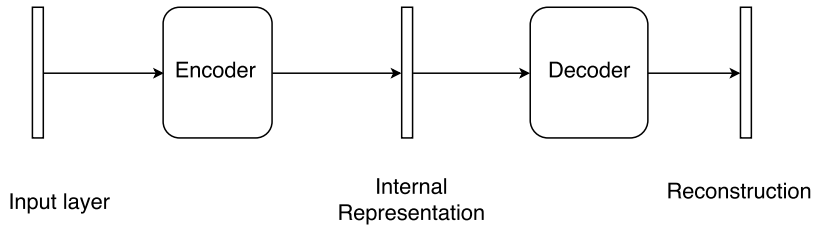


Figure 5.3: General architecture of an autoencoder; the input layer is processed by an encoder that transforms it. The derived representation is then decoded by a decoder that yields a reconstruction.

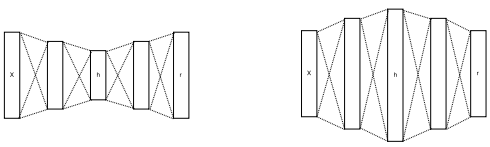
In general, autoencoders are good at capturing the structure of the input and tend to represent it well [35]. We can represent an autoencoder as a neural network that tries to copy its input to its output. We can represent its general architecture with the following terminology:

- the input x,
- the encoder function, $f(x)$, composed of some hidden layers,
- the code, or internal representation, of the input h,
- the decoder function, $g(x)$, composed of other hidden layers,
- the reconstruction obtained from the decoder, r.

A loss function, $L(x, r)$, should also be defined to evaluate how close the reconstruction is with respect to the input. A simple loss function is shown as Equation (5.7).

$$L(x, r) = \sum_i (x_i - r_i)^2 \tag{5.7}$$

However, if we do not impose any restriction, the transformation it learns would be an identity, since this would perfectly reconstruct the input. To prevent this transformation, two approaches have been devised [35]. The first, called undercomplete autoencoder, adopts a representation smaller than the input (see Figure 5.4a), while the latter, overcomplete autoencoder, uses a representation with a dimension larger than the input (see Figure 5.4b). Although increasing the representation size is counterintuitive in order to sparsely represent the input, this technique has proven its effectiveness.



(a) **(b)**

Figure 5.4: (**a**) Representation of an undercomplete autoencoder, where the representation has a lower dimension with respect to the input. (**b**) Representation of an overcomplete autoencoder, which uses a larger space to represent the input.

The undercomplete autoencoder is the oldest and best known architecture for the autoencoder and also the one we use in our work. The code the autoencoder derives is smaller than the input: it learns how to combine the input so that the relevant information is retained in a lower dimensional

space. The combination is often non-linear due to a non-linear activation function of the neurons. On the other hand, the decoder function learns how to unroll such compressed code in order to obtain an extended representation. This architecture could be trained using the classic back-propagation approach, although different methods are available [35].

The overcomplete autoencoder, instead, increases the dimension of the representation; however, as already mentioned, the model could easily learn the identity transformation. In order to tackle this issue, the common approach is to apply a regularization term over the weights of the network. In general, a l_1 penalization is added to the loss function. In this way, the representation will be sparse since some of the dimensions are set to zero. However, this new architecture would be more complex to train since the model would be larger than the previous one, although still providing a sparse representation. Despite its sparsity, this architecture is able to extract more information from the input, and it is often used to capture also the input distribution and not just the structure of x. It is not possible to estimate such a distribution with an undercomplete autoencoder since such a task would require a higher representation capacity.

In this work, we adopt an undercomplete autoencoder, since our goal is to represent an image in a lower dimensional space. Our architecture has multiple layers: the first layer takes only the input, while the following ones decrease in size following the power of two. We represent the input with a code having 64 dimensions; thus, from an input of 1024 features, we derive a representation of 64 components using an autoencoder.

The reconstruction of the input by the decoder function has a very low loss, below 0.05%. The layers obtained by the encoder are often difficult to understand by people; for instance, in Figure 5.5a–c, we can observe the first layer, and we cannot derive any structure or meaning from it. However, looking at deeper levels, the layers show a clear structure; in Figure 5.5g–i, we can observe that the filters are extracting horizontal and vertical patterns. Indeed, most of the original images show a cross pattern, or multiple crosses; therefore, this representation seems to be able to represent the input data with a lower dimension.

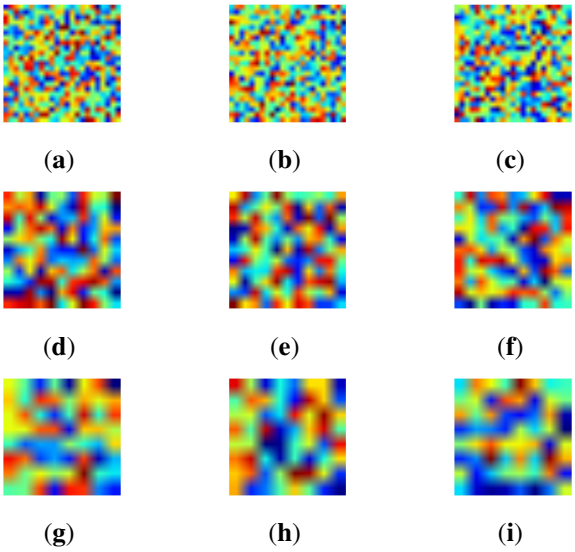


Figure 5.5: Example layers from the trained autoencoder. (**a**–**c**) represent the information extracted by the first layer of the autoencoder. It is hard to grasp the meaning of the representation derived. Going deeper, in (**d**–**f**), the representation seems to become more clear and with a structure. Indeed going directly to the obtained code, in (**g**–**i**), we can see a more accentuate structure composed mainly of horizontal or vertical lines.

5.3.3 Problem Statement

This section formulates the problem of modeling the player as a probabilistic mixture of uncovered player styles from data. There are at least three basic dimensions (or problems) of interest that must be considered when building a model of this kind for PIRGs. Such dimensions are related to the specific capabilities of a mobile robot to achieve a reasonable in-game performance. We delineate such dimensions as follows:

- Dimension #1: Fundamental robot behavior, which concerns the fundamental aspects of the robot locomotion, timing of actions, localization and other abilities to keep it alive and well-functioning during the game play.
- Dimension #2: In-game competitive capabilities, which are more related to the intelligent selection of actions aimed at achieving in-game goals. In another words, this dimension captures the intricacies of planning and rational behavior, using solutions for Dimension #1.
- Dimension #3: Adaptive power, which is a higher level dimension concerned with how and when during the game to adjust the robot perfor-

mance to match that of the current player.

Dimension #1 concerns the setup of an appropriate robotic platform, including aspects like: electrical disturbances, power consumption, kinematic constraints, processing capabilities, sensing and safety. These are recurrent issues that must be appropriately dealt with. In turn, Dimension #2 encompasses the problem of finding a plan or strategy to achieve the robot goals. This is the problem most related to the competitive behavior the robot can impose on the human player, since action selection is driven by the tendency to maximize the robot's own pay-off. After all, the robot has to give the idea that it is at best trying to beat/help its opponent(s)/companion(s).

Dimension #3, acting as the superset for #1 and #2, defines constraints on the robot behavior such that the action selection is not longer just governed by the tendency to be competitive/helpful, but is also driven by the compelling necessity to support the player interaction, in real time, as robustly as possible.

One can see that the present work as a step towards advancements in Dimension #3. Here, we exploit data coming from a single accelerometer placed on the human player's body; thus, player movement is our primary source of information for describing the playing style and engagement.

The basic assumption is that a highly active player would naturally express his/her status by a very turbulent motion signal that is compatible with strong in-game activity. Consider, for instance, the difference in signal shapes reported in Figure 5.6.

Following this line of reasoning, we also assume players do not fake their engagement by unnecessarily performing strong activities, like running, when they actually are not engaged in the game or feel bored by the robot behavior. Therefore, we assume that a highly motivated player would act more intensively and frequently than a non-motivated one, thus showing a more turbulent acceleration signal profile.

In situations like PIRG, movement information is useful and should not be disregarded. Furthermore, a model that is able to extract a description of the player's movement profile for the purpose of robot behavior adaptation would generate interesting dynamics able to hide existing platform limitations (kinematics constraints, tracking problems, etc.) and boost game experience.

One important aspect not considered yet by our assumptions is the role of skill level. For example, a highly skilled player could try to minimize his/her energy expenditure by performing precise movements towards winning the game and in theory could perfectly show a low-movement profile. Moreover,

these assumptions cannot directly measure player engagement in the game, given that it is also possible for a player to move not too much, but, at the same time, be highly engaged.

In summary, we are interested in investigating time series data, especially acceleration patterns as proxy and/or engagement cues (given the mentioned assumptions) towards the creation of auto-adjusting robotic companions or opponents.

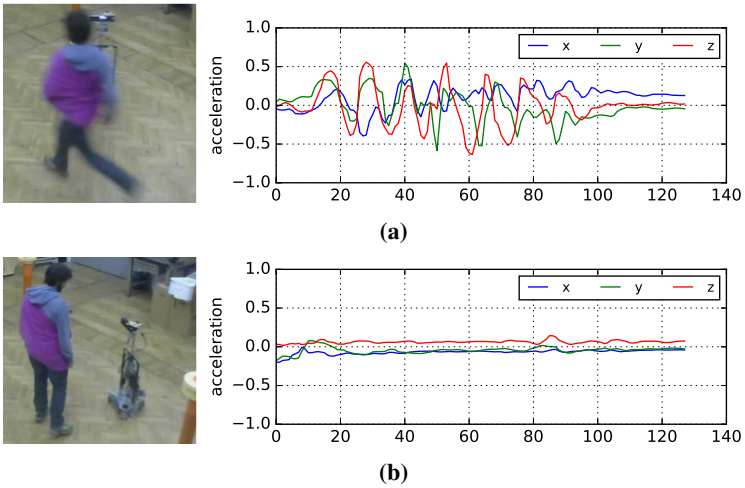


Figure 5.6: Three-axis accelerometer signal during game play. (**a**) A running player. (**b**) A player standing still. Notice that signal shape and amplitude are very characteristic for the respective activity.

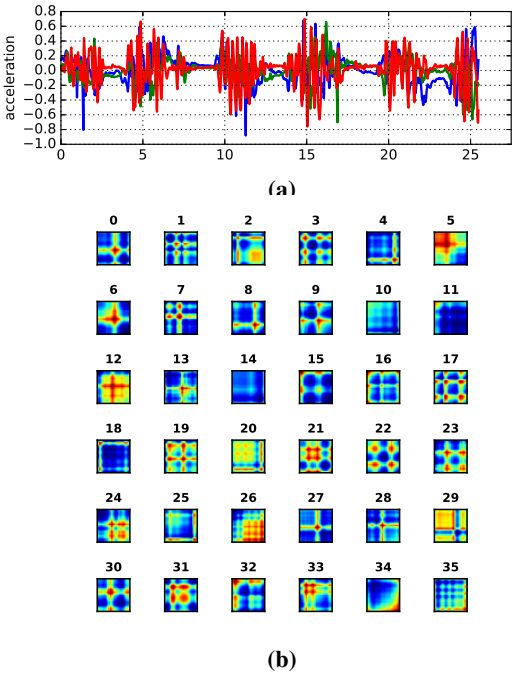


Figure 5.7: Example of the conversion of a time series into GASF images. (**a**) Three-axis accelerometer data (48.8 Hz) of about 30 seconds of real game play. (**b**) Thirty five GASF segments of a linear combination of the multidimensional time series. Each window comprises 0.65 seconds with 32 samples.

5.3.4 Technical Approach

This section describes our approach while emphasizing the differences and similarities with respect to existing methods. We take inspiration from approaches like [95, 117, 118] and aim at modeling game sessions as "documents", where player's acceleration data are considered as "words". We use LDA [13] to categorize the player's motion profile as a composition or mixture of game play motion types, the latter being analogous to the "topics" in the general application of LDA for document modeling.

Differently from [95], our input word atoms are not discrete. Instead, they are continuous (acceleration data) and also are not attached to any explicit meaning. Take, for instance, the natural interpretability of joystick input signals like "left" and "right", which convey their respective meaning when controlling a game character, and compare it with an acceleration signal that carries no obvious absolute meaning.

The adopted method for selecting the input words is based on sliding windows (just as in [95]), without overlap. Considering each game session

as a document, the windowed segments play the role of words. Since the space is continuous, a procedure of discretization and dictionary learning for LDA had to be performed. For this, we clustered the segments and used the so-obtained cluster centroids as dictionary words.

We have used GASF images instead of GADF because the mapping functions of the rescaled time series in $[0, 1]$ are bijections, which allows for precise reconstruction of the original time series [118]. The reconstruction takes advantage of the the main diagonal of GASF, i.e., $\{G_{ii}\} = \{\cos 2\phi_i\}$, and is performed as reported in Equation (5.8) [118]:

$$\cos(\phi) = \sqrt{\frac{G_{ii} + 1}{2}} \qquad \phi \in [0, \frac{\pi}{2}] \tag{5.8}$$

An example of signal decomposition into GASF images is shown in Figure 5.7. Given recent advances in deep learning and pattern recognition for images, we take the route of [117] and exploit the applicability of autoencoders for extracting unsupervised features from GASF images (see Section 5.3.2). This is relevant since the process of feature engineering is time-consuming and laborious. Moreover, unsupervised feature extraction from such methods has been proven to work well [118, 119]. Sparse encoding via linear-algebra methods for dictionary learning is also a possibility. However, in our experiments, it turned out to be not as efficient as autoencoders. We present details of our experimental activity in the next section. Our methodology is summarized in Figure **??**.

5.3.5 Experimental Results

In this section, the exploration of an undercomplete representation of the GASF images using a simple autoencoder architecture is firstly reported, then we discuss how we transform the extracted features into input words for the LDA model. Finally, we show the results when confronting the model output with human judgment. All experimental code used is available at the link in supplementary materials.

Feature Extraction

The first question we need to answer concerns the size of the sliding window. This parameter regards the definition of the GASF images used as building blocks for the dictionary learning. The window size should be large enough to capture motion patterns relevant for the player characterization, but not so large to confound their interpretation. We tried different window sizes, and

in the end, we followed [70] and decided to consider half a second of data per window without overlap, which allows for a fine-grained data representation.

We applied no preprocessing to the signal, since we were interested in observing how an autoencoder would perform using the unfiltered input. The expectation was that it would naturally compensate for the differences in the images, decisively choosing to capture important features (the main signal characteristic) and disregarding non-representative ones (the small noise associated). We defined an architecture composed by dropout and dense layers where only hyperbolic tangent activation functions where used, except in the last layer, where a linear activation function was used instead. The best performing architecture was composed by the following layers: dropout (0.1 drop fraction), dense (size = 256), dropout (0.2) and dense (size = 64). The same layers (in reversed order) were used for the decoder part.

A window of half a second corresponded to 32 samples given our accelerometer average frequency of 48.8 Hz. As a consequence, we obtain GASF images (without using PAA) with a total of 1024 pixels. Using the autoencoder, we manage to reduce the representation down to a 64-dimensional input, resulting in the reconstruction presented in Figure 5.8. The choice of a 64-dimensional representation is empirical, meaning that we were in principle searching for an undercomplete representation of the data and, upon testing with different configurations, we have chosen the one with the lowest reconstruction error. As it turned out, a 64-dimensional vector representation was the best result, given our dataset.

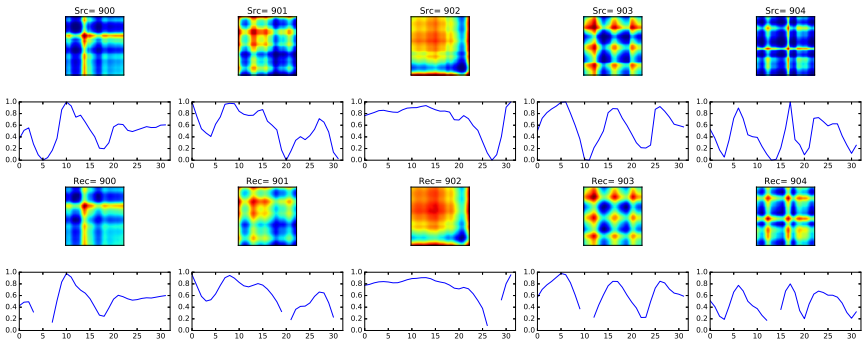


Figure 5.8: Autoencoder reconstruction for five input segments corresponding to half a second of data (linear combination of the acceleration axis). (Top line) Input GASF images. (Second line) Original time series for each GASF input. (Third line) The autoencoder reconstruction for the input images. (Forth line) Time series from the reconstructed GASF images.

It is possible to notice from the reconstruction result that, despite such a

large compression (from 1024 to only 64 values), the input image structures are captured. This can also be seen in the reconstructed time series. A closer look reveals that the autoencoder practically smoothed the time series while preserving the overall shape, this being a nice property that can help to prevent overfitting.

From Figure 5.8, we can also see that the obtained time series reconstruction presents some holes. This seems to be caused by the information loss in the reconstruction and demands further investigation. However, in our experimentation, it did not seem to affect performance.

Dictionary Learning and Input Word Definition

Latent Dirichlet allocation works by considering the relationship among word tokens with respect to a collection of documents. Therefore, in order to use the method, we needed to come up with a way to encode the taxonomy of GASF images into a discrete set of fixed tokens describing the motion primitives appearing in each game session. We opted to follow [76] and encode the features extracted from the GASF using the autoencoders as visual words, composing the game session document.

After getting the 64-dimensional vector representation for each GASF image (see Section 5.3.2), we cluster them into V groups using the K-means algorithm. In this way, we define a dictionary of size V, with each word corresponding to a cluster centroid. Each game session is then translated into a collection of V possible clusters, which can then be fed into the LDA algorithm.

From text mining and information retrieval, we know that having documents with different sizes can be a potential issue. This is because given the bag of words representation used in LDA, a small document may be included as a subset in a larger one. Additionally, it might be a problem to evaluate longer documents in our scenario, since, at least from a human perspective, it is not simple to define the overall behavior.

In text mining, this issue is handled by imposing a weight based on the length of the document; in our case, instead, we decided to split each session into smaller segments of 15 s each. In this setting, the evaluation is more consistent, and it is possible that the behavior of a player will be maintained within segments. We decided this length for each document both for convenience with respect to the window length and from the assumption that in each fraction of 15 s, the player can show a consistent behavior, which can be somehow easily identifiable. In the next section, we present the results achieved with the approach we just presented.

5.3.6 Validation

Our ultimate objective is to represent the player motion style by a mixture of topic proportions following the LDA framework. Unfortunately, the method by itself demands the programmer to initialize the desired number of topics. Thus, we had to perform a search for the optimal value given our dataset.

By performing grid search, we selected a range of values for the two main hyperparameters, namely dictionary size, for the input word definition, and number of topics. We would like to keep the number of topics low, preferably less than 10, since by observing our dataset, it is not possible to distinguish many groups of different motion styles; this suggests that a coarse-grained definition would be better. Choosing the number of topics to be at most 10, as in [95], the evaluation of the uncovered topics remains manageable, while maintaining a certain degree of freedom for the existing motion diversity.

The distribution of cluster centroids (words) given topics (our player motion types) is shown in Figure 5.9. Each line represents a topic, while each column represents a word. As one can see, there are no topics represented by the same distribution over the same word. Figure 5.9 shows how much each word is important for the specific topic.

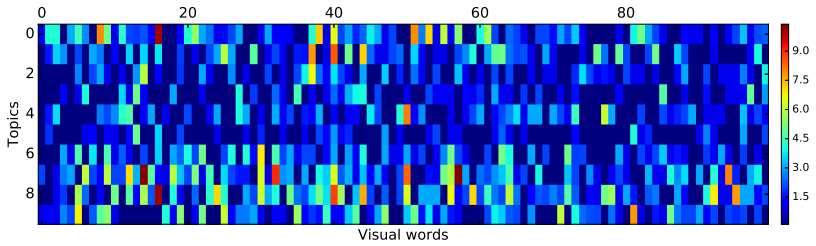


Figure 5.9: Schematic representation of the game: each line represents a topic, while each column represents a word that is representative of the specific topic.

In order to validate the discovered topics, we use human judgment. We invited volunteers to watch a set of pairs of recorded game segments (15 s each) and point out the similarities between players in their motivation as perceived by their motion. Subjects had to judge how similar, on a scale from 0–5, the movements of two players were. With this, we generated a similarity matrix to be used as the ground truth and main guide for deciding which hyper-parameter value to use and to check whether the proposed method could achieve reasonable results.

The similarity matrix was composed of 406 independent evaluations from six different human subjects. The generated evaluation matrix was then com-

pared with a cosine similarity matrix generated after training the LDA in our dataset. The similarity, in this case, is based on the mixture proportions among the discovered topics.

We used this information to define the number of topics and the number of words. The best set of parameters is the one that presents a higher similarity compared to the one expressing human judgment, thus the smallest Mean Squared Error (MSE). The graph in Figure 5.10 shows the results obtained varying the size of the dictionary (number of cluster centroids) and the number or topics. From it, we have decided to pick 100 as the number of visual words and 10 for the topics present in our dataset.

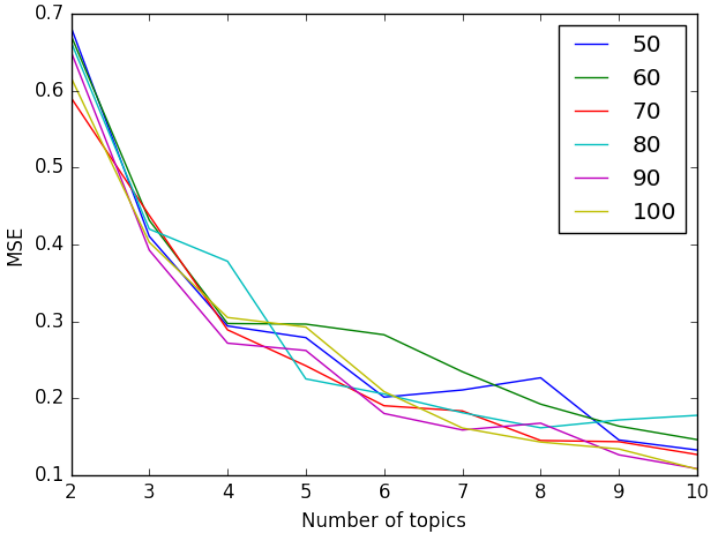


Figure 5.10: The figure shows the hyperparameter tuning results. The Mean Squared Error (MSE) is estimated with respect to the similarity expressed in the matrix containing the human judgment. The different lines represent the MSE obtained for different dictionary sizes and number of topics.

The observed similarity results were verified by visual inspection of video logs from estimated similar players; that is, by checking how reported mixture proportions actually look similar in video. However, we state that much work should still be performed towards easily and reliably assessing model correctness and selection. This also relates to the possibility of confidently giving names to discovered motion types, very much in the way as is done in classic LDA for text categorization.

Another aspect that demands further investigation is how to avoid the decrease in accuracy when considering variable-sized game sessions. As men-

tioned before, game sessions with different sizes may artificially increase similarity in case one is a subset of another. The solution to this problem provides only a baseline, and much more sophisticated approaches to the problem should be devised.

We believe that a potential way for improvements in interpretability and accuracy is that of removing the vector quantization in defining each topic. It would be interesting to remain in the continuous domain defined by our input data representation (GASF features), instead of relying on a fixed predefined vocabulary set used for training the LDA. Such vectorization is known to lead to information loss [39]. In avoiding such an issue, one possibility may be replacing the topic multinomial distribution by a mixture of Gaussians [39].

5.3.7 Discussion

In this work, we introduced an approach to identify the player motion styles from measurements obtained by the use of a single three-axial accelerometer sensor. Such a signal is then transformed using the Gramian angular field in an image that represents its content. The reason for such representation stems from recent publications aiming at using deep learning and state-of-the-art image processing techniques for achieving better classification results. Additionally, using the autoencoder for representing the obtained GASFs reduced the workload necessary for hand-crafting features, as often done when using machine learning classification/clustering methods with time series data.

By applying latent Dirichlet allocation to characterize the motion behavior of different players, we were interested in getting a description that would foster the development of self-adjusting PIRGs agents capable of improving/sustaining engagement. In this, we had shown that the model is able to capture the overall human judgment by matching reported similarities.

This is the first work, to our knowledge, that tries to model the motion behavior of the player using this type of signal in conjunction with LDA in a robogame scenario. We have demonstrated that the proposed approach is able to cluster and represent different behaviors learned from data. We have evidence that our methodology can achieve better performances than that of relying solely on the measurements of the accelerometer as input for the dictionary learning step. We are still working to provide a fair comparison between the two approaches.

As future work, we want to test our approach with a series of different input signals in order to assess its capability. One alternative could be the

incorporation of proximity information in order to capture how proximity patterns relate to a player game play style. Moreover, we are interested in measuring the player's effort expenditure in terms of difficulty of motion given a difficult setting for the robot.

In our method, we also wish to incorporate time dependency. We believe such information would help to describe variations in playing style during play, given the opportunity to detect change points in styles and favor game (robot behavior) personalization. This also may help to avoid the rubber-band effect, a system instability concern often present when designing auto-adjusting game systems.

Another means for improvements, and one that we are currently working on, is that of removing the vector quantization step necessary for the definition of the multinomial distribution of words that defines each topic. We believe that it would be interesting to remain in the continuous domain defined by our input data, instead of relying on a fixed predefined set of tokens, i.e., the vocabulary set used for training the LDA. Our current experimentation goes along the lines of testing how a Gaussian version of LDA would work in our domain and whether that could give better results. Additionally, comparing our results with a continuous version of LDA would help to answer questions regarding stop word filtering, such as: To what extent removing certain common segments would improve or worsen accuracy?

Finally, we are working to redesign the experiments in order to allow for collecting user survey data, which comprise a valuable resource regarding player style and engagement levels. This is all in order to increase our understanding in interpreting the obtained results. We plan to collect self-reported engagement levels when comparing with different robot behavior configuration while allowing for adaptation. Regarding such adaptation, we currently are investigating how to relate the robot game's difficult settings (such as velocity, tower LED control rules, among others) and the mixture proportion of motion types discovered by our reported method. One alternative is to design a utility function that would describe what is the pay-off in facing a certain player given a certain difficult setting, therefore turning the adaptation problem into a full on-line optimization procedure where we try to match player motion types with robot behavior implied by difficulty settings.

Chapter 6

Adjusting robot playing behavior through latent skill modeling for increasing player satisfaction

Overleaf link: https://v1.overleaf.com/23518282qswvwgkvmcnr

Chapter 7

Using social interaction analysis and deception for entertainment support in a physically interactive robogame

A game is an activity that one engages in for amusement. In the quest for new playing experiences, robots are being developed to take part in interactive games with humans. Making believable, playing robots able to keep human players engaged and satisfied with the playing experience is the main challenge for this kind of applications. With this research aim, this paper investigates the quality of interaction between a human player and a mobile robot in a physical, interactive robogame. We apply previous development in social interaction analysis and, in particular, we focus on the applicability of deception theory as a mean to support engagement. By analyzing the social situation between two players (human and robot), identifying the need for deception, and acting upon it, we aim at decreasing predictability while increasing engagement and amusement, which are related to the perception of an opponent smart enough to compete at the right level. Experiments have been conducted on a population of 93 people which have been asked to play the game and answer a questionnaire. The majority of the participants enjoyed the game and perceived the robot as a rational agent that aims to win the game. Deception was also perceived by most of the players.

7.1 Introduction

As technology grows, the amount of devices that are part of our daily life rises as well. Since childhood, the new generations are learning to interact with technology while playing with new games implemented in embedded systems. This phenomenon can be seen on a large scale with robot games. New researches are made on robots developed to interact with human players in a game environment. Games not only make people have fun, but they are one of the most powerful tools for socialization, and cognitive development [18, 74, 112]. While playing competitive games, a person is encouraged to learn the opponent's behavior for various reasons; for example, having a good knowledge of the opponent's behavior is essential for building a winning strategy, and this plays an important role in creating amusement.

The main issues that should be faced in this field concern the robot's intelligence as perceived by the human player: low or too high levels may induce the player to leave because the robot is not considered as a valid competitor, and this lowers the entertainment level of the game.

We are focusing on a challenging type of games, where the players are involved in a physical, quite demanding activity. This type of games has been introduced as PIRG [62]). In PIRGs, it is important to model the player, by using the only available data that could describe well enough the movements of the player.

In different and not physical games, deception has been recognized as a way to increase attributions of mental states to the robot[91].

Our research aims at analyzing human-robot interaction in games. More precisely, we try to improve the study on conflicting games, where the use of intelligent strategies by the players is the main path for winning the game. Deception is one of these strategies and it is not trivial communicating it to a human being using a robot. Our work gives a baseline for detecting and communicating deception in PIRG applications.

To the best of our knowledge, this is the first research work that applies deception theory to support engagement in a PIRG with mobile robotic platforms. The paper is organized as follows. Section 7.2 introduces related works, section 7.3 explains in details the game, in section 7.4 the method for detecting the need of deception and how to translate it into motion activity are described, experiments and validation of the proposed method are presented in section 7.5, finally, discussions and conclusions are discussed in sections 7.6 and 7.7.

7.2 Related Works

7.2.1 Deception and robots

Interpersonal situations and interaction between individuals have been studied in sociology. Interdependence theory has been introduced in [46], where the authors studied the fact that people adjust their interactive behavior as consequence of their perception of a social situation. The adjustment is dictated by rewards and costs that every choice in a determinate moment in time can lead to. Every situation, then, can be expressed as a multiple choice of rewards in a matrix form, called outcome matrix. This concept can be seen as an equivalent game theory's normal form game. Furthermore, the authors introduced a four dimensional space where mapping of social situation occur using the constructed outcome matrices. Interdependence, correspondence, control and symmetry are the four dimensions that define that space. The first two respectively represent how much individuals influence each other reward and if outcomes of the individuals are consistent. An algorithm for analyzing social situations for robot's interactive behavior that uses the interdependency theory proposed in [46], is described in [115]. Our proposed method takes inspiration as well from [46], but in opposition to [115], revisits the interdependence space (only 2D space interdependence-correspondence) and the way of mapping the outcome matrices as it will be discussed in Section 7.4. In [116] an algorithm for detecting when and whether a robot should deceive has been proposed; the decision is based on the mapping of the outcome matrix to the interdependence-correspondence space. This algorithm has been used in [114], where the analysis of the social situation is performed to provide robots with the capacity of determining if deception is needed.

The works of [46, 114–116] have laid the foundation for many research activities in social aspects involving humans and robots, in particular on the aspect of *deception*. From the taxonomy in [91], it is possible to classify deception according to the interaction object. Objectively, there are two types of deception, human-robot and nonhuman-robot. An example of nonhuman-robot deception is [92], where a squirrel-robot was capable of deceiving another one for gaining more food. On the other hand, deceiving a human is not trivial. Studies about a robot able to deceive a human have been reported in [104], where the authors discuss whether the feeling of being deceived by a robot would be an indicator that the human treats the robot as an intentional entity. Further experiments about the engagement increasing in a game due to a robot able to deceive, have been conducted in [93]. This cheating 'rock-

paper-scissors' interactive robot takes advantage of the deception for its own benefit. Differently from that robot, the robot presented in [113] has been programmed for following a multi-player, reaction-time and conflict game, where its main role is to establish who is winning. The robot does not play, but it sometimes changes the wons in order to make the game more social engaging when the players discovered the cheating behavior of the robot.

The most important part in a deceiving algorithm is to properly transmit to the deceived the false communication. This is a delicate part because the deceived should not understand it is under deception. In [27] the authors propose a way to study the communication of false information (deception) using a robotic arm. On the work, the author reinforce the applicability of robot deception in making games against robot more engaging. All the paper is concentrated on robot deception in goal-directed motion, by learning deceptive trajectories in which the robot is concealing its actual goal. The present work is similar to [27] in the sense that it also explores goal-directed motion by following trajectories, but differ from that work on using a full mobile robot in a real game scenario.

7.2.2 Physically Interactive Robogames

Robot toys are presently the largest amount of robots delivered in our homes, at a rate of millions every year. In almost all the cases, the interaction is limited to stimulus-response reactions, but the introduction of low-cost sensors and computer power are making it possible to introduce a richer interaction and games specifically designed for mobile robots. In a PIRG, physical autonomous (often mobile) agents are actively engaged in a game that creates some sort of interaction, either competitive or cooperative, between humans and robots, moving in the physical world. Robogames will be one of the next robotic products for the mass technological market, thus demanding a large exploration of new methodologies and applications, especially for what concerns methods for enabling high autonomy, intelligence, and adaptive behavior, in order to respond to demands in engagement support like the ones expressed in [122, 124, 126]. Being able to evaluate how people play is crucial for an adaptive game. In the virtual game industry, several studies have been published, most of them using artificial intelligence and machine learning algorithms. For instance, [26] reports about emergent self-organizing maps for grouping types of players.

There have been attempts for presenting robots as toys over the years, where, in most cases, the robot acts more or less like a mobile pet (e.g., [17, 32]). In these cases, interaction is often seen as limited to almost static

positions, not exploiting rich movement, nor high level of autonomy; the credibility of these toys to lively engage people, such as kids, are said to be constrained [15, 62].

Some PIRGs have been reported over the last few years. Jedi Trainer 3.0. [62] was a PIRG mimicking a situation of the first Star Wars saga movie "Episode IV - A New Hope": it was able to show some apparent adaptation to the player's playing style as well as some realism of the drone's behavior, which could be understood as some kind of rational behavior by the human player.

Queball [86] was a robotic ball proposed to engage autistic children in games that could have therapeutic goals.

Teo [16] is a huggable, mobile robot designed to play games with autistic children, and to provide the possibility to make free play as well as structuered play experiences.

RoboTower [70–72], where humans physically interact with an omnidirectional robot by trying to protect towers while avoiding letting the robot pushing them down. Despite being examples of successful applications, only [71] and [72] present an attempt to quantify the human behavior during gameplay.

An approach based on genetic algorithms for capturing and modeling individual entertainment is given in [124], where the main goal is to construct a player model, in this case a child playing a Playware game. "Playware is the use of intelligent technology to create the kind of leisure activity we normally label play" [58]. The system can predict the answers to a question asking which variants of the game are more or less "fun". In that work, the model is constructed from physiological signals measured during play.

In summary, pretty much all the related works agree that being able to label player's behavior in a game environment can enable the robotic agent to modify its interactions and playing-style in order to adapt to the skills of its human counterpart. This is a general statement shared with most human-robot interaction applications. On this work, we go further and investigate the introduction of deceptive behavior as another factor for supporting engagement.

7.3 The game environment

7.3.1 Playground

The playground is a rectangular area of 4 m $\times$ 4 m, where, on each corner, props representing towers are placed. Each tower is equipped with a button

and four LEDs that can be progressively turned on. Each tower LED requires the tower button to be pushed without interruption for 2.5 seconds in order to be turned on, meaning that a tower takes about 10 seconds to light up all four of its LEDs.

After all LEDs of a particular tower are turned on, it is considered as captured by the human player, and the robot will no longer consider it. The activity of turning on LEDs can be distributed in different moments, i.e., the players will not lose their progress if they leave the button before a tower is captured. Tower information such as: tower status (whether active, fallen, or captured) and button presses are transmitted to the robot via wireless communication.

7.3.2 Robotic agent

The robot is a holonomic platform, 70 cm high, 50 cm wide and 50 cm long. It is capable of navigating autonomously in the environment, using the ROS[1] framework. The robot localization is performed by using Monte Carlo localization algorithms through the AMCL[2] node, using laser range scans. For the management of sensor information we have implemented custom ROS nodes. By relying on its laser scanners, the robot can also perceive the human player during the game, so as to track her/him, and also to avoid hitting him/her while moving. Additionally, we also consider the player's distance to towers, extracted from the system coordinate transformations using the player's position and towers fixed reference frames in the robot's map.

7.3.3 Rules

In order to win, the human player must be able to secure all the existing towers without letting a single one be knocked down. If, at anytime, a tower falls (because of the robot or player), the game ends, and the human player loses.

Since the robotic player has holonomic kinematic properties, it is able to move across the entire playground just as the human can and it is only constrained by the fact that an already captured tower, or one whose button is currently being pushed by the player, cannot be teared down. The player can also block the robot path by staying on it. Notice that, while the player is trying to capture a given tower, the robot can try to tear down another one.

[1] http://www.ros.org
[2] http://wiki.ros.org/amcl.

7.4 Method

7.4.1 Detecting the need for deception

During the game, the robot continuously (re)calculates the payoff of attacking (moving closer to) a given tower. For doing that, the robot uses two arrays, $\overrightarrow{t_{robot}}$ and $\overrightarrow{t_{player}}$. The former quantifies the robot's own payoff and the later quantifies the player's payoff. Both payoff are defined by spatial relation. Considering the set of tower positions $\mathcal{T} = \{\tau_i\}_{i=1}^{N=4}$, the arrays are defined as in 7.1 and 7.2.

$$\overrightarrow{t_{robot}} = \begin{bmatrix} \delta(\tau_1, player) - \delta(\tau_1, robot) \\ \delta(\tau_2, player) - \delta(\tau_2, robot) \\ \delta(\tau_3, player) - \delta(\tau_3, robot) \\ \delta(\tau_4, player) - \delta(\tau_4, robot) \end{bmatrix} \tag{7.1}$$

$$\overrightarrow{t_{player}} = \begin{bmatrix} \frac{1}{\delta(\tau_1, robot)} + \frac{1}{\delta(\tau_1, player)} \\ \frac{1}{\delta(\tau_2, robot)} + \frac{1}{\delta(\tau_2, player)} \\ \frac{1}{\delta(\tau_3, robot)} + \frac{1}{\delta(\tau_3, player)} \\ \frac{1}{\delta(\tau_4, robot)} + \frac{1}{\delta(\tau_4, player)} \end{bmatrix} \tag{7.2}$$

Where $\delta(a, b)$ refers to the Euclidean distance between argument a and b. Choosing the target tower is done by detecting the maximum value in the arrays:

$$target_{robot} = \arg\max_i \overrightarrow{t_{robot}}^{(i)}, \tag{7.3}$$

, where i refers to the vector component index.

$$target_{player} = \arg\max_i \overrightarrow{t_{player}}^{(i)} \tag{7.4}$$

The robot calculates the target preferences for the player as well, in order to be able to predict what would be the opponent's move, as described later.

For recognizing the particular moment in time where deception may be needed, a set of outcome matrices that model the in-game interaction have been defined. Such matrices quantify the payoff, also referred to as utility, associated with each player's actions. As actions, the players can choose to move towards one of the four towers.

As the game is a conflicting game, the two players gain benefit in each other's loss. For this reason, the design of the outcome matrices has been made as described here below: the robot earns a higher payoff when itself and the opponent choose a different target. From a matrix point of view,

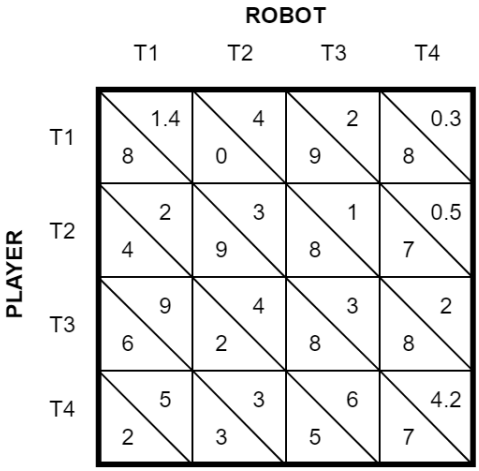


Figure 7.1: Example of outcome matrix. The columns represent the robot's payoff in attacking one of the four towers. The rows represent the player's one. Each element is a couple of values, namely the player's payoff and the robot's payoff.

this is seen as a lower payoff in the diagonal elements, because it represents when the two opponents choose the same action. On the other hand, the player's outcome matrix is built starting from the fact that the human player earns a higher payoff when his/her choice is the same as the robot's one. For this reason, the values of the diagonal elements must be greater than the non-diagonal ones.

The approach for calculating the robot's outcome matrix is the following:

- *Non-diagonal Elements:*

$$\gamma_r \cdot \frac{\overrightarrow{t_{robot}}^{(i)}}{\sum_i \overrightarrow{t_{robot}}^{(i)}} \cdot \frac{\delta(\tau_i, player)}{\sum \delta(\tau_i, player)}, \forall \tau_i \in \mathcal{T} \tag{7.5}$$

  , where $\overrightarrow{t_{robot}}^{(i)}$ refers to the component i on the vector t_{robot}.

- *Diagonal Elements:*

$$\delta(\tau_i, robot) \tag{7.6}$$

  Where γ_r is a tuning constant. In the robot's outcome matrix, γ_r has been used to weight the values of the non-diagonal values, equation (5), while in the player's matrix, another tuning constant has been used for the diagonal values, equation (8). Tuning the values of the outcome matrices is needed for highlighting the fact that the two players win

in two different situations. Giving the right value to γ is necessary for having a balanced values for the correspondence-interdependence value. The second term in (5) has been used for giving a weight to the payoff: the more an individual is interested in a given target (supposed that this is a consequence of rational / maximizer thinking), the higher the payoff is if the player can obtain it. The third and last term in (5) expresses how much advantage the player has on the other one.

Similarly, the player's outcome is described below:

- *Non-diagonal Elements:*
  $\forall \tau \in \mathcal{T}$

$$\frac{\overrightarrow{t_{player}}^{(i)}}{\sum_i \overrightarrow{t_{player}}^{(i)}} \cdot \frac{\delta(\tau_i, robot)}{\sum_i \delta(\tau_i, robot)}, \forall \tau_i \in \mathcal{T} \tag{7.7}$$

- *Diagonal Elements:*

$$\gamma_p \cdot \frac{1}{\delta(\tau_i, robot)} \tag{7.8}$$

From this, a player's utility is periodically (re)computed and then incorporated into the interdependence-correspondence space [114].

For our purposes, the *interdependence* dimension represents the correlation between the two players' outcome matrices, while *correspondence* one quantifies how much conflict exists between the two selected actions.

The two values are calculated keeping in mind three concepts: the variation of the robot's outcome matrix resulting from its own decisions; the variation of the player's outcome matrix resulting from the partner's decisions; the variance in outcome matrix resulting from both, joint, interactive decisions.

The calculations for interdependence are made as in algorithm 1, where $\alpha \in [-1; 0]$, and Δ expresses the maximum variation the player can obtain on the robot's outcome matrix. The variable *total* expresses the range used for normalization, and $n(\mathcal{T})$ the number of towers.

The procedure to calculate the correspondence value is defined in algorithm 2, where $\beta \in [0; 1]$. The procedure works by going through every action the robot and the player can take. For example, if the robot decides to take action 1 (go to tower 1), it temporally ignores all the other columns of the robot's outcome matrix and checks what are the indexes that maximize the robot's outcome and the player's one. It then calculates the difference between the outcomes using the two indexes (for both robot and player); a negative Δ means for the subject that the situation is a conflict (the values of

input : Robot's outcome matrix
output: Float value

for $i \in range(1, n(\mathcal{T}))$ **do**
  $\Delta_{outcomes} = |max_{outcome(:,i)} - min_{outcome(:,i)}|$
  $total = max_{outcome(:,i)} + min_{outcome(:,i)}$
  $\alpha \mathrel{+}= \frac{\overrightarrow{t_{robot}}^{(i)}}{\sum_i \overrightarrow{t_{robot}}^{(i)}} \cdot \frac{\Delta}{total}$
end

Algorithm 1: Interdependence algorithm

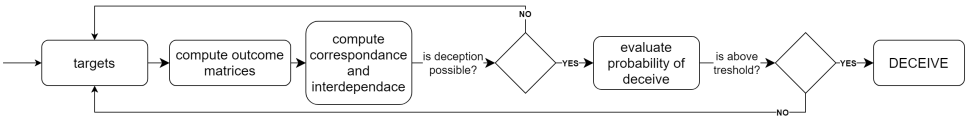


Figure 7.2: The procedure to select deception

the outcome matrix are positive). The variations are normalized and multiplied by each other and then multiplied by a factor that expresses how likely the action would be taken (if an action will not be chosen, β will have a lower impact). Similarly, the same procedure is done w.r.t. the player's actions.

In *Figure 7.3* the correspondence and interdependence space distributions are represented. By making a discretization of the playground into a grid and placing the robot at (4,5), while varying the player's position on every other coordinate, it is possible to notice that the diagonal that leads to the towers has a higher value of dependency and conflict.

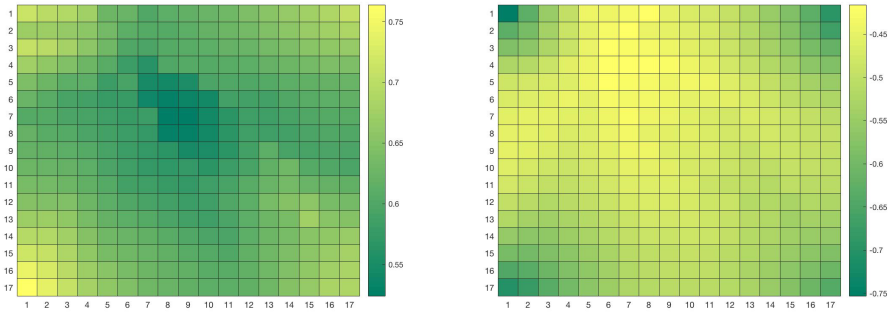
The mapping of the situation to the space shows when deception may be needed. Once an extremely dependent and conflicting situation is detected, deception is triggered.

Once deception is warranted, the algorithm calculates the fake target to be communicated to the player. The fake target is chosen in the following steps: after the real target is established, the robot calculates two actions that would give the robot the highest rewards and, finally, it calculates which of the two maximizes the player's payoff. This last step expresses the incentive for the player to approach the fake target. The algorithm is detailed in Figure 7.2.

input : Outcome matrices
output: Float value

Avg(outcome ∀ Robot's action)
Avg(outcome ∀ Player's action)
for $i \in range(1, n(\mathcal{T}))$ **do**
    amr = tower index that maximizes robot's outcome
    amp = tower index that maximizes player's outcome
    $\Delta_{robot} = outcome_{robot}(amr, i) - outcome_{robot}(amp, i)$
    $\Delta_{player} = outcome_{player}(amr, i) - outcome_{player}(amp, i)$
    $total_{robot} = outcome_{robot}(amr, i) + outcome_{robot}(amp, i)$
    $total_{player} = outcome_{player}(amr, i) + outcome_{player}(amp, i)$
    $\beta \mathrel{+}= \frac{1}{2} \cdot \frac{Avg_{robot}}{\sum Avg_{robot}} \cdot \frac{\Delta_{robot}}{total_{robot}} \cdot \frac{\Delta_{player}}{total_{player}}$
end
for $i \in range(1, n(\mathcal{T}))$ **do**
    amr = tower index that maximizes robot's outcome
    amp = tower index that maximizes player's outcome
    $\Delta_{robot} = outcome_{robot}(i, amr) - outcome_{robot}(i, amp)$
    $\Delta_{player} = outcome_{player}(i, amr) - outcome_{player}(i, amp)$
    $total_{robot} = outcome_{robot}(i, amr) + outcome_{robot}(i, amp)$
    $total_{player} = outcome_{player}(i, amr) + outcome_{player}(i, amp)$
    $\beta \mathrel{+}= \frac{1}{2} \cdot \frac{Avg_{player}}{\sum Avg_{player}} \cdot \frac{\Delta_{robot}}{total_{robot}} \cdot \frac{\Delta_{player}}{total_{player}}$
end

Algorithm 2: Correspondence algorithm



(a) **(b)**

Figure 7.3: Space distribution of the correspondence and interdependence when the robot is placed in (4,5). (a) Interdependence space distribution; darker cells correspond to lower interdependence. (b) Correspondence space distribution; brighter cells indicate lower correspondence.

7.4.2 Communicating false goals

Having presented how to trigger deception, in this section we describe means of effectively using deceptive behavior by communicating false goals, i.e., false attempts to knock down towers. We begin by describing the two methods used.

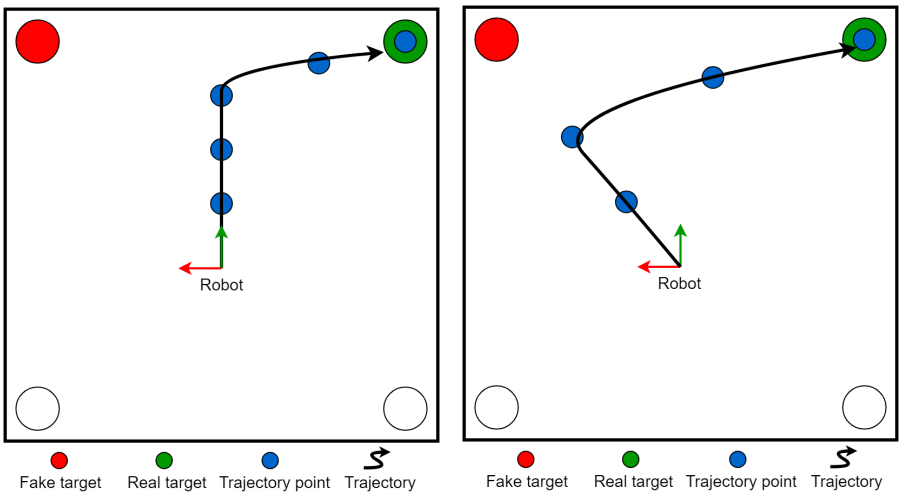
Static trajectory approach

In this first method the trajectories are established a priori. It means that once the navigation node receives the fake and real target information, it calculates, based on the actual position of the robot, the series of points the robot needs to follow in order to communicate the deception. The algorithm can choose between two different types of trajectories. The first one is activated when the robot is halfway through the fixed frame of localization (map, on the ROS jargon). The robot will then move forward without letting the player know which tower is going to be hit, leaving 50% of chance of guessing. Only when "close", within a predefined threshold, to the midpoint of a virtual line connecting the two towers (true and fake target, respectively), the robot changes trajectory towards the true target.

The second type of deception will try to communicate from the beginning the fake target in order to let the player run and stay to a particular tower, and in this way take spatial advantage when trying to knock down the real target. A pictorially description of such trajectories is shown in Figure 7.4.

The trajectories are implemented by deciding a series of points to be followed. Those points are spatially distributed along the desired trajectory (Figure 7.4).

In the present proposal, deception is considered as a complete 'path' that starts when the robot receives the deception information (true and fake tower) and lasts until the robot hits the tower. Since no look-ahead for determining the player position in the future has been considered, it can happen that the player understands the deception and/or the player is really reactive and blocks the robot, so that it would be difficult to finish the deception procedure coherently. For this reason, the algorithm calculates the time required for successfully making the deception and, every time something goes wrong (for example, the player blocks the robot), the estimated time expires and the deception procedure is aborted. Once it happens, the normal procedure starts again to compute a regular target, until a new need for deception is detected.



(a) Moving forward and then changing the direction.

(b) Moving toward the false target and then changing direction.

Figure 7.4: The two types of deception when using the static trajectory approach.

Dynamic steering behavior approach

The second approach we propose in order to implement a deceiving trajectory is based on steering behaviors [80]. The paper by Reynolds proposes a force-based approach to guide an actor in a life-like and improvisational manner. Given a target, it will generate a force, either attractive or repulsive, based on its position with respect to the robot: by applying this force on the robot, it will be driven either towards or away from the target following a smooth path.

Being our robot holonomic, it has been represented as a point mass to calculate the results of the application of the forces generated by the steering behavior. This approach gives us the possibility to dynamically change the robot response to forces by changing the kinematic properties of our representation during the calculation process.

Instead of planning the complete trajectory, only the point where we want to change the motion parameters of the robot and reveal the true target is computed and set as a temporary target. The steering behavior framework drives the robot to reach this target following a slightly different trajectory each time based on its initial velocity and position.

In order to follow a deceptive trajectory as previously described, the dexterity of the robot is dynamically increased when revealing its real target: once the temporary target is reached the virtual mass of the robot is lowered

and a higher force is allowed to be applied to it, along with a slight increase on its maximum velocity: then the target to be reached is set to the real target. This procedure generates a sharp turn and an acceleration of the robot towards the real goal. A visualization of the parameter change effect can be seen in Figure 7.5.

Given this framework, we wanted to investigate whether such change of motion pattern helps the player to realize that the intention carried out by the robot had been to deceive and whether diversity of trajectories increases the appeal of the game by making the robot movements less predictable.

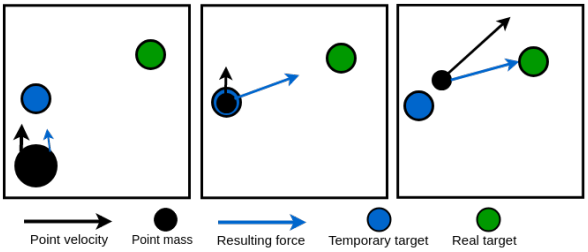


Figure 7.5: Update in vehicle parameters and target will produce an increment in robot velocity as well as a smooth bend in the real target direction: on the left we're approaching the temporary target; on the center we change the parameters as we reach the temporary target; on the right we move towards the real target with new parameters

7.5 Validation

In this section, we describe the acceptability of our strategy within the mentioned interactive scenario. We first present the experimental setup, stating the hypothesis and the use of a post-match survey for evaluating them. The analysis and related discussion follow next.

7.5.1 Experimental setup

Hypotheses

H1 The subjects enjoy the game.

H2 The subjects consider the robot as a rational agent, aiming at winning since it is participating to a competitive game.

H3 The two trajectory approaches are both able to create a recognizable level of deception.

H4 The dynamic steering approach appeals more than the static one.

Factors

During our experiments we have kept the parameters that control the general game difficulty discrete such as: maximum speed and maximum acceleration. It was part of our project decision to limit the effect of such variables in this study, while, at the same time, improve safety (since a possible collision against a fast robot is dangerous) and reduce the risk of mechanical problems, since frequent, abrupt acceleration/deceleration might damage the wheel-motor joints.

Speed and acceleration values define two different levels: easy (speed = $0.5m/sec$ and acceleration = $0.1m/sec^2$) and medium (speed = $0.7m/sec$ and acceleration = $0.5m/sec^2$).

However, during experimentation we have noticed that the difficulty level set to easy turned out to be too slow. Therefore, in order to maintain consistency in the analysis while also improving fun we have decided to keep the difficulty fixed at the medium level for all the subjects.

In summary, the only varying factors were the approaches for communicating false goals, i.e., the static trajectory and dynamic steering behavior.

7.5.2 Post-Match Survey

After every match, a questionnaire was administered to each player. It included few questions about the subject (including age and gender) and eleven questions about the game. In this paper, we present results about only four of them, relevant to evaluate the specific hypotheses.

7.5.3 Participants

We recruited 91 participants among the visitors of a science fair. Most of the experiments were conducted with children, spanning from 5 to 15 years old, while some adults from 16 to 54 years old also accepted to play the game. The distribution of subjects is reported in table 7.1. Data collection was performed with subjects being sampled on different times of the day in the attempt to randomize the subjects as best as possible.

7.5.4 Results

All the subjects either agreed or strongly agreed with the statement: "I have enjoyed the game" (median = "strongly agree").

Table 7.1: Gender distribution during the experiments with the static and dynamic trajectory approach.

Trajectory	Subject Age	Gender		*Total*
		Male	*Female*	
Static	Children (< 16)	29	19	48
	Adults (≥ 16)	6	7	13
Dynamic	Children (< 16)	12	16	28
	Adults (≥ 16)	2	0	2

Static deceiving strategy

The distribution of the answers to the question: "Was the game too short?" aimed at evaluating the effort to participate to the game, is reported in Figures 7.6a and 7.6b

The distribution of the agreement with the statement: "The robot aimed at winning", aimed at evaluating the perception of playing against a rational agent, is reported in Figures 7.6c and 7.6d.

The distribution of the agreement with the statement: "The robot was deceiving", aimed at the perception of the deceiving strategy, is reported in Figures 7.6e and 7.6f.

Dynamic deceiving strategy

The number of adult subjects playing with the dynamic deception strategy active was too small to be considered, so we report in this section only data concerning young subjects.

The distribution of the agreement to the statement: "The game was too short" aimed at evaluating the effort to participate to the game, is reported in Figure 7.7a.

The distribution of the agreement with the statement: "The robot aimed at winning", aimed at evaluating the perception of playing against a rational agent, is reported in Figure 7.7b.

The distribution of the agreement with the statement: "The robot was deceiving",aimed at the perception of the deceiving strategy, is reported in Figure 7.7c.

7.6 Discussion

We designed a game that obtained a good acceptance by all the participants, most of which queued for participating and were pleased after the game. From the values of the agreement to the statement "I have enjoyed the game", none of which was lower than "Agree", it can be said that hypothesis H1 is satisfied.

The robot was generally perceived as a rational agent, aiming at winning the game. So we can consider hypothesis H2 as satisfied. This is a good result since the implemented strategies purposefully reduced the capabilities of the robot to adapt its behavior to match those of the players.

Furthermore, deception was perceived by most of the players, with a slightly higher number of subjects among the children strongly agreeing on the fact that the robot was actually deceiving. Hypothesis H3 can also be considered as satisfied.

No significant difference can be detected between the two algorithms in the children population, where it is possible to compare the data, so hypothesis H4 has to be rejected for this sample. Possibly, the difference between the trajectories has been washed out by the quick dynamics of the game, and the high requirement of attention to many different aspects such as the robot movement and position, the position w.r.t. the towers, and the LEDs on the towers.

7.6.1 Limitations

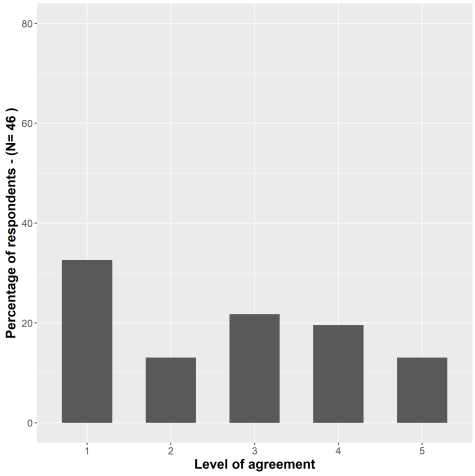
Among the limiting conditions of our work is the lack of a player model that can be used to tailor the deceptive motion on line. For instance, Machine Learning algorithms can be used to model the player's actions and try to maximize the level of surprise [9] regarding tower attack. Another limitation comes from constraints that prevent quick changes in motion direction, imposed for safety reasons, which, in turn, may produce negative impacts on the perception of the deceptive movements.

7.7 Conclusion and Future Works

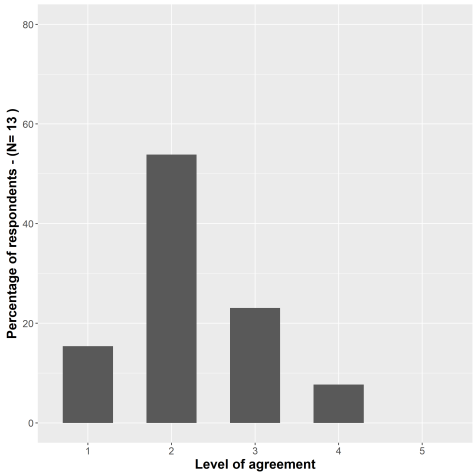
We have implemented a lively, enjoyable, physical robogame, where the robot is perceived as a rational agent aiming at winning the game, even if its capabilities are strongly reduced w.r.t. its possible top performance. This is an example where effective human-robot interaction is obtained by implementing robots that are not behaving optimally, in the sense of minimizing some performance parameter, but they match the expectation of interacting

people to attribute rationality to the robot companion, which is one of the aims of human-robot interaction. This result has been obtained also because of the two different deception algorithms that we have implemented, together with the system that triggers them by analyzing the game situation. These implemented deceptive trajectories are not optimal to win, but obtain the desired effect on the player.

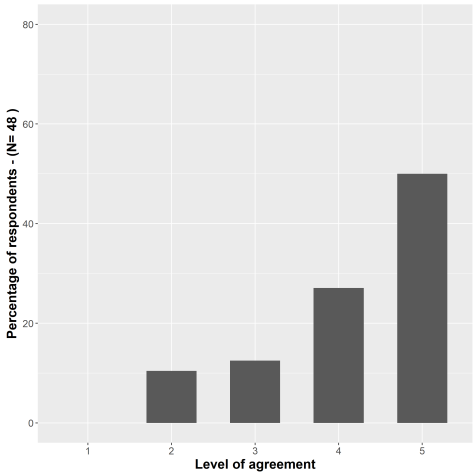
In the next future, we plan to collect sufficient data to perform sound statistical analyses, and to finalize a system to adapt the behavior and strategy of the robot to the perceived behavior, ability and strategies of the human players, in real time, so to optimize their satisfaction and enjoyment.



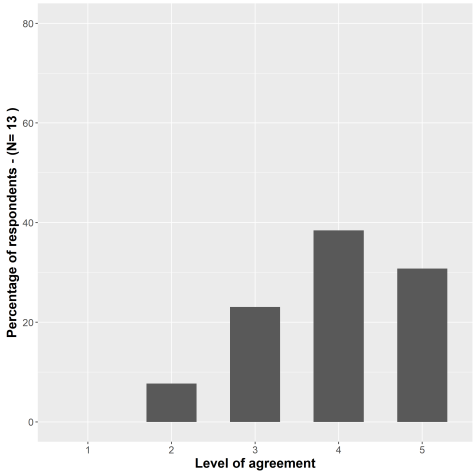
(a) Too short - children.



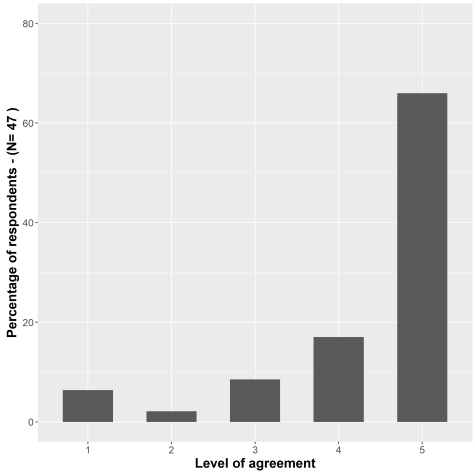
(b) Too short - adults.



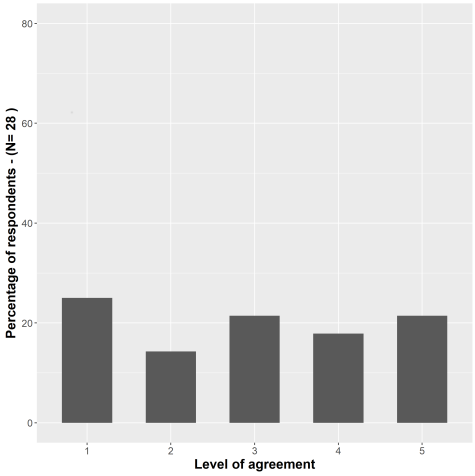
(c) Robot wants win - children.



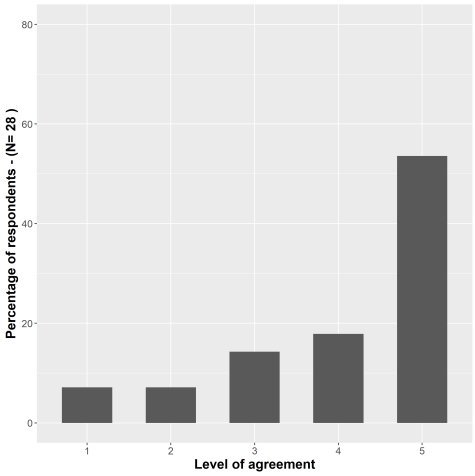
(d) Robot wants win - adults.



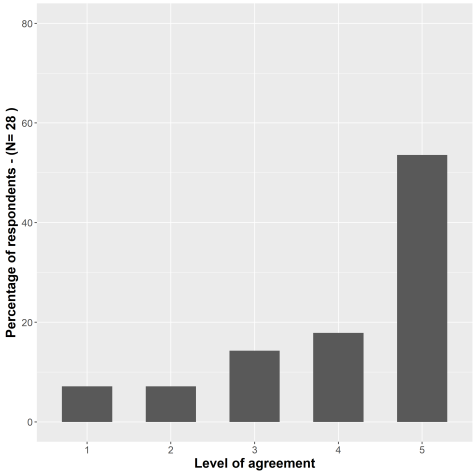
(e) Deception - children.



(a) Too short.



(b) Robot wants win.



(c) Deception.

Figure 7.7: "The game was too short" (subfig a)."The robot aimed at winning" (subfig b). "The robot was deceiving" (subfig c) 1=Fully disagree 2=Disagree 3=Indifferent 4=Agree

Chapter 8

Key issues in designing engaging physically interactive robogames

Chapter 9

Future work and conclusion