Interactive Virtual Cinematography

Paolo Burelli
Center For Computer Games Research
IT University of Copenhagen

A thesis submitted for the degree of
Doctor of Philosophy

April 2012

To Diego and Rosalba

Abstract

A virtual camera represents the point-of-view of the player through which she perceives the game world and gets feedback on her actions. Thus, the virtual camera plays a vital role in 3D computer games and affects player experience and enjoyability in games. *Interactive virtual cinematography* is the process of visualising the content of a virtual environment by positioning and animating the virtual camera in the context of a interactive applications such as a computer game.

Camera placement and animation in games are usually directly controlled by the player or statically predefined by designers. Direct control of the camera by the player increases the complexity of the interaction and reduces the designer's control on game storytelling. A completely designer-driven camera releases the player from the burden of controlling the point of view, but might generate undesired camera behaviours. Moreover, if the content of the game is procedurally generated, the designer might not have the necessary information to define a priori the camera positions and movements.

Automatic camera control aims to define an abstraction layer that permits to control the camera using high-level and environment-independent rules. The camera controller should dynamically and efficiently translate these rules into camera positions and movements before (or while) the player plays the game. Automatically controlling the camera in virtual 3D dynamic environments is an open research problem and a challenging task. From an optimisation perspective it is a relatively low dimensional problem (i.e. it has a minimum of 5 dimensions) but the complexity of the objective function evaluation combined with the strict time constraints make the problem computationally complex. Moreover the multi-objective nature of the typical camera objective function, introduces problems such as constraints conflicts, over-constraining or under-constraining.

An hypothetical optimal automatic camera control system should provide the right tool to allow designers to place cameras effectively in dynamic and unpredictable environments. However, there is still a limit in this approach: to

bridge the gap between automatic and manual cameras the camera objective should be influenceable by the player. In our view, the camera control system should be able to learn camera preferences from the user and adapt the camera setting to improve the player experience. Therefore, we propose a new approach to automatic camera control that indirectly includes the player in the camera control loop.

To achieve this goal we have analysed the automatic camera control problem from a numerical optimization perspective and we have introduced a new optimization algorithm and camera control architecture able to generate real-time, smooth and well composed camera animations. Moreover, we have designed and tested an approach to model the player's camera preferences using machine learning techniques and to tailor the automatic camera behaviour to the player and her gameplay style.

Experiments show that, the novel optimisation algorithm introduced successfully handles highly dynamic and multi-modal fitness functions uch as the ones typically involved in dynamic camera control. Moreover, when applied in a commercial-standard game, the proposed automatic camera control architecture, shows to be able to accurately and smoothly control the camera. Finally, the results of a user survey to evaluate the suggested methodology for camera behaviour modelling and adaptation, shows that the resulting adaptive cinematographic experience is largely favoured by the players and it generates a positive impact on the game performance.

Contents

List of Figures

Chapter 1

Introduction

According to the Oxford Dictionary Of English (2005), cinematography is *"is the art of photography and camerawork in film-making"*; following this definition it is possible to define the concept of *virtual cinematography* (or virtual camera control) as the application of this art to virtual reality. With the term *interactive virtual cinematography* we identify the process of visualising the content of a virtual environment in the context of an interactive application such as a computer game. In this thesis, we propose the automation of such a creative process using automatic camera control, we investigate the challenges connected to camera automation and we examine solutions for them. In addition we investigate means for the player to influence the automatic control process.

1.1 Problem Definition and Motivation

A virtual camera represents the point-of-view of the player through which she perceives the game world and gets feedback on her actions. The perspective camera model in OpenGL [1] defines a virtual camera with six parameters: position, orientation, field of view, aspect ratio, near plane and far plane (see Figure 1.1). Camera position is a three-dimensional vector of real values defining a Cartesian position. Camera orientation can be defined either using a quaternion, a set of three Euler angles or a combination of two three-dimensional vectors describing the front direction and the up direction.

Camera placement — i.e the configuration of the camera parameters within the virtual environment — and *animation* — i.e. the process of transitioning from one set of camera parameters to another one — play a vital role in 3D graphics interactive applications and it deeply influences his way to perceive the environment and his ability to effectively accomplish any task. In applications such as 3D modellers and computer games, the virtual camera provides means of interaction with the virtual environment and has a large impact

[1] Open Graphics Library - http://www.opengl.og

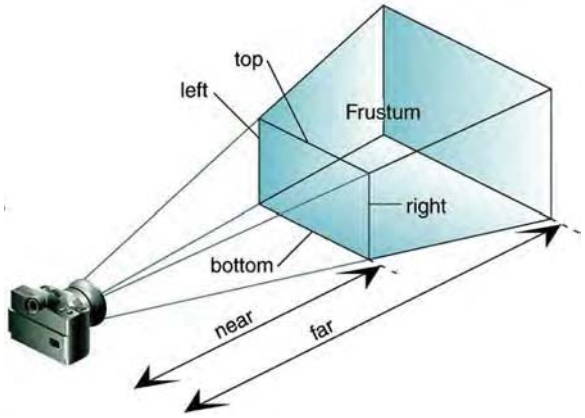


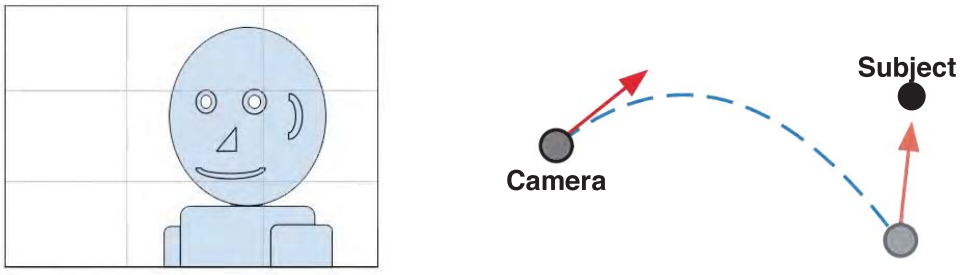
Figure 1.1: Standard virtual camera parameters.

on the usability and the overall user experience (Pinelle and Wong, 2008). Moreover, in 3D computer games the presentation of the game events largely depends on the camera position and movements, thus virtual camera control has a significant impact on aspects such as game-play and story-telling.

In computer games, the virtual camera is usually directly controlled by the player or manually predefined by a game designer. While the player's control is often assisted by the game, direct control of the camera by the player increases the complexity of the interaction and reduces the designer's control over game storytelling. On the other hand, designer placed cameras release the player from the burden of controlling the point of view, but they cannot guarantee a correct visualisation of all possible player actions. This often leads the game designers to reduce the range of possible player actions to be able to generate a more cinematographic player experience. Moreover, in multi-player games or in games where the content is procedurally generated, the designer has potentially no information to define a-priori the camera positions and movements.

Automatic camera control aims to define an abstraction layer that permits the control of the camera using high-level and environment-independent requirements, such as the visibility of a particular object or the size of that object on the screen. Given these requirements and the game state at any time, the camera controller should dynamically and efficiently calculate an optimal configuration. An optimal camera configuration is defined as the combination of camera settings which maximises the satisfaction of the requirements imposed on the camera, known as *camera profile* (Bares et al., 2000).

The camera requirements describe the desired camera in terms of abstract properties such as frame composition or motion. Frame composition (see Figure 1.2a) describes the way in which the objects should be framed by the camera, such as their position in the frame or the size of their projected image. Camera motion requirements describe the way



(a) Composition (b) Animation

Figure 1.2: An example of an automatic camera control problem: the camera controller is requested to find the position and orientation of the camera that produced the shot defined (a) and it has to animate to camera to the target position according to the characteristics of the animation (b)
.

the virtual camera should be animated in the virtual environment while framing the subjects (see Figure 1.2b). The definition of these requirements originates from the rules used to shoot real-world scenes in cinematography and photography (Arijon, 1991).

Finding the optimal camera that maximises the fulfilment of the designer requirements is a complex optimisation problem, despite its relatively low dimensionality (i.e. it has a minimum of 5 dimensions); the complexity of the objective function evaluation and the short execution time imposed to ensure a real-time execution, make the problem computationally hard. The evaluation of visual aspects such as *visibility* or *size of the project image* require computationally expensive operations such as rendering or ray-casting and their evaluation functions often generate terrains that are very rough for a search algorithm to explore. Moreover, in real-time camera control, the algorithm needs to find the best possible solution within the rendering time of a frame (typically between 16.6 and 40 ms in commercial games) and needs to maintain it throughout the computation while the environment changes due to the dynamic nature of ergodic media such as games.

Several techniques for automatic camera control have been proposed in the past — the reader is referred to (Christie et al., 2008) and Chapter 2 of this thesis for a comprehensive review. The most common approaches model the camera control problem as a constraint satisfaction or optimisation problem. These approaches allow the designer to define a set of requirements on the frames that the camera should produce and on the camera motion. Depending on the approach, the controller positions and animates one or more virtual cameras that attempt to satisfy the predefined requirements.

While automatic camera control aims to automatise the translation process between high level requirements and the camera animation and placement, the definition of the requirements is commonly delegated to a human designer which hand-crafts manually the

cinematographic experience. Some efforts have been dedicated also to the automation of these requirements (Bares and Lester, 1997a; Bares et al., 1998; Tomlinson et al., 2000). In particular the studies of Bares et al. (1997a; 1998) have investigated the personalisation of the cinematographic experience through task and user modelling.

In this thesis we propose a novel approach to interactive virtual cinematography in which we address both the problem of efficient and effective automatic camera control (see Section 1.3) and camera behaviour adaptivity (see Section 1.4). For this purpose we present an automatic camera control framework, we name CamOn, that approaches both with virtual camera composition (i.e. finding the best camera configuration to fulfil the composition requirements) and camera animation (i.e. finding the best camera trajectory to fulfil the motion requirements), we introduce a novel hybrid search algorithm able find and track the optimal camera configuration in dynamic real-time environments and, finally, we present a methodology to build camera user models and use them to generate adaptive cinematographic experiences for interactive 3D virtual environments.

1.2 Objectives and Key Contributions

This thesis aims to contribute primarily to the field of virtual cinematography; however, minor contributions to areas of research peripheral to camera control, such as player modelling, natural interaction and dynamic optimisation, are also present. An impact is expected also on the computer games industry as we believe that the instruments for interactive virtual cinematography described in this thesis could expand the design possibilities for future computer games.

The main research questions that will be answered in this thesis are as follows.

1. How to effectively approach camera composition and animation in real-time in dynamic and interactive environments.

2. How does automatic camera control affect the player experience in three-dimensional computer games.

3. Within the framework of automatic camera control, how can the player affect the cinematographic experience.

In order to answer these research questions, this thesis pursuits two objectives: develop and evaluate a real-time *automatic camera controller*, and design and evaluate a methodology for *adaptive camera control*. Our first hypothesis is that a combination of optimisation

and path planning can be used to successfully animate the camera in dynamic and interactive environments; moreover, we believe, that by hybridizing a population-based optimisation algorithm with a local search algorithm, a camera controller can achieve sufficient efficiency, accuracy and robustness to be employed in such conditions. We also hypothesise that controlling the camera using such an approach can effectively increase the quality of the player experience in computer games and can improve the player's performance. Our last hypothesis is that player modelling can be employed to implicitly involve the player in the control loop of the camera also in the context of automatic camera control.

In summary, the main contributions of this thesis are as follows.

- A novel architecture for automatic camera control that is able to address both the problems of virtual camera composition and virtual camera animation in interactive virtual environments (Burelli and Yannakakis, 2010a, 2012a).
- A novel hybrid optimisation algorithm for dynamic virtual camera composition that combines a Genetic Algorithm with an Artificial Potential Field based local search algorithm (Burelli and Jhala, 2009a,b; Burelli and Yannakakis, 2010b, 2012b).
- A novel methodology to generate personalised cinematographic experiences in computer games based on player modelling and adaptation (Picardi et al., 2011; Burelli and Yannakakis, 2011).

In the remanding of the chapter we introduce the concepts of automatic camera control and adaptive camera control.

1.3 Automatic Camera Control

To address the key problems of camera composition and animation, we propose a novel approach that combines advantages from a set of algorithms with different properties. In particular we combine a local search algorithm with a population-based algorithm to couple the computational efficiency of the first with the robustness of the second. Finally the solution found through the combination of these algorithms is used as a target to guide a 3D path planning algorithm.

At each iteration the system evaluates two types of input: the current state of the environment and the camera requirements. The first input class includes the geometry of the scene and the camera configuration defined as a three-dimensional position and a two-dimensional rotation (represented as spherical coordinates). The scene geometry is stored in a engine-dependent data structure (usually a scene tree) and it is used to evaluate the frame constraints that define the composition problem.

The second input class includes the desired frame composition properties — which describe how the scene rendered through the virtual camera should look like — and the desired camera motion properties. Frame composition properties describe the disposition of the visual elements in the image (Arijon, 1991); following the model proposed by Bares et al. (2000) we define a set of properties each of which may be applied to an object of interest for the camera.

The optimal camera configuration is calculated by optimising an objective function which is proportional to the satisfaction level of the required frame composition properties. For this purpose we have designed an hybrid optimisation algorithm that combines a Genetic Algorithm (Holland, 1992) with an Artificial Potential Field (Khatib, 1986) based search algorithm. The structure of this hybrid meta-heuristic algorithm follows the structure of a Genetic Algorithm with one main difference: a mutation operator that is driven by an APF is added to the standard crossover and mutation operators. Such an operator is added to exploit the knowledge of the objective function by employing an Artificial Potential Field optimisation algorithm based on an approximation of the composition objective function derivative.

The results of a comparative evaluation show that this hybrid Genetic Algorithm demonstrates the best reliability and accuracy across most of the investigated scenarios independently of task complexity.

Furthermore, the overall architecture combining the hybird Genetic Algorithm with a Probabilistic Roadmap Method (Kavraki et al., 1994) for camera animation is evaluated in a commercial-standard three-dimensional computer game production. A user survey on this game reveals both high levels of player satisfaction for the automatic camera and a significant preference for the automatic camera over the manual camera scheme.

1.4 Adaptive Camera Control

Virtual camera parameters are commonly hand-crafted by game designers and do not consider player preferences. Including the player in the definition of these parameters requires the construction of a model of the relationship between camera motion and player experience (Martinez et al., 2009). We aim to close the loop of automatic camera control by allowing the player to implicitly affect the automatic camera controller; for this purpose we investigate player preferences concerning virtual camera placement and animation, and we propose a model of the relationship between camera behaviour and player behaviour, and game-play. This model is used to drive the automatic camera controller and provide a personalised camera behaviour.

In this thesis we present a methodology to build user models of camera behaviour from the combination of player's gaze, virtual camera position and player's in-game behaviour data. Including gaze information allows for a finer analysis of the player's visual behaviour permitting, not only to understand what objects are visualised by the player, but also which ones are actually observed. Information on player's visual focus also allows to filter exactly which object is relevant for the player among the ones visualised by the player through her control of the virtual camera.

From this data a set of camera behaviours is extracted using a clustering algorithm and the relationship between such behaviours and the players' playing style is modelled using machine learning. This model is then used to predict which kind of camera behaviour does the user prefer while playing the game in order to appropriately instruct the camera controller to replicate such a behaviour.

1.5 Thesis Structure

The thesis is organized into chapters as follows.

Chapter 2 outlines the state-of-the-art in control, user modelling and their application to games; furthermore, it presents an extensive literature review of automatic virtual cinematography.

Chapter 3 presents a camera control architecture designed to successfully solve the virtual camera composition and animation problems and it describes the algorithms and techniques employed.

Chapter 4 describes a methodology to design adaptive cinematographic experiences in games by building user models of the camera behaviour and using them to control camera movements.

Chapter 5 presents the virtual environment developed to evaluate the algorithms and methodologies presented in this thesis.

Chapter 6 presents an evaluation of the optimisation algorithm at the core of the automatic camera control architecture presented in the thesis.

Chapter 7 showcases the application of the automatic camera control architecture presented in this thesis to a commercial-standard 3D platform game and it tests its performance trough a user evaluation experiment.

Chapter 8 describes a case study that showcases the application of the adaptive camera control methodology pesented in Chapter 4 to a 3D platform game. Moreover it presents a user evaluation of the resulting adaptive camera controller.

Chapter 9 summarizes the thesis' main achievements and contributions and discusses the proposed approaches' current limitations. Moreover, potential solutions that might embrace these drawbacks are presented.

Chapter 2

Related Work

This chapter briefly outlines the state-of-the-art of navigation and control, then it narrows its scope by presenting an extensive literature review of virtual camera control. It follows with an analysis of artificial intelligence and camera control in computer games and it concludes with a review of the techniques used for player modelling and adaptation in games.

2.1 Navigation and Control

Navigation (i.e. motion planning) has attracted the interest of different communities, such as nonlinear control, robotics and artificial intelligence. A classical motion planning problem raises several challenges (Salichs and Moreno, 2000): first, the agent must be able to control its movements while interacting with the external environment; second, the agent must be able to collect knowledge of the environment and be aware of its state within the environment, finally the agent must be able to identify the goal location and an optimal path that connects its current location to the goal.

In real-world motion control problems, such as autonomous vehicle control (Frazzoli et al., 2000), the controller is required to handle the agent's interaction with the physical world and it has to deal with aspects such as obstacle avoidance, inertia or speed. Common techniques for motion control include Artificial Potential Fields (Khatib, 1986), Visual Servoing (Corke, 1994) or Certainty Grids (Moravec and Elfes, 1985); such techniques address the motion control problem in different environment types and with different types of inputs, for instance Artificial Potential Fields require a global spatial information, while Visual Servoing uses local visual information acquired through one or multiple cameras.

Furthermore, if the world model is unknown, an agent also needs the ability to explore and learn about their environment, build a model of the world and identify its state within this model. The representation of such a model greatly varies among different approaches

depending on the type of information available about the world and the moment in which the learning takes place. Occupancy Grids (Filliat and Meyer, 2002) is a deterministic graph based model of the environment which is built during the navigation process, while Certainty Grids (Moravec and Elfes, 1985) employ Artificial Neural Networks to store the probability of a certain position to be occupied by an obstacle.

In order to reach the goal destination, an agent needs to identify a plan that connects its current position to the target one. Probably the most popular technique used for this task is A* (Hart et al., 1968). A* performs a best-first search to finds the least-cost path from a given initial cell to the goal cell in a discrete space. Other algorithms such as Reinforcement Learning (Sutton and Barto, 1998), Monte-Carlo Search (Barraquand and Latombe, 1990) or Probabilistic Roadmap Methods (Kavraki et al., 1994) address the path planning problems under different conditions such as a continuous space or an unknown distance heuristic function.

Controlling the view point in a 3D application shares some of the same problems as the virtual camera needs to be moved through a virtual environment composed by complex three-dimensional geometries; the next section will introduce how the problem has been modelled within the field of virtual camera control and it will show how some of the aforementioned techniques have been successfully applied to address different tasks.

2.2 Virtual Camera Control

Since the introduction of virtual reality, virtual camera control attracted the attention of a large number of researchers (refer to Christie et al. (2008) for a comprehensive review). Early approaches focused on the mapping between the degrees of freedom (DOF) for input devices to 3D camera movement. Ware and Osborne (1990) proposed the following set of mappings:

Eyeball in hand: the camera is controlled by the user as if she holds it in her hands; rotations and translations are directly mapped to the camera.

Scene in hand: the camera is pinned to a point in the world, and it rotates and translates with respect to that point; the camera faces always the point and rotates around it, the forward direction of the movements is the relative direction of the point. Such a metaphor has been explored also by Mackinlay et al. (1990) in the same period.

Flying vehicle control: the camera is controlled similarly to an aeroplane, with controls for translation and rotation velocity.

While these metaphors are currently still common in many virtual reality applications, direct manipulation of the several degrees of freedom of the camera soon demonstrated to be problematic for the user, leading researchers to investigate how to simplify camera control (Phillips et al., 1992; Drucker and Zeltzer, 1994). Another control metaphor, called through-the-lens camera control, was introduced by Gleicher and Witkin (1992) and was refined by Kyung and Kim (1995). In through-the-lens camera control, the user controls the camera by translating and rotating the objects on the screen; the camera parameters are recomputed to match the new users desired locations on screen by calculating the camera displacement from the object's screen displacement. Christie and Hosobe (2006) extended this metaphor by including an extended set of virtual composition primitives to control the features of the image.

In parallel to the research on control metaphors, a number of researchers investigated the automation of the camera configuration process. The first example of an automatic camera control system was showcased in 1988 by Blinn (1988). He designed a system to automatically generate views of planets in a space simulator for NASA. Although limited in its expressiveness and flexibility and suitable only for non interactive applications, Blinn's work served as an inspiration to many other researchers that investigated more efficient solutions and more flexible mathematical models able to handle more complex aspects such as camera motion and frame composition (Arijon, 1991) in static and dynamic contexts.

2.3 Automatic Camera Control

Automatic camera control identifies the process of automatically configuring the camera in a virtual environment according to a set of requirements. It is a non-linear automatic control problem (Pontriagin, 1962): the system's input are the requirements on composition and motion, while the internal state of the system is defined by the camera parameters — e.g. position and rotation. Following this model it is possible to identify three main research questions:

- How to identify the best inputs for the system.
- How to find the optimal state of the camera with respect to the given inputs.
- How to control the dynamic behaviour of the camera.

The first problem, the definition of the system's inputs, is often referred as camera/shots planning and it deals with the selection of the shot type to be used to frame a certain event in the virtual world. The second problem deals with the approach to be used to find the optimal camera configuration that satisfies the input requirements. Constraint satisfaction

or optimisation techniques are often used for this purpose. The last problem deals with the control of the camera movements during the framing of one scene and during the transitions between two subsequent shots.

2.4 Camera Planning

The camera planning problem was defined for the first time in (Christianson et al., 1996) as the problem of automatically scheduling the sequence of shots to film one or more events in a virtual environment. Christianson et al. proposed a language (DCCL) to define shot sequences and to automatically relate such sequences to events in the virtual world. He et al. (1996) extended the concept of idioms within DCCL by modelling them as a finite state machines and allowing for richer expressiveness. Bares and Lester (1997b; 1998) suggested and evaluated a system that selects the most appropriate camera settings to support different user's tasks in a virtual learning environment. Charles et al. (2002) and Jhala and Young (2005) investigated the automatic generation of shot plans from a story. Moreover, Jhala and Young (2009; 2010) proposed an evaluation method for such task, based on the users' understanding of the story represented.

An alternative approach to shot generation through planning has been proposed by various researchers. Tomlinson et al. (2000) modelled the camera as an autonomous virtual agent, called CameraCreature, with an affective model and a set motivations. The agent shots the most appropriate shot at every frame according to the events happening in the environment and its current internal state. Bares and Lester (1997a) investigated the idea of modelling the camera behaviour according to the user preferences to generate a personalised cinematographic experience. The user model construction required the user to specifically express some preferences on the style for the virtual camera movements. In this thesis, we propose a methodology to build user profiles of camera implicitly by capturing a user's gaze movements on the screen during the interaction.

2.4.1 Virtual Camera Composition

The aforementioned approaches addressed the problem of automatic shot selection, however, once the shot has been selected it is necessary to identify the best camera configuration to convey the desired visuals. This process involves two aspects: the definition of the desired shot in terms of composition rules and the calculation of the camera parameters.

The first seminal works addressing virtual camera composition according to this model (Jardillier and Languénou, 1998; Bares et al., 2000; Olivier et al., 1999) defined the concept of frame constraint and camera optimisation. These approaches require the designer to define a set of required frame properties which are then modelled either as an objective

function to be maximised by the solver or as a set of constraints that the camera configuration must satisfy. These properties describe how the frame should look like in terms of object size, visibility and positioning.

Jardillier and Languénou (1998) as well as Bares et al. (2000) modelled the visual properties as constraints and looked for valid camera configurations within the constrained space. The solver suggest by Bares et al., as well as all the other constraint satisfaction approaches, returns no solution in the case of conflicting constraints. Bares and Lester (1999) addressed the issue by identifying conflicting constraints and producing multiple camera configurations corresponding to the minimum number of non-conflicting subsets.

In contrast to constraint satisfaction approaches, global optimisation approaches (Olivier et al., 1999; Halper and Olivier, 2000) model frame constraints as an objective function (a weighted sum of each required frame property) allowing for partial constraint satisfaction. These approaches are able to find a solution also in case of conflicting constraints; however, they may converge to a near-optimal solution and their computational cost is usually considerably higher than the constraint satisfaction ones. A set of approaches (Pickering, 2002; Christie and Normand, 2005; Burelli et al., 2008) address this computational cost issue by combining constraint satisfaction to select feasible volumes (thereby reducing the size of the search space) and optimisation to find the best camera configuration within these spaces. While such solutions are more robust than pure constraint satisfaction methods, the pruning process might still exclude all possible solutions.

An algorithm's computational cost becomes a critical factor especially when the algorithm is applied for control on real-time dynamic virtual environments. In the camera control context the controller is required to produce a reasonable solution at short intervals (potentially down to 16.6ms) for an indefinitely long time. For this reason, researchers have also investigated the application of local search methods; for instance, Bourne et al. (2008) proposed a system that employs sliding octrees to guide the camera to the optimal camera configuration.

Local search approaches offer reasonable real-time performance and handle well frame coherence, but often they converge prematurely to local minima. This characteristic becomes critical when the camera control system has to optimise the visibility of a subject in the frame since the visibility heuristic consists of many local minima areas with almost no gradient information available to guide local search.

Occlusion

Successfully handling object occlusion constitutes a vital component of an efficient camera controller (Christie et al., 2008). Object visibility plays a key role in frame composition.

For instance, an object satisfying all frame conditions (e.g. position in frame and projection size) does not provide any of the required visual information if it is completely invisible due to an occlusion.

The object occlusion problem can be separated in two dissimilar yet dependent tasks: occlusion evaluation/detection and occlusion minimisation/avoidance. An occlusion occurs when one of the subjects the camera has to track is hidden (fully or partially) by another object. A common technique to detect occlusion consists of casting a series of rays between the object of interest and the camera (Burelli et al., 2008; Bourne et al., 2008). A similar approach (Marchand and Courty, 2002) generates a bounding volume containing both the camera and the object of interest and checks whether other objects intersect this volume. A third approach (Halper et al., 2001) exploits the graphic hardware by rendering the scene at a low resolution with a colour associated to each object and checking the presence of the colour associated to the object of interest. All the optimization algorithms examined in this thesis perform occlusion-detection using ray casting as presented in Chapters 6,7 and 8 of the thesis.

Based on their occlusion detection technique, Halper et al. (2001) deal with the problem of occlusion avoidance by sequentially optimising each constraint and solving visibility as the last problem in the sequence; therefore, occlusion minimisation overrides all the previous computations. Bourne et al. (2008) devised an escape mechanism from occluded camera positions which forces the camera to *jump* to the first non-occluded position between its current position and the position of the object of interest. Their approach, however, considers just one object of interest.

Pickering (2002) proposed a shadow-volume occlusion avoidance algorithm where the object of interest is modelled as a light emitter and all the shadow-volumes generated are considered occluded areas. However, the aforementioned approach to occlusion avoidance is not suitable for real-time applications like games due to their high computational cost. Lino at al. (2010), within their system on real-time cinematography, use a similar approach, based on a cell-and-portal visibility propagation technique, which achieves real-time performance.

2.4.2 Camera Animation

The aforementioned virtual camera composition approaches calculate an optimal camera configuration given a shot description. However they do not take into consideration the dynamic nature of the scene and the camera. The camera is positioned in order to maximise the satisfaction of the current requirements without considering the previous and the future states of the system. Nevertheless, these aspects are extremely relevant in the automatic generation of camera animations and the automatic control of the camera in dynamic

environments. In these conditions, continuity in camera position, smooth animations and responsiveness become critical aspects.

Beckhaus et al. (2000) proposed a first approach to automatise the camera animation for virtual museum tours generation. Their system used an Artificial Potential Field (APF) to guide the camera through the museum: each painting was modelled as low potential area attracting the camera and every time a painting was reached the potential of the area was deactivated so that the camera would smoothly continue towards the next paining. The system suggested by Bourne et al. (2008) is able to control also the camera animation during the optimisation process, this is a characteristic common to local search approaches, as subsequent solutions are normally close to each other an can, therefore, be used to animate the camera.

Such an approach combines both the identification of the target camera configuration and the animation of the camera towards the target. However, due to the hill-climbing nature of the algorithms adopted these approaches often fail to find a smooth path for the camera as they tend to prematurely converge into local optima. A set of approaches (Nieuwenhuisen and Overmars, 2004; Baumann et al., 2008; Oskam et al., 2009) have focused purely on the camera path planning task and have proposed different forms of the probabilistic roadmap method to generate smooth and occlusion aware camera paths.

The camera control framework presented in this thesis addresses the problems of camera composition and animation by combining different algorithms. In particular, we couple a local search algorithm with a population-based algorithm to combine the computational efficiency of the first with the robustness of the second (Burelli and Yannakakis, 2012b). Finally the solution found through the combination of these algorithms is used as a target to guide a 3D path planning algorithm (Burelli and Yannakakis, 2012a).

2.5 Camera Control in Games

Computer games stand as a natural benchmark application for automatic camera control techniques as they impose the necessity for real-time execution; moreover, the camera needs to react to unexpected and dynamic changes of the environment and to accommodate the behaviour of the player to foster an optimal interactive experience. Despite this, in the game industry, virtual cinematography has received considerably less attention than other aspects such as visual realism or physics, and camera control has been mostly restricted to the following standard interaction paradigms (Christie et al., 2008):

First person: the camera position and orientation corresponds to the player's location in the virtual environment; therefore, the camera control scheme follows the character

control scheme. Examples of games adopting such a camera control scheme include *Quake* (id Software, 1996) and *Halo: Combat Evolved* (Microsoft Game Studios, 2001).

Third person: the camera either follows the character from a fixed distance with different angles to avoid obstacles in the environment or multiple cameras are manually placed in the environment and the view point changes according to the position of the main game character. A different variant of the first type of camera control paradigm is used in strategy or managerial games; in these games the target of the camera is freely selectable by the player. Examples of games adopting such a camera control scheme include *Tomb Raider* (Eidos Interactive, 1996) and *Microsoft Game Studios* (Microsoft Game Studios, 2007).

Cut-scenes and replays: in these non interactive phases of the games, the camera focuses on representing the important elements of the story rather than focuses on fostering the interaction. It is often used in sport games (i.e. replay) and in story-heavy games (i.e. cut-scenes). Games featuring such a camera control scheme include *Metal Gear Solid* (Konami, 1998) or most sport video games.

Recent games demonstrate an increasing tendency to extend this list of paradigms and enhance the player experience by employing cinematographic techniques to portrait narrative and interaction in games. Examples such as *Heavy Rain* (Sony Computer Entertainment, 2010) or the *Uncharted* (Sony Computer Entertainment, 2007) series show extensive usage of cinematic techniques to frame the in-game actions (see Figure 2.1a). In such games, however, the cameras are set manually in place during the development of the game; reducing heavily the movement and the actions the player can take. Moreover, such a solution would be inapplicable in games in which the content is not known in advance since it is either procedurally generated (Yannakakis and Togelius, 2011; Shaker et al., 2010) or crowd sourced — e.g. *World Of Warcraft* (Vivendi, 2004).

Some simple dynamic techniques have been applied to more action oriented games such as *Gears Of War* (Microsoft Game Studios, 2006) (see Figure 2.1b), in which the camera changes relative position and look-at direction automatically to enhance some actions or allow for a better view of the environment.

In games, a common camera control problem, involves following and keeping visible one or more subjects while avoiding obstacles. Moreover, aesthetics and narrative can be supported during and between actions by applying high composition and animation rules. Halper et al. (2001) proposed an automatic camera control system specifically designed for computer games, in which he highlights the necessity of frame-coherence (smooth changes in camera location and orientation) to avoid disorienting the player during the game.

(a) A screen-shot from Heavy Rain by Quantic Dream, demonstrating usage of cinematographic techniques.

(b) A screen-shot from Gears Of War by Epic during a running action; in such context the camera moves downward and shakes to enhance the haste sensation.

Figure 2.1: Examples of advanced camera control in modern computer games.

Hamaide (2008) presented a camera system developed by *10Tacle Studios* based on the controller by Bourne et al. (2008); this system extends the one by Bourne et al. by adding more advanced control on the camera dynamic behaviour by supporting for predefined movements paths.

None of these systems, however, supports both composition on multiple subjects and animation contemporary, being unable to support cinematographic representations of the game actions. Moreover, these works focus primarily on the automation of camera animation and occlusion avoidance not considering aspects such as shots planning and editing. This thesis addresses these aspects by applying machine learning to model the players' preferences on camera behaviour; such computational models are used to identify the most appropriate shot to be selected during gameplay

2.6 Artificial Intelligence in Games

Artificial Intelligence in games has been employed in a variety of forms to address a multitude of problems such as procedural content generation (Togelius et al., 2010), game playing (Lucas, 2008) or player modelling (Houlette-Stottler, 2004; Yannakakis and Maragoudakis, 2005).

Early attempts to use AI to learn how to play a game date back to the fifties (Shannon, 1950); however, it was not before the nineties that machine learning was employed for the task (Tesauro, 1994; Thrun, 1995). Learning to play a computer game very often stands as a more complex task than board or card games, due to larger space of states, higher unpredictability and larger number of possible actions. Therefore several approaches attempted to learn playing computer games such as Super Mario Bors (Togelius et al.,

2009), Pacman (Lucas, 2005) or TORCS (Cardamone et al., 2009).

Another prominent direction of research in the field investigates the application of machine learning for content generation in games. The primary challenges of procedural content generation are: how to represent the game content, how to evaluate the content quality and how to find the best content configuration. Machine learning has been used to address all these problems: for instance, evolution has been employed to generate tracks for racing games (Togelius et al., 2006) or to generate strategy game units (Mahlmann et al., 2011) and ANNs have been used to model the weapons in a multi-player space fight game (Hastings et al., 2009). Shaker et al. (2010) built a set of ANNs models of player experience and employed them as evaluation functions for the game content.

Bares and Lester (1997a) proposed an explicit method to build user models of camera behaviour used to generate personalised camera control experiences, this thesis draws upon this work and modern player modelling to design a method to generate adaptive cinematographic experiences in computer games.

2.6.1 Player Modelling

The term player modelling identifies the application of user modelling to games (Charles and Black, 2004; Houlette-Stottler, 2004; Yannakakis and Maragoudakis, 2005). Player modelling has been employed on different aspects of games; however, it is possible to isolate two main purposes: a post-hoc analysis of the players' in-game behaviour (Drachen et al., 2009) or as as the first step to adapt game content (Yannakakis and Togelius, 2011).

Clustering techniques have been used to isolate important traits of the player behaviour: self-organizing maps have been used to identify relevant game-play states (Thurau et al., 2003) or to identify player behaviour types from game-play logs (Thawonmas et al., 2006; Drachen et al., 2009) and neural gas (Thurau et al., 2004) has been applied to learn players' plans. Thurau et al. (2003; 2004) coupled clustering techniques with supervised learning to build believable *Quake II* (Activision, 1997) bots.

Viewing player modelling as an initial step in the design process of adaptive behaviour in games, Yannakakis and Maragoudakis (2005) combined naive Bayesian learning and on-line neuro-evolution in Pac-Man to maximize the player's entertainment during the game by adjusting NPCs behavior. Thue et al. (2007) built a player profile during the game, based on theoretical qualitative gameplay models, and used this profile to adapt the events during an interactive narrative experience.

Yannakakis at al. (2010) studied the impact of camera viewpoints on player experience and built a model to predict this impact. That research study demonstrates the existence of

a relationship between player emotions, physiological signals and camera parameters; however, since the relationship is built on low level camera parameters, the findings give limited information about the visual features which are more relevant for the player. Therefore, in the light of these results, in this thesis we further investigate the relationship between camera and player experience to automate the generation and selection of the virtual camera parameters. More specifically, in this thesis, we attempt to incorporate alternative player input modalities (i.e. gaze) to model the user's visual attention for camera profiling.

2.7 Gaze Interaction in Games

Eye movements can be recognised and categorised according to speed, duration and direction (Yarbus, 1967). In this paper, we focus on *fixations*, *saccades* and *smooth pursuits*. A fixation is an eye movement that occurs when a subject focuses at a static element on the screen; a saccade occurs when a subject is rapidly switching her attention from one point to another and a smooth pursuit is a movement that takes place when a subject is looking at a dynamic scene and she is following a moving object.

Research on gaze interaction in computer games include studies on the usage of gaze as a direct player input (Nacke et al., 2009; Munoz et al., 2011) and studies on gaze as a form of implicit measure of the player's state. El-Nasr and Yan (2006), for instance, used an eye tracker to record eye movements during a game session to determine eye movement patterns and areas of interest in the game. Moreover, they employed this information to calculate the areas of the game that necessitate a higher graphic quality.

Sundstedt et al. (2008) conducted an experimental study to analyse players' gaze behaviour during a maze puzzle solving game. The results of their experiment show that gaze movements, such as fixations, are mainly influenced by the game task. They conclude that the direct use of eye tracking during the design phase of a game can be extremely valuable to understand where players focus their attention, in relation to the goal of the game. Bernhard et al. (2010) performed a similar experiment using a three-dimensional first-person shooter game in which the objects observed by the players were analysed to infer the player's level of attention. We are inspired by the experiment of Bernhard et al. (2010); unlike that study however, in this thesis, we analyse the player's gaze patterns to model the player's camera movements, and moreover, investigate the relationship between camera behaviour, game-play and player-behaviour.

Chapter 3

Automatic Camera Control

This chapter[1] presents the algorithms and techniques adopted to successfully tackle the virtual camera composition and animation problems.

In the first part of the chapter, virtual camera composition and animation are defined as numerical optimisation problems and the concepts of frame and animation constraints are introduced and each constraint is described in detail with references to the state-of-the art. The chapter continues with an overview of the architecture of CamOn, a detailed description of the optimisation and path planning algorithms employed in its development and evaluation, and it concludes with a summary of the chapter's content.

3.1 Frame constraints

In order to define virtual camera composition as an optimisation problem, we primarily need to identify the number and types of key attributes that need to be included in the objective function of the optimisation problem.

The characteristics of the search space of the optimisation problem depends on the number and type of parameters used to define the camera. The choice of parameters affects both the dimensionality of the optimisation problem (thus, the performance of the optimisation process) and the expressiveness, in terms of shot types that can be generated. A virtual camera is commonly defined by six parameters: position, orientation, field of view, aspect ratio, near plane and far plane. Camera position is a three-dimensional vector of real values defining a Cartesian position. Camera orientation can be defined either using a quaternion, a set of three Euler angles or a combination of two three-dimensional vectors describing the front direction and the up direction. by including the last four parameters, the domain of the automatic camera control objective function is at least 10-dimensional. However, since aspect ratio, field of view, near and far planes and the camera up vector are

[1] Parts of this chapter have been published in (Burelli and Jhala, 2009b,a; Burelli and Yannakakis, 2010a,b) and have been submitted for publication in (Burelli and Yannakakis, 2012b)

Bares et al.	CamOn
OBJ PROJECTION SIZE	Object Projection Size
OBJ VIEW ANGLE	Vantage Angle
OBJ IN FIELD OF VIEW	Object Visibility
OBJ OCCLUSION MINIMISE	
OBJ EXCLUDE OR HIDE	
OBJ OCCLUSION PARTIAL	
OBJ PROJECTION ABSOLUTE	Object Projection Position
CAM POS IN REGION	Camera Position

Table 3.1: Comparison between the frame constraints defined by Bares et al. (2000) and the frame constraints supported by the camera controller described in this thesis.

commonly constant, we reduce the domain to five dimensions describing camera position and orientation.

Bares et al. (2000) described the concept of frame constraint as a set of requirements that can be imposed to define a camera control problem. Every frame constraint is converted into an objective function that, in a linear combination with all the constraints imposed, defines a camera control objective function (Olivier et al., 1999). An example of a frame constraint is *OBJ PROJECTION SIZE* that requires the projection of an object to cover a specified fraction of the frame.

We consider a reduced set of frame constraints: *Vantage Angle*, *Object Projection Size*, *Object Visibility* and *Object Projection Position*. These four constraints serve as representatives of all the constraints listed by Bares et al. (2000). Table 3.1 contains a comprehensive comparison of such constraints and their equivalent frame constraint in the CamOn system. In the remaining of this section we present these frame constraints, their corresponding objective functions as well as how each constraint relates to one or more constraints of the aforementioned list. Note that the terms *fitness* and *objective function value* will be used interchangeably in this chapter to describe the same concept as most of the algorithms considered are based on population-based meta-heuristic algorithms.

3.1.1 Vantage Angle

This constraint binds the camera position to the position and rotation of a target object. It requires the camera to be positioned so that the angle between the target object front vector and the front vector of the camera equals to a certain value. A vantage angle constraint is defined by three parameters: the target object, the horizontal angle and the vertical angle. Figure 3.1 depicts three sample shots showcasing the relationship between the angles, the target object and the generated shot. The objective function f_θ of this frame constraint

(a) Angle: 0, 0 (b) Angle: -90, 0 (c) Angle: 45, 45

Figure 3.1: View angle sample shots. Each shot is identified by a horizontal and a vertical angle defined in degrees.

quantifies how close to the required angle is the position of the camera and it is defined as follows:

$$
\begin{aligned}
f_\theta &= f_\alpha \cdot f_\beta \\
f_\alpha &= 1 - \frac{|\hat{\vec{P}}_x - \vec{H}_x| + |\hat{\vec{P}}_z - \vec{H}_z|}{4} \\
f_\beta &= 1 - \frac{|\vec{V}_y - \hat{\vec{P}}_y|}{2} \\
\vec{P} &= \vec{C} - \vec{T} \\
\vec{V} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \times \vec{F} \\
\vec{H} &= \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \times \vec{F}
\end{aligned}
\tag{3.1}
$$

where α is the desired horizontal angle, β is the desired vertical angle, $\vec{F}$ is the target's front vector, $\vec{C}$ is the current camera position, $\vec{T}$ is the current target position and $\hat{\vec{P}}$ is the normalised relative direction of the camera with respect to the target object. Using this constraint it is also possible to control only one angle; in which case, f_θ equals either to f_α or f_β depending on the angle that should be constrained.

This frame constraint is equivalent to *OBJ VIEW ANGLE* constraint of the Bares et al. (2000) list (see Table 3.1).

3.1.2 Object Projection Size

This constraint binds the camera position and rotation to the position and size of a target object. It requires the area covered by the projection of a target object to have a specific

(a) Size: 1.0 (b) Size: 0.5 (c) Size: 1.5

Figure 3.2: Object projection size sample shots. Each shot is identified by a projection size defined as the ratio between the the longest side of the object's projection bounding box and the relative side of the frame.

size. The object projection size constraint is defined by two parameters: the target object and the fraction of the frame size that the projection should cover. Figure 3.2 shows three sample shots demonstrating the relationship between the projection size, the target object and the generated shot. The objective function f_σ of this frame constraint quantifies the proximity of the current camera position to the closest position which generates a projection of the object covering the desired size. It is calculated as:

$$f_\sigma = \begin{cases} \frac{\sigma_c}{\sigma_d} & \textit{if } \sigma_d > \sigma_c, \\ \frac{\sigma_d}{\sigma_c} & \textit{otherwise.} \end{cases} \tag{3.2}$$

where σ_d is the desired projection size and σ_c is the actual projected image size of the target object. The size of an object's projected area can be calculated with different levels of approximation. Maximum accuracy can be obtained by rendering to an off-screen buffer and counting the area covered by the object. The bounding box and sphere can also be used effectively for the area calculation. While these approximations drastically decrease the computational cost of the objective function evaluation they also provide less accuracy. Using the bounding sphere of the object is the fastest evaluation method but it approximates poorly most of the possible targets, especially human-shaped objects. In the current implementation of the evaluation function the target object is approximated using its bounding box and the projected area size is calculated using Schmalstieg and Tobler's (1999) method. This frame constraint corresponds to the *OBJ PROJECTION SIZE* constraint of the Bares et al. (2000) list (see Table 3.1).

(a) Visibility: 1.0 (b) Visibility: 0.6 (c) Visibility: 0.5

Figure 3.3: Object visibility sample shots. Each shot is identified by a visibility value defined as the ratio between the visible area of the object and its complete projected area.

3.1.3 Object Visibility

This constraint binds the camera position and rotation to the position and size of a target object. It requires the target object to be included in the frame and not hidden by any other object; both conditions are necessary to identify the target object as visible. In order to respect these two requirements the camera should be placed at a sufficient distance from the target and oriented in order to frame the target. Moreover, the volume between the camera and the target object should not contain obstacles that hide the target object.

Every opaque object in the virtual environment can potentially act as an obstacle and generate an occlusion. Figure 3.1 illustrates three sample shots showcasing the relationship between the visibility value, the target object and the generated shot. The objective function f_γ of this frame constraint quantifies the rate between the actual visible area of the projected image of the object and its total projected area and it is defined as:

$$\begin{aligned} f_\gamma &= 1 - |\gamma_d - \gamma_c| \\ \gamma_c &= \frac{\sum_{i=1}^{N} infov(\vec{v}_i)}{N} \frac{\sum_{i=1}^{N_e} (1 - occ(\vec{e}_i))}{5} \\ infov(\vec{x}) &= \begin{cases} 1 & \textit{if } \vec{x} \textit{ is in the view frustum,} \\ 0 & \textit{otherwise.} \end{cases} \\ occ(\vec{x}) &= \begin{cases} 1 & \textit{if } \vec{x} \textit{ is occluded,} \\ 0 & \textit{otherwise.} \end{cases} \end{aligned} \tag{3.3}$$

where γ_c is the current visibility value of the target object, γ_d the desired visibility value, $\vec{v}_i$ is the position of the i^{th} vertex of the object's mesh, N is the number of vertices of the mesh, function $infov(\vec{x})$ calculates whether a point is included in the field of view or not, $\vec{e}$ is the list containing the positions of the four extreme vertices in field of view — i.e. the

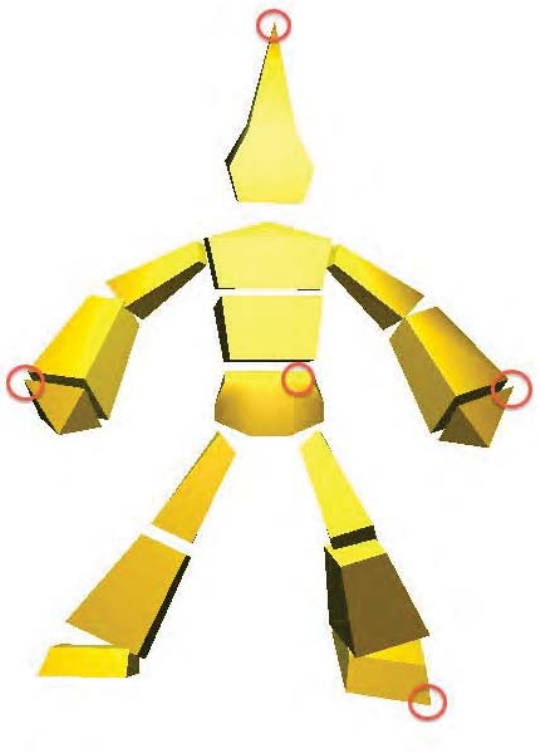


Figure 3.4: Example of the 5 points used to check visibility in the Object Visibility objective function.

top, bottom, left and right vertices on screen — and the one closer to the center of the projected image, and N_e is equal to 5 (an example of these points is depicted in Fig. 3.4). The $occ(\vec{x})$ function calculates whether the point $\vec{x}$ is occluded by another object or not.

The first part of the visibility function returns the fraction of the object which is in the field of view, while the second part returns the fraction of that part which is not occluded, the product of these two values is the overall visibility.

The implemented version of the function is optimised not to calculate the second part of the function if the first part is equal to 0. The occlusion check is implemented by casting a ray towards the point defined by the vector $\vec{x}$ and then checking whether the ray intersects any other object other than the target. The $infov(\vec{x})$ function is implemented by checking whether the point defined by $\vec{x}$ is included within the six planes composing the view frustum.

The object visibility constraint includes the *OBJ IN FIELD OF VIEW*, *OBJ OCCLUSION MINIMISE*, *OBJ EXCLUDE OR HIDE* and *OBJ OCCLUSION PARTIAL* constraints of the list proposed by Bares at al. (2000) (see Table 3.1). The first two can be obtained by setting the desired visibility to 1, the third by setting it to 0, while any number between these two cases expresses the fourth constraint.

(a) Position: 0.25, 0.25 (b) Position: 0.75, 0.75 (c) Position: 0.5, 0.5

Figure 3.5: Object frame position sample shots. Each shot is identified by a two-dimensional vector describing the position of the object's center in the frame.

3.1.4 Object Frame Position

This constraint binds the camera position and rotation to the position of a target object. It requires the target object to be included in the frame at a specific two-dimensional location. The object frame position constraint is defined by three parameters: the target object, the desired horizontal position and the desired vertical position. Figure 3.5 shows three sample shots demonstrating the relationship between the frame position, the target object and the generated shot.

The objective function f_ρ of this frame constraint quantifies how close to the required orientation is the camera and it is defined as follows:

$$f_\rho = 1 - \frac{|\vec{p_a} - \vec{p_d}|}{\sqrt{2}} \tag{3.4}$$

where $\vec{p_a}$ is the two-dimensional position of the target object in the frame and $\vec{p_d}$ is the desired position. Both vectors are defined between (0,0) and (1,1) where (0,0) corresponds to the lower left corner of the frame and (1,1) corresponds to the upper right corner of the frame. By combining object frame position and object projection size constraints it is possible to express the *OBJ PROJECTION ABSOLUTE* constraint, since it is possible to control both size and location of the object's projected image.

3.1.5 Camera Position

This constraint binds the camera position to a specific location in the virtual environment. It is improperly considered a frame constraint as it does not bind the solution to a characteristic of the image to be rendered; however, it is often useful to be able to bind the camera to a certain location. Moreover, the constraint has been included for completeness

with respect to the reference list defined by Bares et al. (2000) as it corresponds to the *CAM POS IN REGION* constraint (see Table 3.1).

The objective function f_ϵ of this frame constraint expresses how close the camera is to the region of space identified by the positions $\vec{v}_{max}$ and $\vec{v}_{min}$ and it is defined as follows:

$$f_\epsilon = \begin{cases} 1 & \text{if } \vec{v} < \vec{v}_{max} \wedge \vec{v} > \vec{v}_{min} \\ 1 - \frac{min(|\vec{v}-\vec{v}_{max}|,|\vec{v}-\vec{v}_{min}|)}{D_{max}} & \text{otherwise.} \end{cases} \tag{3.5}$$

where D_{max} is the maximum distance between two points in the virtual environment.

3.1.6 Composition Objective Function

The complete virtual camera composition objective function is a linear combination of the four aforementioned objective functions. Each objective function corresponds to a frame constraint imposed on a certain object, the complete objective function f is given by:

$$f = \sum_{i=1}^{N_\gamma} w_{\gamma i} f_{\gamma i} + \sum_{i=1}^{N_\sigma} w_{\sigma i} f_{\sigma i} + \sum_{i=1}^{N_\theta} w_{\theta i} f_{\theta i} + \sum_{i=1}^{N_\rho} w_{\rho i} f_{\rho i} + \sum_{i=1}^{N_\epsilon} w_{\epsilon i} f_{\epsilon i} \tag{3.6}$$

where N_γ, N_σ, N_θ, N_ρ and N_ϵ are, respectively, the number of object visibility, object projection size, vantage angle, object frame position and camera position constraints. $w_{\gamma i}$ and $f_{\gamma i}$ are, respectively, the weight and the objective function value of the i^{th} object visibility constraint; $w_{\sigma i}$ and $f_{\sigma i}$ are, respectively, the weight and the objective function value of the i^{th} object projection size constraint; $w_{\theta i}$ and $f_{\theta i}$ are, respectively, the weight and the objective function value of the i^{th} vantage angle constraint; $w_{\rho i}$ and $f_{\rho i}$ are, respectively; the weight and the objective function value of the i^{th} object frame position constraint; and $w_{\epsilon i}$ and $f_{\epsilon i}$ are, respectively, the weight and the objective function value of the i^{th} camera position constraint.

3.2 Animation Constraints

The objective function presented in Eq. 3.6 describes the ideal shot the camera should generate; however, in a dynamic virtual environment, the camera cannot jump continuously to the optimal configuration as this would often provoke a completely disorienting behaviour (e.g. a virtual camera instructed to frame flickering object, such as a flame, would continuously jump forward and backward to maintain the object's projection size).

In dynamic environments, it is necessary for an automatic camera controller to moderate also the camera motions/animation properties. For this purpose we identify a minimal set of four key motion constraints that the system should support, which are as follows:

- **Camera Movement Speed**: Defines the speed in space units per second at which the the camera follows the ideal position.
- **Camera Rotation Speed**: Defines the speed in degrees per second at which the camera adjusts towards ideal rotation.
- **Frame Coherence**: Defines the threshold value (minimum percentage of visible targets) for generating a cut — i.e. if the current visible surface of all the targets is below the fraction defined by frame coherence (or there is no available path connecting the current camera position with the next one) the controller will generate a cut, and will place to camera to the new position without generating any animation.
- **Obstacle Avoidance**: A predefined boolean value that controls whether the camera should or should not avoid the objects in the scene during its motion.

More constraints could be included in the set to support more cinematographic animations such as constraints on motion directions or constraints on frame properties during motion (Lino et al., 2010). However, the proposed set guarantees an essential control of the camera dynamic behaviour.

3.3 Controller's Architecture

The structure of CamOn (see Fig. 3.6) is the result of an iterative design process. Through the different design phases we have analysed the virtual camera composition objective function (Eq. 3.6) and we have evaluated different algorithms. The final results of this empirical process are presented in Chapter 6.

The architecture of the controller includes two main components: an optimisation module and an animation module (see Figure 3.6). The controller operates iteratively: at each iteration, the optimisation module finds the next best camera solution according to the frame constraints defined to describe the current shot, while the animation module animates the camera towards this solution according to the animation constraints. The inputs considered by the controller are: the current camera configuration, the current geometrical configuration of the virtual environment, the frame constraints and the animation constraints. The output of the controller is a new virtual camera configuration. The parameters of the camera which are not controlled by CamOn (e.g. field of view) can be directly set on the camera.

The controller balances the computational resources given to the two modules at each iteration in order to guarantee real-time execution; therefore, depending on the computational resources available, the controller will pause the optimisation process to guarantee

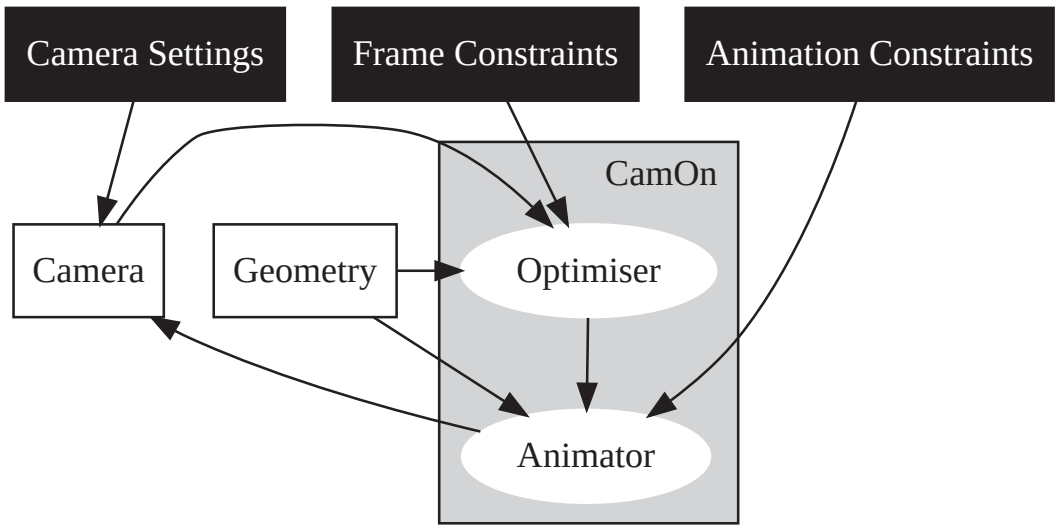


Figure 3.6: Controller's architecture. The black squares identify the controllable settings, while the white squares identify the virtual environment features considered by the controller. The two modules of the architecture are identified by the white ellipses.

an execution of each iteration in 16.6 ms (60 iterations per second). For this reason, only few steps of the optimisation are executed each cycle; therefore, the optimiser has to search an optimal solution in a dynamic problem in which the geometry might change during the optimisation process. After the optimisation phase is concluded the animation module calculates the new configuration of the virtual camera according to the animation constraints and the best solution found so far by the optimiser.

The controller's execution flow is as follows:

1. The system takes the current camera configuration, the frame constraints, the animation constraints and the current virtual environment's geometry configuration as input.

2. The next potential camera configuration that optimises the frame composition objective function is calculated by the optimisation module.

3. A path connecting the potential new camera configuration to the current one is calculated by the animation module.

4. If a path connecting the two points is available, the camera position and rotation are animated towards the new solution according to the animation constraints; otherwise the current camera is replaced by the newly found one, generating a cut.

5. The execution restarts from step 1

3.4 Optimisation Module

The optimisation module within the CamOn controller is responsible of finding and tracking the camera configuration that optimises the given set of frame constraints. The optimisation is executed in real-time while the virtual environment changes; therefore the module needs to search an optimal solution of a dynamic optimisation problem.

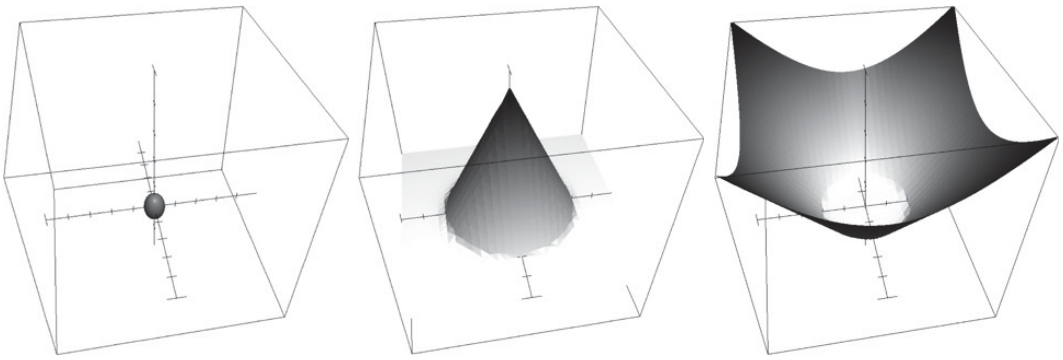
We propose a novel approach, based on an hybrid meta-heuristic, to the problem of finding and tracking the optimal camera in real-time in a dynamic environment. The proposed search algorithm, combining an Artificial Potential Field and a Genetic Algorithm, stands upon the author's earlier work on camera control (Burelli and Jhala, 2009b; Burelli and Yannakakis, 2010a) and extends it by introducing a novel coupling mechanism of the two algorithms. The proposed combination of local search and population-based search permits the exploitation of the speed of convergence of the first while maintaining the reliability and the ability to diversify the solutions of the latter.

The reminder of the section will describe a set of algorithms for virtual camera composition and it will present the hybrid Genetic Algorithm used to optimise the virtual camera composition objective function in CamOn. The algorithms that will be described alongside our solution will be: Artificial Potential Field, Sliding Octree, Genetic Algorith, Particle Swarm Optimisation and Differential Evolution. These algorithms stand as state-of-the-art optimisation for virtual camera composition and they are part of the comparative study presented in Chapter 6.

3.4.1 Artificial Potential Field

The first algorithm presented in this section is the Artificial Potential Field (APF) (Khatib, 1986). APF is an iterative technique commonly adopted in the area of robotics for controlling the navigation of robots in dynamic environments. Robots are modelled as particles moving in a field of potentials attracted by low potential areas; the position to be reached generates an attractive force (a low potential zone) and obstacles generate repulsive forces (high potential zones). At each iteration the particle moves along the force resulting from the sum of all repulsive (obstacle avoidance) and attractive (goals) forces influencing the particle's current position; the particle continues to move until it reaches a stable state.

In automatic camera control, APF has initially been employed for simple navigation tasks (Drucker and Zeltzer, 1994; Beckhaus et al., 2000), while we have adapted it to solve the full camera optimisation problem. In its application to camera optimisation, obstacle avoidance is modelled using repulsive forces and frame composition is obtained by



(a) Target object (b) Position potential field (c) Orientation potential field

Figure 3.7: Position (b) and orientation (c) potential fields produced by a visibility constraint on a sphere (a). The two three-dimensional potential fields are sampled along the XZ plane on at the Y coordinate of the sphere.

translating frame constraints into forces affecting both the position and the look-at point of the camera.

Each frame constraint produces one force attracting or repulsing the camera's position and one force attracting or repulsing the camera's look-at point; the system treats these two aspects of the camera as two different particles moving into two different potential fields. An example of the two potential fields created by a frame constraint (the target sphere has to be fully visible in the projected frame) can be seen in Figure 3.7. The two potential fields shown are a sample of the 3D field measured along the horizontal plane passing through the sphere centre. The particle representing the look-at point of the camera and the one representing its position are attracted by the low potential areas; resulting in the camera turning towards the target sphere and moving at a sufficient distance to be able to frame the complete object.

The *Vantage Angle* constraint produces one force $\vec{F}_{p\theta}$ that affects the camera position defined as follows:

$$\begin{aligned} \vec{F}_{p\theta} &= (1 - f_\theta)(\vec{c}\hat{} - \vec{t}) \\ \vec{t} &= \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & -\sin(\beta) \\ 0 & \sin(\beta) & \cos(\beta) \end{bmatrix} \times \vec{fw}_c |\vec{c} - \vec{s}| \end{aligned} \tag{3.7}$$

where $\vec{c}$ is the camera position, $\vec{s}$ is the subject position, $\vec{fw}_c$ is the subject's front vector and the f_θ function is the vantage angle objective function defined in Equation 3.1.

The *Object Projection Size* constraint also produces only a force $\vec{F}_{p\sigma}$ that affects the

camera position defined as follows:

$$\begin{aligned} \vec{F}_{p\sigma} &= d_p(1 - f_\sigma)(\vec{c} \hat{-} \vec{s}) \\ d_p &= \begin{cases} 1 & if\ \sigma_c < \sigma_d \\ -1 & otherwise \end{cases} \end{aligned} \tag{3.8}$$

where σ_c is the current projections size value and σ_d the desired one, $\vec{c}$ is the camera position, $\vec{s}$ is the subject position and the f_γ function is the projection size objective function defined in Equation 3.2.

The *Object Visibility* constraint produces one force that affects the camera position and one force that affects the camera look-at point. The camera position force $\vec{F}_{p\gamma}$ is defined as follows:

$$\begin{aligned} \vec{F}_{p\gamma} &= (v\vec{u}_w + h\vec{r}_c)|\vec{c} - \vec{s}|(1 - f_\gamma) \\ v &= \begin{cases} d_p & if\ occ(\vec{e}_{bottom}) \wedge \overline{occ(\vec{e}_{top})} \\ -d_p & if\ occ(\vec{e}_{top}) \wedge \overline{occ(\vec{e}_{bottom})} \\ 0 & otherwise \end{cases} \\ h &= \begin{cases} d_p & if\ occ(\vec{e}_{left}) \wedge \overline{occ(\vec{e}_{right})} \\ -d_p & if\ occ(\vec{e}_{right}) \wedge \overline{occ(\vec{e}_{left})} \\ 0 & otherwise \end{cases} \\ d_p &= \begin{cases} 1 & if\ \gamma_c < \gamma_d \\ -1 & otherwise \end{cases} \end{aligned} \tag{3.9}$$

where γ_c is the current visible fraction value and γ_d the desired one, $\vec{c}$ is the camera position, $\vec{s}$ is the subject position, $\vec{u}_w$ is the world's up vector and $\vec{r}_c$ is the camera's right vector. The $\vec{e}$ vectors, the f_γ function and the $occ()$ function are the ones defined in equation 3.3. The camera orientation force $\vec{F}_{o\gamma}$ is defined as follows:

$$\begin{aligned} \vec{F}_{o\gamma} &= d_o(\vec{s} - \vec{c} - \vec{fw}_c((\vec{s} - \vec{c}) \cdot \vec{fw}_c))|i - \gamma_d| \\ d_o &= \begin{cases} 1 & if\ i < \gamma_d \\ -1 & otherwise \end{cases} \\ i &= \frac{\sum_{i=1}^{N} infov(\vec{v}_i)}{N} \end{aligned} \tag{3.10}$$

where γ_d the desired visible fraction, $\vec{v}_i$ is the position of the i^{th} vertex of the subject's mesh, N is the number of vertices of the mesh, function $infov(\vec{x})$ calculates whether a point is included in the field of view or not, $\vec{c}$ is the camera position, $\vec{s}$ is the subject position and $\vec{fw}_c$ is the camera's front vector.

The *Object Frame Position* constraint produces one force $\vec{F}_{p\rho}$ that affects the camera

look-at point an it is defined as follows:

$$\begin{aligned} \vec{F}_{o\rho} &= (1 - f_\rho)(v\vec{u}_w + h\vec{r}_c) \\ v &= \begin{cases} 1 & if\ \vec{\rho}_{dy} > \vec{\rho}_{cy} \\ -1 & otherwise \end{cases} \\ h &= \begin{cases} 1 & if\ \vec{\rho}_{dx} < \vec{\rho}_{cx} \\ -1 & otherwise \end{cases} \end{aligned} \tag{3.11}$$

where ρ_c is the current on screen position value and ρ_d the desired one, $\vec{u}_w$ is the world's up vector, $\vec{r}_c$ is the camera's right vector and the f_ρ function is the ones defined in equation 3.4.

The *Camera Position* constraint produces one force $\vec{F}_{p\epsilon}$ that affects the camera position defined as follows:

$$\vec{F}_{p\epsilon} = (1 - f_\epsilon)(\vec{\epsilon} \hat{-} \vec{c}) \tag{3.12}$$

where $\vec{c}$ is the current camera position, $\vec{\epsilon}$ is the desired camera position and the f_ϵ function is the camera position objective function defined in Equation 3.5.

The force value at each point is described as a linear combination of scalar weighted forces, where each force corresponds to a frame constraint and each scalar value to the relative subject importance; the resulting values define the gradients of the potential field. An Euler method is used to integrate the resulting potential curve. The initial iteration step s starts from an arbitrary initial value and, during the optimisation, decreases according to the following formula:

$$s_i = s_0 \frac{e^{10(1-F_{i-1})} - 1}{e^{10(1-F_{i-1})} + 1} \tag{3.13}$$

where s_0 is the initial step, F_i is the best fitness value before iteration i and s_i is the step length that should be at the applied at iteration i. Note that F_{i-1} can assume values between 0 and 1, thus, the length reduction is also constrained in the same interval. Such a fitness-based decrease is intended to speed-up the convergence when the objective function value of the current solution is low and to stabilise the convergence when the solution is near optimal.

3.4.2 Sliding Octree

In the second version of their camera control system, Bourne et al. (2008) introduced a local search algorithm, the Sliding Octree (SO), that explores the search space through cubic spaces that get smaller at each iteration step.

SO is a best-first search algorithm, following the node of the octree with the highest fitness value. The search algorithm starts by generating an octree that contains 8 points

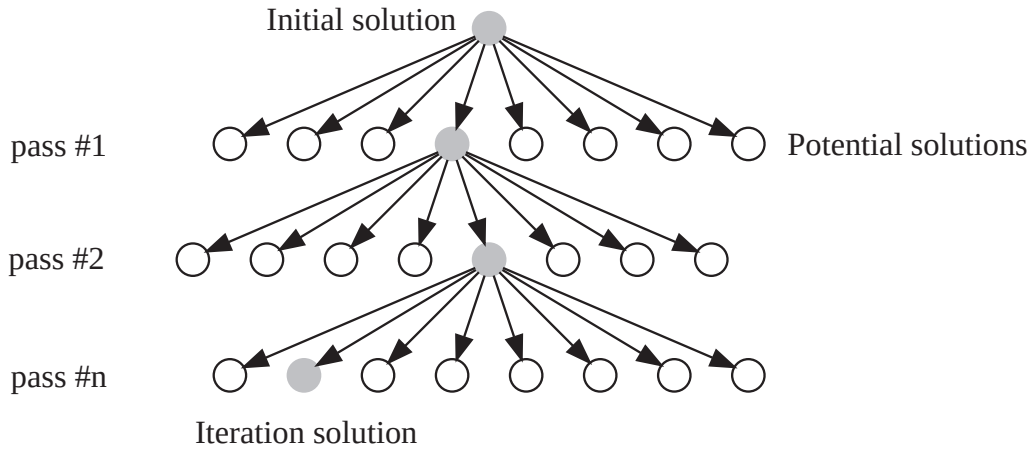


Figure 3.8: Example of a Sliding Octree iteration step. At each pass of the iteration the branch of the octree containing the best solution is explored, the distance between the children nodes and the parent node decreases by 25% at each level of the tree.

surrounding the current camera position. The distance between the points is equivalent to the maximum distance the camera can move during one algorithm iteration. During each iteration the algorithm performs a number of passes in which the octree slides towards the point having the highest evaluated fitness value and a new evaluation of the octree nodes is performed. At each pass, the distance of the octree points is decreased by a linear scaling factor, so that the search process starts coarsely and becomes increasingly focused around the solution. After all the passes have been performed a new octree is generated around the new solution and the execution proceeds with the next iteration.

The distance scaling factor and the number of passes are the parameters of the algorithm. If the scaling factor is too large, the octree shrinks too quickly and large portions of the search space may be missed. If the scaling factor is too small, the solver takes significantly longer to converge on the optimal solution. Bourne at al. suggest a linear scaling factor a_s equal to 0.75 and a number of passes N_p calculated as follows:

$$N_p = 3D_s \tag{3.14}$$

where D_s is the number of dimensions of the search space.

The version of the optimisation algorithm evaluated in Chapter 6 differs slightly from this implementation due to the dimensionality of the search space. The original version of the search algorithm optimised only the camera position: therefore, it searched for a solution only in a three-dimensional space. To apply the same heuristic to the five-dimensional search space explored by the CamOn optimisation module, the number of nodes at each level of the tree is 32 (2^5) instead of 8 (2^3).

3.4.3 Genetic Algorithm

A Genetic Algorithm (GA) (Holland, 1992) is a biologically-inspired population-based metaheuristic that mimics the process of evolution. It operates by generating a population of random individuals in the search space, each encoding a possible solution. At each iteration of the algorithm, each individual of the population is evaluated against the function that needs to be optimised (objective function). Then a new population of individuals is generated from the application of genetic operators such as crossover and mutation on a selection of individuals belonging to the previous generation.

Despite the vast variation of GAs used in the literature, there are some common basic algorithmic steps which are described as follows:

1. A population of chromosomes (i.e. individuals or solutions) is randomly initialised.
2. Each chromosome is evaluated and its fitness value is calculated as the objective function value corresponding to the solution described by the chromosome.
3. A subset of parents is selected according to their fitness. Several selection schemas che be applied such as elitism or roulette wheel.
4. Genetic operators (crossover and mutation) are applied to the previously selected subset of chromosomes and a new generation of potential solutions is generated.
5. The new solutions are evaluated and re inserted in the initial population by substituting other solutions according a replacement strategy.
6. The algorithm re-iterates starting from step 3.

In GAs, a crossover operator is a genetic operation that generates one or more new potential solutions from the combination of two parents. The chromosome of each new potential solution is the result of a recombination of two parent chromosomes. A mutation operator is a genetic operation that alters a potential solution by randomly changing the values contained in the selected chromosome. Both operators exist in different variations and can be applied to both binary and real valued solutions.

In the author's initial study on the combination of APF and GA for automatic camera control (Burelli and Yannakakis, 2010a), a GA was employed to optimise a reduced objective function built only on occlusion requirements. Occlusion depends only on camera position; therefore, by reducing the dimensionality of the search space, the computational cost of the GA was significantly lower. However, due to this solution, the GA often converged to undesirable areas of the search space with very low overall objective function value and did not allow the local search process to converge to a better solution.

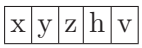


Figure 3.9: Chromosome of an individual of the Genetic Algorithm, containing 5 real values describing the camera position and orientation.

For the purpose of this thesis, a GA is employed to optimise the complete virtual camera composition objective function; therefore each individual represents a possible camera configuration. The chromosome of each individual contains 5 float values representing the camera position and orientation, using 3 Cartesian coordinates for the position and 2 spherical coordinates for the orientation, as displayed in Fig. 3.9. The crossover operator applied is a uniform crossover, while the mutation operator applies a uniform random mutation.

3.4.4 Differential Evolution

Differential Evolution (Storn and Price, 1997) is an evolutionary algorithm which, similarly to a GA, encodes the solutions as chromosomes belonging to a population of individuals which are iteratively selected and recombined during the optimisation process. DE, however, employs a particular selection and recombination scheme: for each individual in the population other three individuals are randomly picked and the new offspring is generated by the combination of these individuals with probability C for each value of the chromosome. The new values of the offspring y are given by the following equation:

$$y_i = \begin{cases} \alpha_i + W_d(\beta_i - \gamma_i) & \textit{if } r < C \\ x_i & \textit{otherwise.} \end{cases} \tag{3.15}$$

where W_d is the weighting factor, x_i is the i^{th} value of the current individual's chromosome, α, β and γ are the corresponding gene values of the other three randomly selected individuals, C is the crossover probability and r is a uniform random number in the range (0,1). DE has proven to be one of the best performing evolutionary algorithms on a large number of problems and demonstrated to be extremely robust to the parameters choice (Storn and Lampinen, 2004; Ali and Törn, 2004). The W_d parameter controls the magnitude of the recombination of the chromosomes, while the C parameter controls its probability.

In its implementation for virtual camera composition, DE encodes in each chromosome a potential five dimensional camera solution defined by the same real-valued parameters described for GA, as displayed in Fig. 3.9.

3.4.5 Particle Swarm Optimisation

Particle Swarm Optimisation (Kennedy and Eberhart, 1995) is a population-based global optimisation algorithm, whose dynamics are inspired by the social behaviour that underlies

the movements of a swarm of insects or a flock of birds. These movements are directed towards both the best solution to the optimisation problem found by each individual and the global best solution. The algorithm initially generates a population of random solutions encoded as particles. Each particle is then initialised with a random velocity. At each iteration n, after the particles moved at the speed they have been previously assigned, the algorithm evaluates the fitness corresponding to their new positions and recalculates their velocity according to the new global and local optima detected. The velocity v_i^n of the i^{th} particle at the n^{th} iteration is calculated according to the following formula:

$$v_i^n = lv_i^{n-1} + c_1 r_1 (o_i^{n-1} - x_i^{n-1}) + c_2 r_2 (o_g^{n-1} - x_i^{n-1}) \tag{3.16}$$

where r_1 and r_2 are two uniformly distributed random numbers in the [0, 1] range, whose purpose is to maintain population diversity. The constants c_1 and c_2 are, respectively, the cognitive and social parameters, which control the influence of the local and global optimum on the convergence of each particle. A high c_1 value favours exploration and diversity in the population, while a high c_2 value favours exploitation of the current known solutions increasing speed of convergence, but also increasing the possibility of early convergence.

The value l is the inertia weight and it establishes the influence of the search history on the current move. Finally, o_i^{n-1} is the best configuration assumed by the i^{th} particle at step $n-1$ and is known as the particle's local optimum, while o_g^{n-1} is the best configuration assumed by any particle at step $n-1$ and is the current global optimum.

Since at each iteration a solution to the problem is available (although it is not guaranteed to be the optimal one), PSO belongs to the family of *anytime* algorithms, which can be interrupted at any moment still providing a solution.

PSO has been first employed in camera control for off-line camera optimisation combined with a space pruning technique (Burelli et al., 2008). In the implementation included in this thesis, each particle moves in a 5-dimensional vector space defined by the same parameters of the camera described for GA in Section 3.4.3.

3.4.6 Hybrid Genetic Algorithm

The optimisation module employed in the CamOn camera controller is based on a hybrid Genetic Algorithm designed to exploit the speed of convergence of APF while maintaining the reliability and the ability to explore the solution space of a GA. Such a hybrid meta-heuristic extends and draws upon the author's earlier work integrating visibility optimisation via a GA combined with an APF based camera controller (Burelli and Yannakakis, 2010a). However, contrarily to the aforementioned approach, in which the two algorithms were executed as separate entities and the GA was only used to minimize occlusion, the algorithm

proposed in this article couples the two algorithms in a tighter fashion. The structure of the hybrid meta-heuristic algorithm follows the structure of a classic GA; however, a new unary operator is added to the classical crossover and mutation: an artificial potential field driven mutation. Such operator is added to exploit the knowledge of the objective function by employing an Artificial Potential Field optimisation algorithm based on an approximation of the composition objective function derivative.

As it is possible to observe in the flow chart depicted in Fig. 3.10, at every execution iteration the flow control passes to a component name *Scheduler*; this component is responsible to decide when to follow two types of policies: exploitation and exploration. When exploration is chosen the algorithm behaves exactly like a standard generational GA; therefore, it initialises the population up to the desired number of individuals and then applies the standard genetic operators. On the contrary, when the *Scheduler* chooses to activate exploitation, the APF based mutation operator is applied to the current optimal solution.

In the execution flow of the algorithm, the *Scheduler* is activated before any action is picked; therefore, the algorithm is able to exploit or explore also during the population initialisation phase. At the beginning of the algorithm execution, the *Scheduler* component follows the exploration policy once every P exploitations, where P is an integer parameter that defines how often the hybrid GA follows the exploitation policy. If the scheduler detects an early convergence, the ratio between the two policies execution is inverted; therefore, the *Scheduler* follows the exploitation policy only once every P explorations. Two conditions may trigger an early convergence: all the objects (that are requested to be visible) are are completely occluded, or the optimal solution's fitness does not increase for more then one frame rendering time. When all the aforementioned conditions cease, the *Scheduler* returns to the initial configuration.

3.5 Animation Module

This section describes the second component of the CamOn architecture; this component is responsible of finding a path between the current camera position and the new solution found by the optimisation module and to animate the camera. If no obstacle avoidance is required for camera motion, this component animates the camera position along a straight line between the current camera position and the next one. On the other hand, if the camera is required to avoid obstacles, the component calculates and updates a path that connects the current camera position with the target one at each frame. This path is computed using the Lazy Probabilistic Roadmap Method (Lazy PRM) path planning algorithm (Bohlin and Kavraki, 2000).

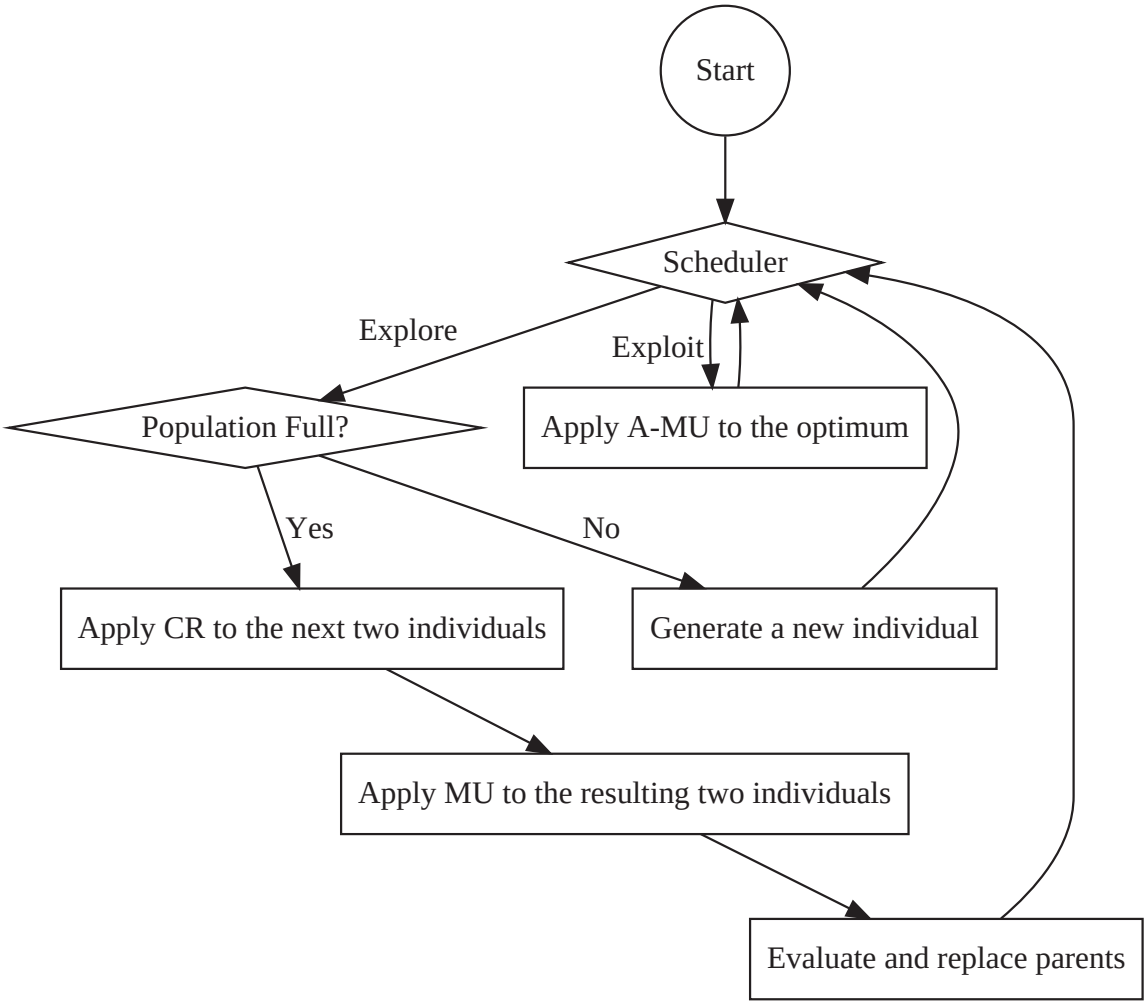


Figure 3.10: Flowchart representing the execution flow of the proposed hybrid GA. The labels CR, MU, A-MU are used to refer, respectively, to the crossover, mutation and APF base mutation operators

In both cases, the camera rotation is animated by linearly interpolating the current camera orientation with the target orientation. The speed movement along the calculated path and the rotation movement are controlled according to the animation constraints given as inputs to the camera controller. Under two conditions, the animation module does not follow the aforementioned procedure and positions and rotates the camera directly to the optimal solution. This happens if the path planning algorithm is unable to find valid path in the allowed computation time or if the camera is required to perform a cut for cinematographic reasons.

The animation component in the aforementioned architecture ensures control over camera animation independently of optimisation convergence and, moreover, effectively reduces

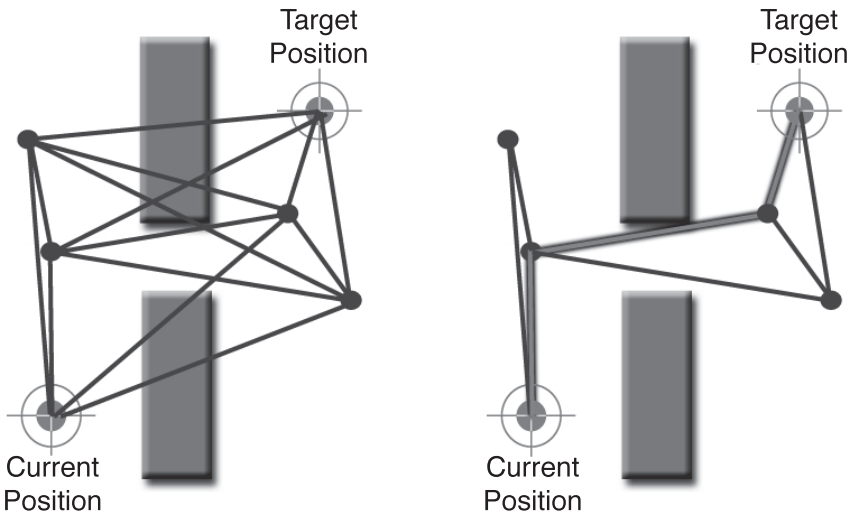
the camera jittering problems caused by the natural oscillation in the local search convergence process.

3.5.1 Lazy Probabilistic Roadmap

The path planning algorithm employed in CamOn is a Lazy Probabilistic Roadmap (Bohlin and Kavraki, 2000), which is an evolution of the original Probabilistic Roadmap Method introduced by Kavraki et al (1994) . In a standard PRM, the path planning process is separated in two phases: the learning and the query phase. In the first phase, the algorithm builds a probabilistic roadmap by randomly sampling the search space and storing the valid solutions as nodes of a fully connected graph (see Fig. 3.11a). Each edge of the graph is then checked for collisions and the ones for which a collision is detected are disconnected. This way, only a graph of the possible transitions is built (see Fig. 3.11b). In the second phase, a query asks for a path between two nodes; to process the query, any graph path finding algorithm can be used, for instance, a Dijkstra's algorithm (Dijkstra, 1959) can be used if the shortest path is required.

In a dynamic problem, such as camera control where the starting and target position change at each iteration, the two phases need to be recomputed continuously to be able to query for a path on a constantly updated probabilistic roadmap. However, the path needs to be re calculated at every frame; therefore the changes of the two nodes are generally very small and we can assume that a large part of the graph and the path computed at the previous iteration is still valid at the next one. For this reason, a Lazy PRM is used compute the path at each iteration. The main difference between this algorithm and its original form (Kavraki et al., 1994) is that, at each iteration of the learning phase, the roadmap is not reconstructed but it is updated by removing the nodes which have been already crossed and by adding a number of new nodes positioned towards the new target configuration. The collision checks are then performed only on the edges connecting the new nodes, thus, substantially reducing the computational cost of each iteration.

According to our implementation, a set of random (normally distributed) points is generated in the space between the two camera positions; the mean of the mixed-Gaussian distribution is located halfway between the two positions and its standard deviation equals half of their distance. All the generated points as well as the current camera position and the newly found solution define the nodes of a 3D graph which is fully interconnected. All the point-connectors which cross an object are removed and the shortest path between the current position and the goal position is calculated (see Fig. 3.11) and used to animate the camera. Since our implementation is based on Lazy PRM, the graph is not recomputed at each iteration and only a new node containing the new position is added to the graph,



(a) Graph generation (b) Shortest path

Figure 3.11: Probabilistic Roadmaps

thereby, reducing the computational cost of path planning and generating a more stable trajectory for the camera compared to the authors' previous solution that employed standard PRM (Burelli and Yannakakis, 2010a).

3.6 Summary

This chapter introduced the problems of virtual camera composition and virtual camera animation and described their characteristics from a numerical perspective. The concepts of frame and animation constraints are defined as well as their relationship with the camera parameters. Virtual camera composition has been defined as an optimisation problem and its objective function has been described in all its components. Such objective function is the linear combination of the objective functions corresponding to each frame constraint used to describe the controller's task. The objective functions of all constraints have also been mathematically defined.

The second part of the chapter described our approach on the composition and animation problems by initially introducing the CamOn camera controller architecture and later describing the algorithms used for the controller's implementation and evaluation. The CamOn controller addresses the key problems of camera composition and animation by combining advantages from a set of algorithms with different properties. In particular we combine a local search algorithm with a population-based algorithm to couple the computational efficiency of the first with the robustness of the second. Finally the solution found

through the combination of these algorithms is used as a target to guide a 3D path planning algorithm.

The hybrid optimisation algorithm proposed is a modified Genetic Algorithm with a new genetic operator based on an Artificial Potential Field algorithm. The path planning algorithm used in the animation module is a Lazy Probabilistic Roadmap Method. A set of other state-of-the-art optimisation algorithms, used for the optimisation module's performance evaluation (see Chapter 6), is also described in the chapter.

Chapter 4

Adaptive Camera Control

This chapter[1] describes a methodology to model the relationship between camera placement and playing behaviour in games for the purpose of building a user model of the camera behaviour that can be used to control camera movements based on player preferences.

An hypothetical optimal automatic camera control system should provide the right tool to allow designers to place the camera effectively in dynamic and unpredictable environments; however, a limit of this definition is that it excludes the player from the control loop. Tomlinson et al., in their paper on expressive autonomous cinematography (Tomlinson et al., 2000), quote a statement by Steven Drucker at SIGGRAPH '99: "It was great! I didn't notice it!". Drucker was commenting Tomlinson's work presented at SIGGRAPH that year and, in his comment, he clearly associates the quality of a camera control system with the lack of intrusiveness; however, how do we achieve such a result or how is it possible to take the control of the camera from the player but still moving the camera the way the user would have wanted (or as close as possible) are open research questions. We believe that, to bridge the gap between automatic and manual camera control the camera objective should be affected by the player.

To achieve this goal we propose a new approach to automatic camera control that indirectly includes the player in the camera control loop. In our view, the camera control system should be able to learn camera preferences from the user and adapt the camera profile to improve the player experience.

For this purpose we investigate player preferences concerning virtual camera placement and animation, and we propose a methodology to model the relationship between camera behaviour, player behaviour and game-play. Camera behaviour is modelled using the combination of gaze and camera position at each frame. Combining gaze data with camera data allows a finer analysis of the player's visual behaviour permitting, not only to understand what objects are visualised by the player, but also which ones are actually observed. This

[1] Parts of this chapter have been published in (Burelli and Yannakakis, 2011; Picardi et al., 2011)

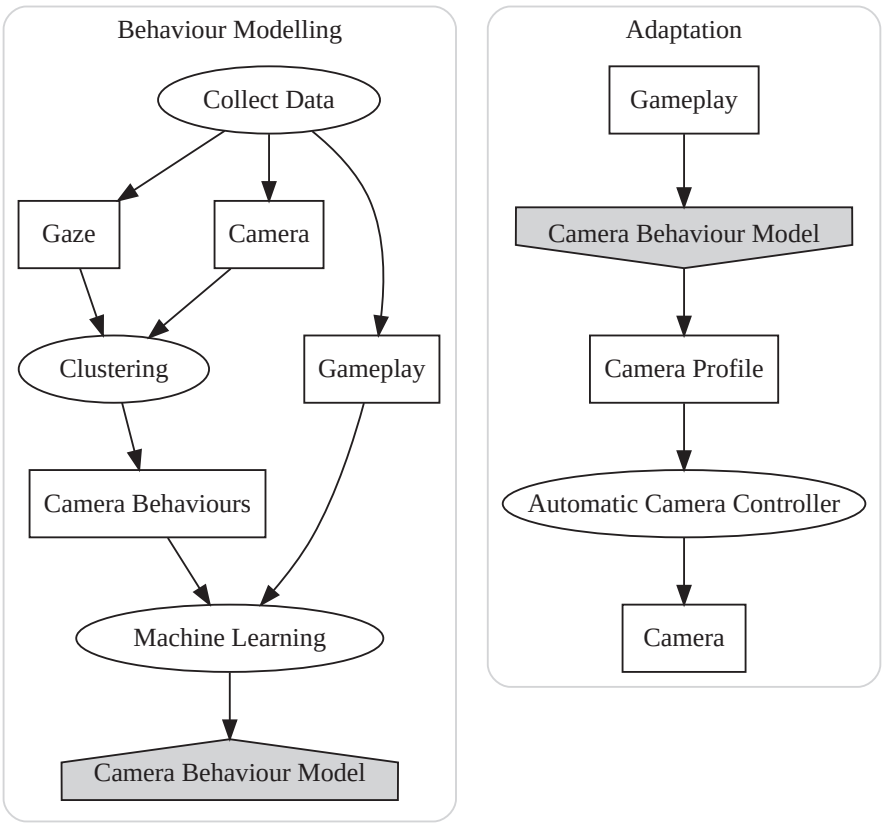


Figure 4.1: Camera behaviour modelling and adaptation phases of the adaptive camera control methodology. The first phase leads to the construction of a camera behaviour model, which is used in the adaptation phase to drive the automatic camera controller and generate the adaptive camera behaviour.

information permits to filter exactly which object is relevant for the player among the ones visualised by the player through her control of the virtual camera. A cluster analysis of the gaze data collected is run to investigate the existence of different virtual camera motion patterns among different players and different areas of the game.

Moreover, the relationship between game-play and camera behaviour is modelled using machine learning. Artificial Neural Networks (ANNs) are employed to build predictive models of the virtual camera behaviour on player behaviour in earlier stages of the game. These models are used to drive the automatic camera controller and provide a personalised camera behaviour.

The two aforementioned phases of the methodology can be seen in Figure 4.1: the step of the first phase is required to build the camera behaviour model for a game (left diagram) which is later used to detect the most appropriate camera behaviour fro the player when the game is played (right diagram).

Figure 4.2: A screen shot of a 3D platform game in which the objects observed by the player are highlighted by green circles.

The remainder of the chapter describes the different techniques used to build the models and the adaptation mechanism. Chapter 8 presents an experiment demonstrating the effectiveness of the methodology on a 3D platform game.

4.1 Camera Behaviour

Camera behaviour can be modelled directly using the data about camera position relative to the avatar. However, this approach would fail revealing which objects the player wants to watch during play. A better approximation would be achieved by analysing the objects present on screen through each area. The presence of a certain object on the screen, however, does not necessarily imply an intentionality of the player; e.g. the object might be on the screen only because it is close to an object the player is interested to. The gaze data available permits to overcome this limitation since, using the gaze position, it is possible to understand which object is actually observed among the ones framed by the player. Therefore, to model the camera behaviour, we combine camera movements and gaze coordinates to identify the objects observed by the player at each frame. Figure 4.2 shows an example of how gaze information con help filtering the objects framed by the camera, identifying the highlighted objects as the only ones receiving visual attention from the player.

In addition to the gaze information, a number of features regarding the dynamic camera behaviour are required, such as the average camera speed or the maximum and minimum acceleration. All this information combined is used to build a model describing how the player moves the camera in a specific game under different game conditions.

To generate the camera behaviour model for a specific game, we propose to collect the aforementioned data regarding the players' camera behaviour while playing the game using a manual camera controller (e.g. during an early stage of the game testing) as seen in Figure 4.3. Subsequently, following the approach outlined by Mahlmann et al. in (2010) we propose to mine the logged data to identify a set of prototypical behaviours and then use machine learning to build a model connecting the player in-game behaviour with the camera behaviour types. This model is used to adapt the camera behaviour during the actual game play based on the player's in-game behaviour.

4.1.1 Gaze

Eye movements can be recognised and categorised according to speed, duration and direction (Yarbus, 1967). In this work, we focus on *fixations*, *saccades* and *smooth pursuits*. These movements, among other, describe how a person looks at a scene and how her visual attention is distributed between the different objects visualised. A consistent pattern of eye movements, called "saccade and fixate" strategy can be found in all humans; a saccade occurs when a person is rapidly switching her attention from one point to another and a fixation is an eye movement that occurs when a subject focuses at a static object. If the object observed is in movement the fixation movement is replaced by a smooth pursuit. According to (Land and Tatler, 2009) There are two reasons for those different movements: first, the area of the eye with maximum acuity of vision, the fovea, covers only one in four-thousandth of the retinal surface, so the eye needs to move to place the desired object's projected image on this area. Second, gaze must remain still on the target object for the photo-reception process to acquire an unblurred image.

During saccadic movements the eye is effectively blind due to the motion blur and active suppression; therefore, to model the player's visual attention, we will focus on fixations and smooth pursuit movements. A fourth movement that Land and Tatler (2009) identify in the human eye movements repertoire is Vergence; however it is not considered in this methodology as it occurs only when looking at objects at different distances from the viewer. This kind of movement does not occur while looking at a virtual environment on a scree, as even objects at different virtual distances have the same physical distance to the eyes.

The detailed characteristics of the aforementioned eye movements are as follows:

Saccade Saccades are fast, stereotyped, jump-like movements of the eyes with speeds up to $900°/s$. Saccades can be motivated by external stimuli, such as an object appearing in the field of view, or by internally generated instructions. These movements occur before and after a smooth pursuit or a fixation movement to relocated the visual focus.

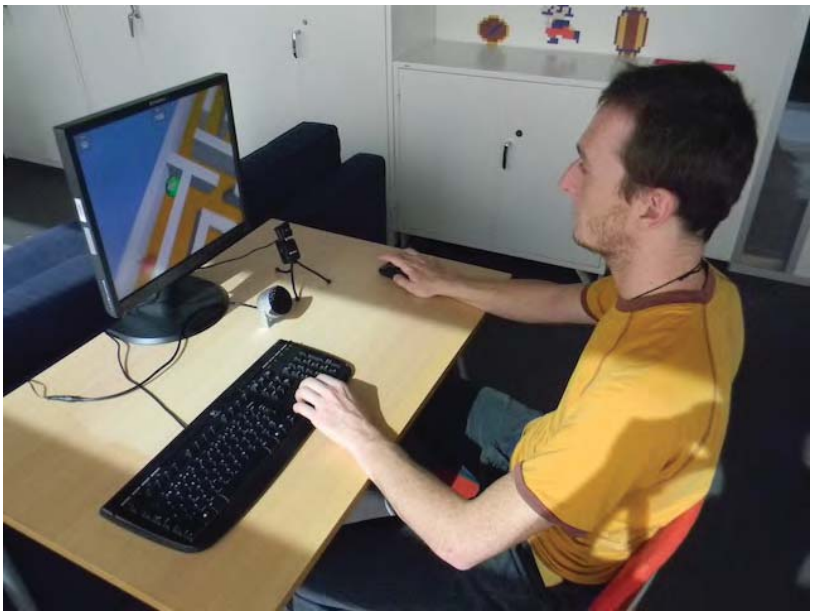


Figure 4.3: Example of a data collection setup. The player plays a game and manually controls the virtual camera. The game keeps track of the virtual camera placement, the player behaviour and the gaze position on the screen; gaze position is recorded using a gaze tracking device. The software used in the setup portrayed in the figure is the ITU Gaze Tracker (http://www.gazegroup.org), and the sensor is an infra-red camera.

Fixation Fixations are pauses in the eye scanning process over informative regions of interest. During these pauses the eye is able to collect visual information about the fixation target area. They are usually intervals between two consecutive saccadic movements and they usually have a duration of at least 100 ms. Longer fixation duration implies more time spent on interpreting, processing or associating a target with its internalized representation. During a Fixation eyes make small jittery motions, generally covering less than 1-2 degrees. This measure is called fixation dispersion and it is one of the measures usually calculated in order to classify a sequence of gaze records as a fixation.

Smooth pursuit Smooth pursuits are a class of movements similar to fixations and they can be observed when the gaze follows a moving object. During a smooth pursuit the velocity will be similar to the target's speed, up to $15°/s$. Above this velocity the smooth pursuit will be interrupted by catch-up saccades, as the eye is unable to follow the target.

We represent the virtual camera behaviour as the amount of time the player spends framing and observing different objects in the game environment while playing the game. This representation of behaviour is chosen over a direct model of the camera position and

motion as it describes the behaviour in terms of the content visualised by the camera and, therefore, it is independent of the absolute position of the avatar, the camera and other objects. The features related to the time spent observing objects are calculated as the sum of the durations of the smooth pursuit and fixation movements of the eyes during which the gaze position falls within an object's projected image.

The gaze and camera behaviour described by these features is game dependent; therefore a different model should be built for each game. To collect the data needed to build such a model, the participants of the data collection experiment should play the game manually controlling the camera (using either the mouse or a joypad) while a logging system records the three-dimensional virtual camera position, the gaze position on screen and the in-game events.

The gaze position on screen can be recorded using a gaze tracking system; these systems usually employ one or more cameras tracking the orientation of the eyes and the head position. Figure 4.3 shows an example of the setup needed during the data-collection phase: the experiment partecipant plays the game at a desk sitting on a chair with rigid back and no wheels (to reduce head movements), the gaze is tracked using an infra-red camera to detect and track eye features.

4.1.2 Behaviour Identification

To investigate and create a model of the relationship between camera behaviour and gameplay, we analyse the collected data through two steps: identification and prediction. In the first step we use a clustering technique to extract the relevant camera behaviours and analyse their characteristics and then, in the second step, we build a model based on this categorisation able to predict the correct behaviour given a set of game-play data.

The number of distinct camera behaviours as well as their internal characteristics can only be based, in part, on domain knowledge. One can infer camera behaviour profiles inspired by a theoretical framework of virtual cinematography (Jhala and Young, 2005) or alternatively follow an empirical approach — as the one suggested here — to derive camera behaviour profiles directly from data. The few existing frameworks focus primarily on story-driven experiences with little or no interaction, thus are not applicable in our context. Therefore, we adopt a data-driven approach and we employ clustering on the gaze-based extracted features for the purpose of retrieving the number and type of different camera behaviours.

Before clustering can be applied to the data, the data has to be divided in records describing each a single triple of player behaviour, gameplay and camera behaviour. Depending on the types of game and the length of the recorded experience the records might

vary in temporal and spacial length. The observation times describing the gaze behaviour should be normalised for each record depending on the records length.

The granularity of the subdivision in records depends on the game characteristics and three different criteria can guide this process: time, space and task. According to task criterion, the logged data for each player is divided in records representing the experience for each task completed by the player. While this solution is probably the most appropriate, as it is based on the game characteristics, it is often inapplicable as some part of the game might have unclear tasks or multiple ones active at the same time. The space criterion is simpler to apply as in many games there are well defined areas, e.g. *Devil May Cry* (Capcom, 2001) or *Halo* (Microsoft, 2001); therefore, it is easy to identify the beginning and the end of each record. In case neither of the previous criteria can be applied, the overall experience can be divided in time chunks of variable length depending on the game characteristics.

After having divided the data in records, a k-means algorithm (MacQueen, 1967) applied on the camera behaviour data can be used to identify the types of behaviour represented by the collected data. K-means is a simple and well known algorithm for data clustering. Each record can be thought of as being represented by some feature vector in an n-dimensional space, n being the number of all features used to describe the objects to cluster. The algorithm then randomly chooses k points in that vector space; these points serve as the initial centroids of the clusters. Afterwards, each record is assigned to the centroid they are closest to. Usually the distance measure is chosen by the user and determined by the learning task. After that, a new centroid is computed for each cluster by averaging the feature vectors of all records assigned to it. The process of assigning records and recomputing centroids is repeated until the process converges.

K-means requires initial knowledge of the number of clusters k existent in the data to minimise the intra-cluster variance. To overcome this limitation, the algorithm can be run with progressively higher k values and the clusters generated at each run are evaluated using a set of cluster validity indexes such as the Davies-Bouldin (1979), the Krzanowski-Lai (1988) or the Calinski-Harabasz (1974). The number of clusters can be selected using a majority voting mechanism; the algorithm runs for a number of times for each k and the run with the smallest within cluster sum of squared errors is picked.

4.1.3 Prediction

Once the camera behaviour patterns are identified, we proceed by modeling the relationship between game-play and camera behaviour types. More precisely, since the model is intended to select the most appropriate camera behaviour that fits the player's preferences in the

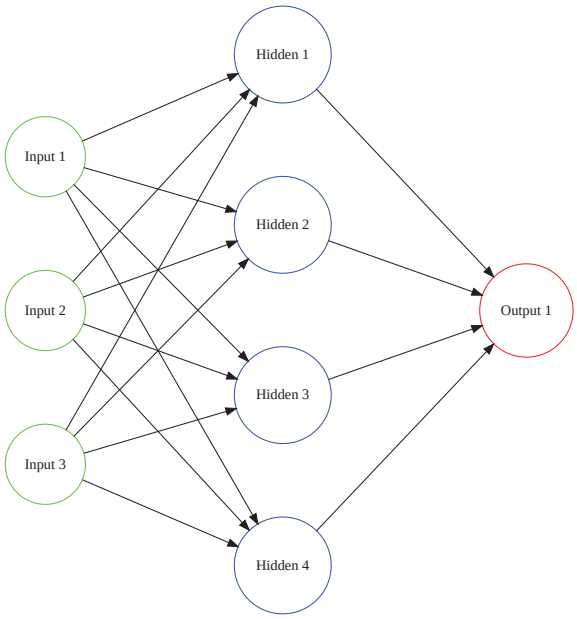


Figure 4.4: An example of a fully connected feed-forward artificial neural network. Starting from the inputs, all the neurons are connected to all neurons in the subsequent layer.

game, we attempt to approximate the function that maps the game-play behaviour of the player to the camera behaviour. For this purpose, we use Artificial Neural Networks (ANNs) which are chosen as a function known for its universal approximation capacity.

An ANN is a bio-inspired data structure used to model arbitrary non-linear mappings between inputs and outputs, typically used for pattern recognition and association (Haykin, 2008). The computation is performed with a number of neurons, the neurons are conneted in a graph, and typically propagate signals through the network (see Figure 4.4). The signal's path through the network is controlled by both the structure of the network and the weights, w, that are placed at each connection between neurons. Each neuron in the network transfers the input values to its output according to a transfer function f_i^T defined as follows:

$$y_i = f_i^T(\sum_J w_{ji}x_{ji} + \theta_i) \tag{4.1}$$

where y_i is the output of the i^{th} neuron, x_{ji} is the j^{th} input of the i^{th} neuron, w_{ji} is the weight of the j^{th} input of the i^{th} neuron, and θ_i is the bias of the i^{th} neuron.

For the network to produce the desired association between its inputs and its outputs it is necessary to find the configuration of size (number of nodes and layers) and weights of the connections that generate the desired function. The process of finding these parameters is called learning; depending on the application of the network and the knowledge of the function to be approximated three types of learning can be performed: supervised, reinforcment

and unsupervised. For the purpose of this thesis we will focus only on supervised learning; however, the reader is advised to refer to (Haykin, 2008) for fundamentals on ANNs and other learning methods.

In supervised learning, a set of valid input-output pairs, called training set, are used to train the network. At each training iteration, an input vector is presented to the ANN along with a corresponding vector of desired (or target) outputs, one for each neuron, at the output layer. The discrepancies between the desired and actual response for each output node is calculated and this error is used to correct the weight of the network according to a learning rule.

One of the most common learning rules for training a neural network is back-propagation (Rumelhart et al., 1986). This algorithm is a gradient descent method of training in which gradient information is used to modify the network weights to minimise the error function value on subsequent tests of the inputs. Using back-propagation, it is possible for a feed-forward artificial neural network to learn the association between the inputs and the outputs in the training set; moreover, if coupled with techniques such as early stopping (Caruana et al., 2001), the resulting network will be able to generalise the association also to previously unseen inputs.

ANNs and back-propagation can be used to learn the connection between the player in-game behaviour and its camera behaviour; however, to be able to adapt the camera behaviour, the network needs to be trained to predict the future player choice over camera instead, so that the automatic camera controller can be instructed to generate the desired camera behaviour exactly at the moment the behaviour is needed.

For this purpose, the ANN should be trained to learn the association between the camera behaviour cluster at at the i^{th} record and the player's in-game behaviour in the previously recorder records (from 0 to $i-1$). Moreover, if the game includes different types of challenges, the ANN should consider also the characteristics of the gameplay at record i (see Figure 4.5).

4.2 Adaptation

Using the model of camera behaviour built in the data collection experiment it is possible to predict the player's preferences on camera during the game and use this information to instruct an automatic camera controller and personalise the cinematographic experience. When the player performs a switch from one chunk (area, task or time frame, depending on the subdivision criterion) to the next one, the prediction of the neural network can be used to decide which camera profile should be assigned to instruct the camera controller for the next chunk of the game.

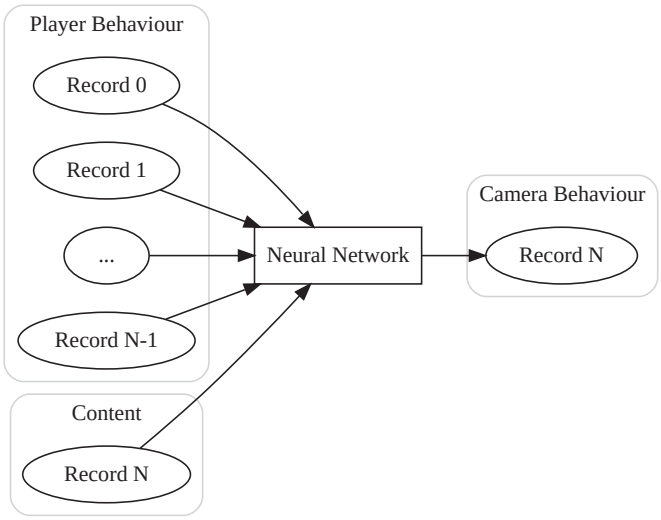


Figure 4.5: Camera behaviour prediction scheme. The neural network is presented with the features describing the gameplay characteristics of the next record and the features about the previous player behaviour as input and returns the next predicted camera behaviour for the next record as the output.

The same input-output scheme depicted in Figure 4.5, used for training, should be used for prediction. The computational model is able to predict the camera behaviour given information about the way the player played up to a certain point of the game. Therefore, before the right camera profile can be selected for the player it is necessary that she plays at least one area for each area type. A variable number of previously played game chunks might be necessary to achieve the best accuracy of prediction. The length of the past tracks depends on the characteristics of the game, therefore, it is necessary to find the best configuration trough en empirical test at the time of training.

Once the next camera behaviour is detected, a camera profile — i.e. a set of frame and motion constraints — should be generated to instruct the automatic camera controller. This translation process can be approached in two ways: either a designer assigns a custom designed camera profile to each behaviour identified by the clustering algorithm — which is then automatically picked by the adaptation mechanism — or the profile is generated completely based on the selected behaviour's characteristics.

Independently of the approach chosen, the translation process between gaze based camera behaviours and camera profiles is hardly generalisable over different games as the number and the quality of the objects present on screen varies considerably. In most action games — e.g. *Tomb Raider* (Eidos Interactive, 1996) or *Super Mario 64* (Nintendo, 1996) — the camera can be instructed to follow the main avatar and maintain the visibility of the objects which have received visual attention in the camera behaviour. The weights of the frame

constraints imposed on each object can be related to the amount of time spent observing the objects of the same kind as this information is related to the amount of attention that and objects receives.

Such an approach can be applied to virtually any game which features an avatar and a third person camera and it is the one used in the test case presented in Chapter 8 for a three-dimensional platform game. The constraints imposed on the avatar and on the other objects included in the behaviour can be changed to alter the overall composition. For very different game genres, such as strategy games or racing games, different strategies can be developed to transform the behaviours in camera profiles.

4.3 Summary

This chapter presented a methodology to model the relationship between camera placement and playing behaviour in games and to generate adaptive camera behaviours. The methodology combines player's gaze, gameplay and virtual camera data to build a model of camera behaviour based on the player's in-game behaviour. The data collected is clustered using k-means to identify relevant behaviours and the relationship between the clusters and the game-play experience is modelled using three ANNs. Finally an adaptation mechanism based on the aforementioned model is presented including a method for translating the camera behaviour clusters into camera profiles suitable for an automatic camera controller.

An application of the method described in this chapter and its evaluation is presented in Chapter 8.

Chapter 5

Test-bed Environments

This chapter[1] presents the virtual environment developed to evaluate the algorithms and methodologies presented in this thesis. The test environments will be described in detail and their choice will be motivated in terms of completeness and generality.

Two test environments are presented in this chapter with two different purposes. The first, a virtual camera sandbox, is an application developed to test the numerical performance of optimisation algorithms and to evaluate the characteristics of the camera composition problem. The sandbox will be used in Chapter 6 to test the algorithms employed in CamOn and to analyse the complexity of different camera composition problems. The second test environment is a 3D computer game developed to test the proposed camera controller and camera adaptation method against human players and to evaluate its effect on player experience. On this basis, this game will be used both in the experiments presented in Chapter 7 and Chapter 8.

5.1 Virtual Camera Sandbox

The first test-bed environment presented in this thesis is a development sandbox featuring a three-dimensional virtual environment and a series of virtual characters with whom the environment can be populated to yield camera control problems.

The purpose of such a tool is to stand as an abstraction layer between the user and the virtual environment allowing him to easily define camera composition problems and test their algorithms. The long term aim of the sandbox development is to create a flexible testing platform the contains a set of standard and replicable camera control problems usable as common benchmarks within the research community.

At the time this manuscript is written, the sandbox is still at an early stage of development; however, it offers already the possibility to build composition problems with any

[1] Parts of this chapter have been submitted for publication in (Burelli and Yannakakis, 2012b,a)

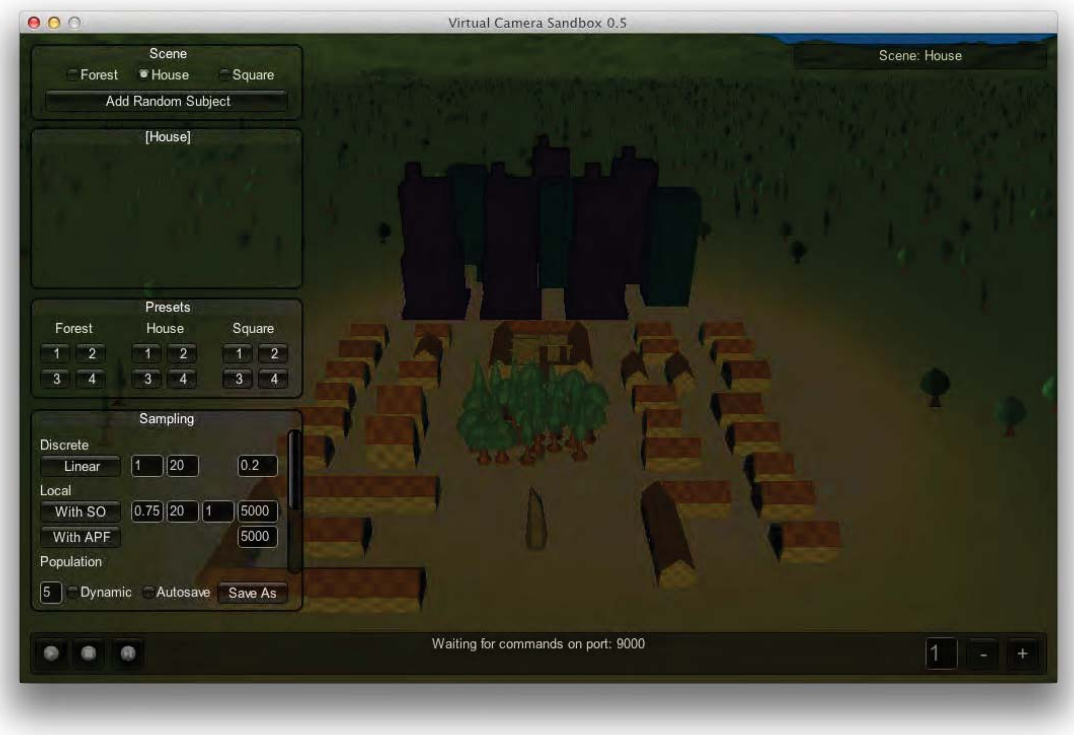


Figure 5.1: Interface of the virtual camera sandbox, showing the virtual environment and the interface elements.

number of subjects and frame constraints. The sandbox allows to test the behaviour of custom solving techniques on dynamic and static problems; moreover it allows to analyse the complexity of such problems.

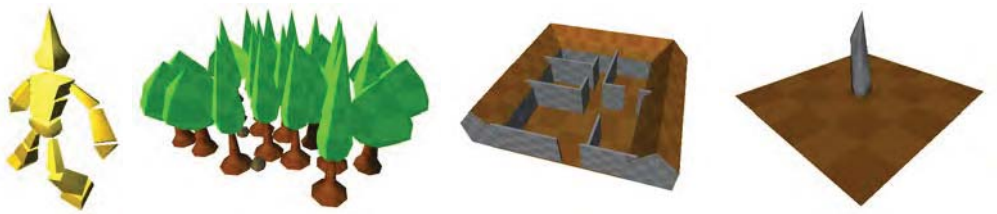
5.1.1 Functionalities

The virtual camera sandbox is a development tool designed to act as a testing platform for virtual camera control algorithms. For this purpose, the sandbox contains a virtual environment composed by a set of areas (see Fig. 5.1) with different geometrical characteristics. The user can place any number of characters in the environment and assign to each of them a set of frame constraints.

The interaction between the sandbox user and the sandbox can be described through the three following steps:

1. The user connects to the sandbox and selects the area in which to perform the experiment.

2. The user places a number of subjects in the environment and sets the frame constraints.



(a) (b) (c) (d)

Figure 5.2: Main components of the virtual camera sandbox. From left to right: the subject, the forest, the house and the square.

3. The user conducts the evaluation.

The sandbox supports four possible areas to pick: *forest* (see Fig. 5.2b), *city* (see Fig. 5.2d), *house* (see Fig. 5.2c) and *everywhere*. The three first options restrict the experiment to the respective area, the last option configures the sandbox in order to allow for sampling in any point of the virtual environment.

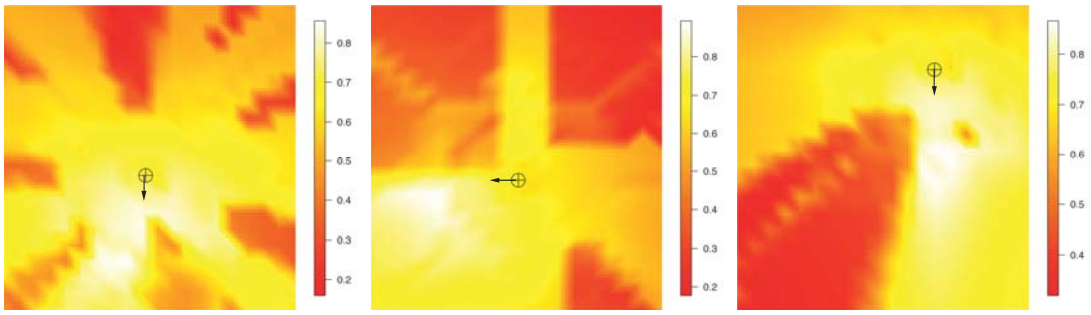
In the second step, a number of subjects can be placed anywhere within the previously selected area. After placement, each subject will be affected by the physical model of the virtual environment; therefore it will move until it reaches a stable condition. At this point, it will be possible to assign one or more frame constraints to the subject just placed in the environment. The sandbox supports all the constraints defined in Section 3.1, i.e. *Object Visibility*, *Object Projection Size*, *Object Frame Position* and *Vantage Angle*.

The evaluation step can be performed with two different ends: to evaluate an algorithm's behaviour or to analyse the problem's objective function. In the first case the sandbox can be used as a black box providing the objective function value of any camera configuration during the optimisation process. In the second case the sandbox provides several functions to sample the objective function, visualize it in the virtual environment and save it to a log file for further analysis.

The evaluation con be conducted on a static problem. In the first case the the subjects stand still in the position they have been placed initially. In the second case the subjects move in random directions turning at random times and avoiding the obstacles in the environment. The user can set the movement speed and can control the animation flow through the play, pause and next functions.

5.1.2 Environment Elements

The sandbox allows the user to place a set of subjects (see Fig. 5.2a) within the virtual environment to define the camera composition problem. These subjects are human-like 3D characters approximately 1.8 meters high (the measure is defined in proportion to the rest



(a) (b) (c)

Figure 5.3: Maximum value of the objective function sampled for each area of the sandbox across the X and Z axis. The position and orientation of each subject is identified by the black marks. The composition problem evaluated in each figure contains three frame constraints: an *Object Visibility* an *Objective Projection Size* and a *Vantage Angle*.

of the environment elements). Each subject can be placed anywhere in the environment and, when placed, it will lean on the ground in standing posture. The orientation of the subject can be selected by the user; however the subject can be rotated only along the vertical axis.

The virtual environment in which the subjects can be placed is a collection of three distinct environments: forest, house and square. Each environment features different characteristics and challenges in terms of camera composition and animation, and overall they represent a selection of typical game environments. This section will present the three areas and will give a short description of the characteristics of each area, using a Long Shot problem as a representative example, to motivate for their presence. An in depth analysis of the camera control problem in each area will be presented in Chapter 6.

The three test environments have been selected to provide the widest possible range of geometrical features commonly present in computer games and interactive virtual environments. Moreover, they have been designed to incorporate a wide variety of camera control challenges with different level of complexity. They include one indoor and two outdoor environments with different geometrical characteristics: a forest, an house and a square.

The forest environment (see Fig. 5.2b) is an outdoor virtual environment composed by a cluster of trees; the subjects are placed between these threes which act as partial occludes and scattered obstacles. As it is possible to observe from the sample landscape depicted in Fig. 5.3a, such environment influences the objective function landscape by increasing the search space modality; this is mostly due to the fact that the three trunks are thin occluders which produce a slicing effect in the objective function landscape.

The second environment, the house (see Fig. 5.2c), is an indoor environment with closed spaces separated by solid walls. As described in (Burelli and Yannakakis, 2010b), walls act

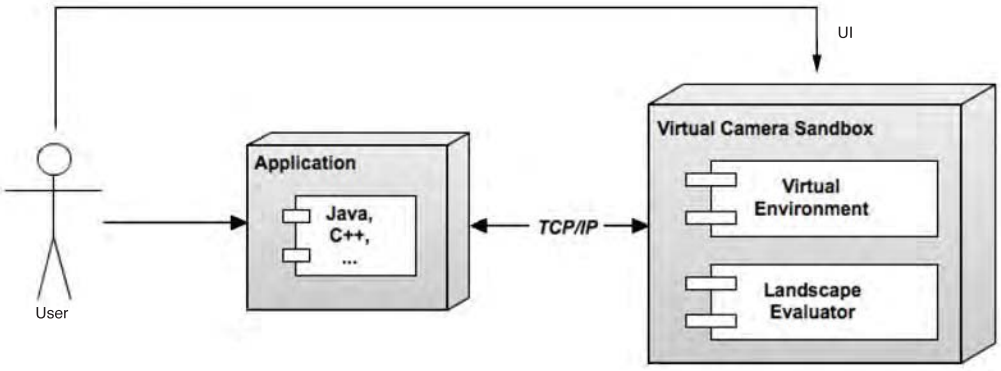


Figure 5.4: Virtual camera sandbox architecture.

as large occluders inducing large areas of the objective function landscape to have little or no visibility gradient, as visible in Fig. 5.3b.

The last environment, the square (see Fig. 5.2d), is the simplest one from a geometrical perspective. It is largely an empty space with one single obstacle placed in the center. This environment is expected to be the simplest in terms of optimisation complexity, as the lack of obstacles produced mostly landscapes with smooth gradients and a small number of modalities. Figure 5.3c illustrates an example of such a smooth objective function landscape.

5.1.3 Architecture

The sandbox is a stand-alone application developed using the Unity 3D game engine. It contains a virtual environment including the aforementioned elements and camera composition objective function evaluation module.

As it is possible to see in Fig. 5.4, the sandbox can be controlled by the user via a TCP socket. This communication method has been chosen for two reasons: using socket communication allows the sandbox to run on a machine different from the one of the user and sockets are supported by most of programming languages.

At the time this document is written, the sandbox includes libraries for both Java and C++; these languages have been initially selected as most of the optimisation algorithms in the literature have been implemented in either of the two. To conduct experiments on the sandbox, a system user needs only to include the sandbox library in her project and use the methods to connect, build the camera control problem and evaluate the objective function.

The user can also interact with the sandbox through its user interface; however it is possible only to setup a problem and test a set of standard optimisation functions included in the software.

(a) (b) (c) (d) (e) (f)

Figure 5.5: Main components of the Lerpz Escape game. From left to right: player's avatar (Lerpz), a platform, a collectible item (fuel canister), an enemy (copper), a respawn point and Lerpz's spaceship.

5.2 Lerpz Escape

The second test environment described in this thesis is a three-dimensional platform game designed to evaluate the applicability and the effects of automatic camera control in computer games. Within the context of this thesis, this test environment is used to evaluate both our automatic camera control system and our approach to camera behaviour modelling and automatic camera adaptation against human players.

5.2.1 Game Mechanics

The game is a custom version of Lerpz Escape, a tutorial game by Unity Technologies[2]. It features an alien-like avatar (Lerpz, see Fig. 5.5a) trapped in a futuristic 3D environment made of floating platforms (see Fig. 5.5b). The platforms can be flat, or be composed by multiple smaller platforms with different heights clustered together. Each platform can be connected to another platform through a bridge or be disconnected, in which case the avatar is required to jump to move from one platform to the other.

The game includes three main elements/objects that avatar can interact with: *fuel canisters*, *coppers* and *re-spawn points*. Fuel canisters (see Fig. 5.5c) are floating items that the player needs to collect to complete the game. Coppers (see Fig. 5.5d) are animated robots which chase the avatar and hit it until it falls from a platform. Coppers are normally static and get activated when the avatar enters the platform they are placed on. The player can kill a copper by moving the avatar close to it and hitting it three times; killing allows the player to collect one extra fuel canister which spawns out of the robot's wreckage. Re-spawn points (see Fig. 5.5e) are small glowing stands placed on some platforms. When the avatar touches a re-spawn point this becomes activated; each time the avatar falls from a platform it reappears on the last activated re-spawn point.

[2]http://www.unity3d.com

(a) Tutorial level environment

(b) Main level environment

Figure 5.6: The two virtual environments employed in the user evaluation. The avatar is initially placed at the right side of the map, close to the dome-like building, and has to reach the space-ship at the left side of the map.

The goal of the player is to control Lerpz and make him reach his spaceship (see Fig. 5.5f) at end of the level. However, the spaceship is surrounded by a force field which does not allow any access. To deactivate the field, Lerpz has to collect a number of fuel canisters (2 for the tutorial level and 6 for the main level) across the virtual environment. Moreover, the task has to be performed within a pre defined time window, otherwise the current level will stop and the game will proceed to the next level whether the goal has been achieved or not.

The game designed for this evaluation stands as a prototypical platform game, as it incorporates the typical aspects of the genre. According to the classification described by Wolf (2001) the platform games are defined as *"games in which the primary objective requires movement through a series of levels, by way of running, climbing, jumping, and other means of locomotion"*; moreover, according to Wolf, such games often *"involve the avoidance of dropped or falling objects, conflict with (or navigation around) computer-controlled characters, and often some character, object, or reward at the top of the climb which provides narrative motivation"*.

Therefore, we can reasonably assume that the results of the evaluation performed using

(a) Fight area

(b) Jump area

(c) Collection area

Figure 5.7: The three different area types met in the test-bed game.

the aforementioned game are generalizable to the whole *platform games* genre. Moreover, still according to Wolf's classification, it is possible to hypothesise that such results would, at least, partially, apply also to genres such as Adventure and Obstacle Course which have a very similar interaction scheme.

Moreover, the game has been designed not only to maximise the applicability of the evaluation results, but also to incorporate a highly interactive gameplay that minimises narrative. The reason for this choice is our desire for the evaluation of automatic camera control to focus on on interaction rather than on story telling.

5.2.2 Stages and Areas

The game consists of two stages: a tutorial and the main level. Each level starts with the avatar standing on the initial re-spawn point, as shown in Fig. 5.8a and Fig. 5.8b, and ends when the avatar reaches the spaceship in the last platform. The tutorial level includes a walk-through of the game controls and an explanation of the purpose and the effects of the different objects available. Moreover, during the tutorial the player is presented with all the four main challenges she will face during the main game: jumping from platform to platform, facing and enemy (i.e. a *copper*), collecting items (i.e. *fuel canisters*) and reaching the end of the level (i.e. the *spaceship*).

In the tutorial level the subjects have to face one copper, collect at most five fuel canisters and activate two respawn points. Only one copper is present in the main level, while fuel

(a) Manual camera control game (b) Automatic camera control game

Figure 5.8: Sceenshots from the game used during the evaluation displaying the different controls schemes. The game interface displays the game controls configuration, as well as the current number of collected canisters and the time remaining to complete the level.

canisters and respawn points are 10 and 3, respectively.

The two stages are composed by a series of sub-stages we name *areas* which are classified as *jump*, *fight* or *collection* areas. The areas are categorised according to the game-play experience they offer and the type of challenge they pose to the player. In case an area offers more than one challenge type, the category is defined by the most threatening challenge. The challenges are sorted in decreasing level of threat as follows: fight, jump and collect.

Figure 5.7a shows a fight area where the main threat is given by the opponent copper at the center of the platform. The jump area depicted in Fig. 5.7b is composed of several small floating platforms; the player needs to make the avatar jump across all the platforms to complete the area. Figure 5.7c shows an area where the main task of the player is to collect the fuel cells placed around the platform.

5.2.3 Controls

The game features two possible control schemes along with two possible camera control schemes: manual camera control and automatic camera control. Figure 5.8a shows how the game controls displayed when the camera is controlled manually, while Fig. 5.8b shows how the same controls are presented when the camera is controlled automatically. Instructions on the level objective and what action to perform next are displayed in the lower left corner.

Control of avatar movements is common to both schemes: pressing the up arrow moves the avatar forward (with respect to its local orientation), while pressing the down arrow moves the avatar slowly backward. The left and right arrows correspond respectively to a leftward rotation and a rightward rotation. Pressing a horizontal arrow key at the same time as a vertical arrow key will make the avatar curve towards the selected direction.

On the contrary, the buttons used to activate the punch and jumps actions differ in the two camera control schemes. If the player controls the camera, the left mouse button activates the punch action and the right mouse button activates the jump action. If the camera is controlled by the automatic camera control system, the punch action is activated by pressing the Z key and the jump action is activated by pressing the X key. The difference of the punch and jump controls between the two camera control schemes has been introduced after a pilot evaluation showed that the players felt uncomfortable at controlling the jump and punch actions using the mouse buttons when not controlling the camera with it.

The last difference in the game controls between the two schemes lie in the control of the camera. When the camera is not automatically animated by the automatic camera controller, the player controls the camera position and orientation using the mouse. In such case, the camera automatically follows the avatar at a fixed distance and angle: the player can control the angles by moving the mouse and the distance suing the mouse scroll wheel. Moving the mouse forward increases the vertical angle, while moving it backward decreases it; the same applies for leftward and rightward mouse movement.

The choice of a mouse plus keyboard control scheme for the game, although not ideal for the genre, is motivated by the necessity to make the game playable in a browser. Such a choice allows to conduct experiments with wider audiences, reduced experimentation effort and a more believable playing environment (the participants play in a location of their choice) which leads possibly to a more genuine experience.

5.2.4 Technical Characteristics

The game has been developed using the Unity 3D game engine and built for the web player platform, in order to be played via a web browser. The avatar model is composed of 3334 triangles, platforms are composed on average of 400 triangles each, enemies of 1438 triangles each, fuel canisters of 260 triangles each, respawn points of 192 triangles each and the spaceship of 1548 triangles. Overall, the tutorial level geometry is comprises of approximately 15500 triangles with 5 light sources, while the main level of approximately 20000 triangles with 6 light sources. All objects are skinned with one or more texture and both the stages contain approximately 86.2 Mb of textures.

5.3 Summary

The chapter presented the two 3D applications that will be used in the experiments presented in Chapter 6 , Chapter 7 and Chapter 8. The two environments have been developed to be able to evaluate both the algorithmic performance of camera controller and the effect of automatic camera control and adaptation on human players.

In summary, the first test environment is a development tool (a sandbox) that allows to easily create a wide variety of camera composition problems, to evaluate the resulting objective function and to test different camera control algorithms. The second test environment is 3D platform game integrating all the typical gameplay elements of the game genre. It has been developed to support both automatic and manual camera to assist the design of comparative user survey experiments that aim to evaluate the differences between camera control paradigms.

Chapter 6

Hybrid Genetic Algorithm Evaluation

This chapter[1] intends to evaluate the optimisation algorithm at the core of the automatic camera control architecture described in Chapter 3. For this purpose we propose an analysis of the real-time virtual camera composition problem from a dynamic optimisation perspective and, based on this analysis, we present a comparative study of the proposed solution against a set of state-of-the-art algorithms in automatic camera control. The performance of this hybrid GA is evaluated on its ability to generate well composed shots in real-time in a set of unpredictable dynamic problems. The results obtained are compared against the optimisation algorithms described in Chapter 3: Genetic Algorithm, Differential Evolution, Particle Swarm Optimisation, Sliding Octree and Artificial Potential Field. The comparative study is conducted on a set of scenarios representing the different challenges usually met in virtual camera optimisation. In each scenario, the algorithms are evaluated on their reliability and accuracy in the optimisation of four different problems: optimising the visibility, optimising the projection size, optimising the vantage angle and the combination of these problems. Moreover, each problem in each scenario is tested with one, two or three subjects. To describe these test problems and to better understand the performance of the algorithms, the objective functions connected to each test problem are analysed and a set of complexity measures are suggested. The problems are evaluated on their optimisation complexity both as static and dynamic optimisation problems.

6.1 Test-bed Problems

This section presents the set of 36 dynamic virtual camera problems designed to evaluate the proposed hybrid GA. This study includes 12 dynamic composition problems tested in

[1]The content of this chapter is currently at the second stage of review for the Computer Graphics Forum journal (Burelli and Yannakakis, 2012b)

the 3 virtual environments described in Section 5.1.2. The 12 composition problems include the optimisation of a Vantage Angle constraint, a Projection Size constraint, a Visibility Constraint and the combination of all three constraints. These four conditions are evaluated with one, two and three subjects.. In each test problem, the subjects are positioned and move around the environment in a random fashion. The subjects speed is also randomised and varies from 0 to the average human walking speed (1.38 m/s). The subjects are shaped as stylized human beings, and have human-like proportions and height (approx. 1.8m tall). The test problems include up to three moving subjects, with up to three frame constraints imposed on each subject. We believe that such a range of problems covers a large number of the possible virtual camera composition tasks in terms of optimisation challenges, making them an ideal benchmark to evaluate and compare the performances of the algorithms included in this thesis.

6.2 Complexity

To analyse and compare the performance of the proposed algorithm fairly we need to define common complexity measures (Törn et al., 1999) within common test problems (Floudas and Pardalos, 1990). In turn, the choice of appropriate case studies is a key aspect of any comparative analysis since those affect both the validity and the extensibility of any conclusions derived. In our effort to measure the complexity of each scenario examined we identified four different complexity heuristics: one fitness-based measure and three measures inspired by the complexity analysis of Törn et al. (1999). Moreover, since we aim to analyse a set of dynamic virtual camera composition problems and to evaluate the algorithms on them, we include also a measure of the problem's dynamism, on top of these four measures.

6.2.1 Fitness-based Measures

The average fitness of a population of solutions has been used as a complexity measure for occlusion minimisation test problems (Burelli and Yannakakis, 2010b). This measure, however, did not prove to be fully reliable as the algorithms tested did not perform according to the estimated problem complexity. Moreover the average fitness does not consider any topological aspect of the fitness landscape so it gives limited space for analysis and not considered in this thesis.

Jones and Forrest (1995) proposed the correlation between fitness and distance to a global optimum as a statistical measure of complexity. Fitness Distance Correlation (FDC) is generally considered a valid complexity measure but, as Jones and Forrest point out, it fails to capture the complexity of problems that exhibit scattered fitness landscapes. As seen in Figure 5.3, the objective functions we consider clearly showcase a scattered fitness

landscape, potentially with a large number of sparse local optima. The complexity of such landscapes is only poorly represented by a linear correlation. Even though the above disadvantages limit the use of FDG for the description of virtual camera control complexity, FDG is included in this study as a reference.

6.2.2 Complexity Measures by Törn et al.

Törn et al. (1999) place global optimisation problems into four categories (unimodal, easy, moderate and difficult) by analysing four key aspects of the problem:

- The size of the region of attraction of global optima, p^*. The p^* region is defined as "the largest region containing global optima, x_m, such that when starting an infinitely small step strictly descending local optimisation from any point in p^* then x_m will be found each time."
- The number of affordable objective function evaluations in a unit of time.
- If and how embedded are the global optima. I.e. if there exists local optima near the global optima region of attraction, so that sampling around these leads to the detection of better solution and, eventually, of a global optimum.
- The number of local optima.

The above-mentioned aspects may be transposed to three complexity measure values: P, E and NoL (number of local optima). The P value represents the probability that the region of attraction is missed when randomly sampling the solution space for one frame time (16.6 ms) and is defined as follows:

$$P = (1 - p^*)^{N_f} \tag{6.1}$$

where P^* is the percentage the whole search space being a region of attraction of global optima and N_f denotes the affordable number of objective function evaluations within 16.6 ms.

The E value measures the degree of how embedded are the global optima within local optimal solutions, which further implies that search near those local optima may lead to optimal solutions. The value of E is given by:

$$E = 1 - \frac{1}{nD_{max}} \sum_{i=1}^{n} D_{min}(x_i) \tag{6.2}$$

where $D_{min}(x)$ is the distance to the closest global optima region of attraction from each solution x and D_{max} is the maximum distance between two points in the search space.

In the experiment's search space, this distance equals to the length of the parallelepiped's diagonal. While this formula is not defined mathematically by Törn et al., the proposed formalisation is as close as possible to their description.

The last complexity measure considered, NoL, is the relative size of the areas of the search space containing local optima and it is defined as follows:

$$NoL = \frac{A_{lo}}{A_{go}} \tag{6.3}$$

where A_{lo} and A_{go} are the sizes of the areas of space containing respectively local optima and global optima.

The measures derived by the study of Törn et al. (1999) appear to be appropriate for capturing problem complexity of virtual camera control. The three measures, collectively, draw a broader picture of problem complexity than FDC and, moreover, P considers the evaluation time as a key component of problem complexity. The latter is an important aspect within the analysis of the camera control paradigm since execution time plays a central role for this optimisation problem.

6.2.3 Dynamism

Since the problems presented in this study incorporate a set of dynamic elements the optimisation complexity cannot be only analysed in static terms. We need, therefore, to define a measure of how complex the problems are in terms of dynamism. Such a measure should capture the speed and the magnitude of the changes in the objective function and should give a comprehensive measure for the whole landscape. For this purpose, we define a measure D of the landscape dynamism calculated as follows:

$$D = \frac{1}{K}\frac{1}{T}\sum_{i=0}^{K}\sum_{j=0}^{N-1}|f(x_i, t_{j+1}) - f(x_i, t_j)| \tag{6.4}$$

where K is the number of samples taken from the landscape at each sampling, T is the duration of the animation, N is the number of times the landscape is sampled during the animation, $f(x_i, t_j)$ is the objective function value of the sample x_i at j^{th} sampling time and $f(x_i, t_j)$ is the objective function value of the sample x_i at time $j + 1$.

This measure is a discrete approximation of the average absolute change in the objective function for all the points in the solutions space over unit of time. High D values correspond to highly dynamic problems, while a D value equal to 0 corresponds to a static optimisation problem.

	P	E	NoL	FDC	D
Forest					
Angle 1	0.00	1.00	0.47	-0.87	1.34
Angle 2	0.02	1.00	0.45	-0.68	0.94
Angle 3	0.10	1.00	3.45	-0.73	0.77
Size 1	0.64	0.91	823.63	-0.01	0.14
Size 2	0.96	0.71	8401.20	0.00	0.15
Size 3	0.94	0.73	6382.89	-0.01	0.09
Visib. 1	0.52	1.00	46.10	-0.16	0.04
Visib. 2	0.84	0.59	158.48	-0.20	0.05
Visib. 3	0.97	0.56	17439.26	-0.06	0.03
All 1	0.47	0.85	879.82	-0.03	0.49
All 2	0.53	1.00	1267.34	-0.07	0.20
All 3	0.78	0.95	3343.36	-0.09	0.33
House					
Angle 1	0.00	1.00	0.00	-0.86	1.15
Angle 2	0.12	1.00	3.23	-0.70	0.93
Angle 3	0.14	1.00	5.57	-0.70	0.89
Size 1	0.63	0.93	131.96	-0.03	0.18
Size 2	0.94	0.69	2773.30	-0.02	0.09
Size 3	0.94	0.67	4017.58	-0.02	0.09
Visib. 1	0.63	0.95	95.04	-0.16	0.05
Visib. 2	0.95	0.79	3186.23	-0.07	0.03
Visib. 3	0.97	0.81	14262.47	-0.05	0.04
All 1	0.64	0.99	2912.77	-0.05	0.44
All 2	0.84	0.99	6249.59	-0.09	0.31
All 3	0.89	0.75	8614.36	-0.06	0.33
Square					
Angle 1	0.00	1.00	0.39	-0.92	1.08
Angle 2	0.15	0.88	11.40	-0.63	0.90
Angle 3	0.13	0.91	3.99	-0.74	0.85
Size 1	0.57	0.94	221.38	-0.05	0.17
Size 2	0.93	0.77	696.86	-0.04	0.11
Size 3	0.95	0.55	1902.31	-0.03	0.10
Visib. 1	0.49	0.98	54.32	-0.21	0.06
Visib. 2	0.80	0.92	533.79	-0.24	0.06
Visib. 3	0.90	0.88	1445.53	-0.18	0.06
All 1	0.35	0.94	631.58	-0.04	0.41
All 2	0.69	0.89	1633.66	-0.09	0.34
All 3	0.82	0.95	2497.88	-0.08	0.34

Table 6.1: Complexity measures for each scenario.

6.2.4 Scenario categorisation by problem complexity

Table 6.1 presents the values of the aforementioned complexity measures for each test problem investigated. All the values have been calculated by discretising the solution space with a resolution of 0.5 meters on the X and Z axis, 1 m on the Y axis and by sampling linearly 32 rotations per position. All the problems have been evaluated by sampling the landscape each half second for 10 seconds, this process has been repeated 20 times for each test problem to minimise the effects of the initial random placement of the subjects. The static complexity values contained in Table 6.1 are averaged over the whole number of samples and repetitions, while the dynamism measure is an average of the dynamism calculated in every repetition. The FDC values are also presented in Table 6.1 for comparison purposes. It is clearly shown that FDC follows the trend of the other static complexity measures proposed in most scenarios; however, the measure becomes inaccurate in the most complex cases, in which the correlation value is always close to 0 despite of the complexity differences revealed by the measure defined by Törn et al. This problem is expected as FDC is known to be inaccurate for scattered fitness landscapes and it does not consider the objective function evaluation complexity.

From a static optimisation perspective, the test problems are sorted into 4 categories following the ranges proposed by Törn et al. (1999). According to this categorisation, the vantage angle angle test problems with one subject are categorised as *unimodal* since their P value equals to 0. The other two vantage angle test problems are categorised as *easy*. These scenarios have a smaller global optima attraction area and higher evaluation times resulting in P values greater than 0. All of the aforementioned scenarios are also highly embedded (high E values), meaning that, even if the search algorithm ends in a local optimum, there are high chances that it can still converge to a global optimum if a small random perturbation is applied. The projection size and visibility test problems with one subject also fall in the *easy* category in all test environments; these problems have all an average probability to miss the global optima attraction space around 50-60% and have highly embedded local optima ($E > 0.91$); however, the number of local optima is more than an order of magnitude higher then the previous problems, identifying these problems as slightly more complex then the previous ones. More precisely, according to the categorisation by Törn et al., the first problems fall in the *E1* category while the second ones in the *E2*. The projection size and visibility test problems with more than one subject are categorised as *moderate* in the square virtual environment, since they demonstrate a high P value and embedded local optima, while they are categorised as *difficult* in the other two environments due to the lower E value. Interestingly, the combined problems do not

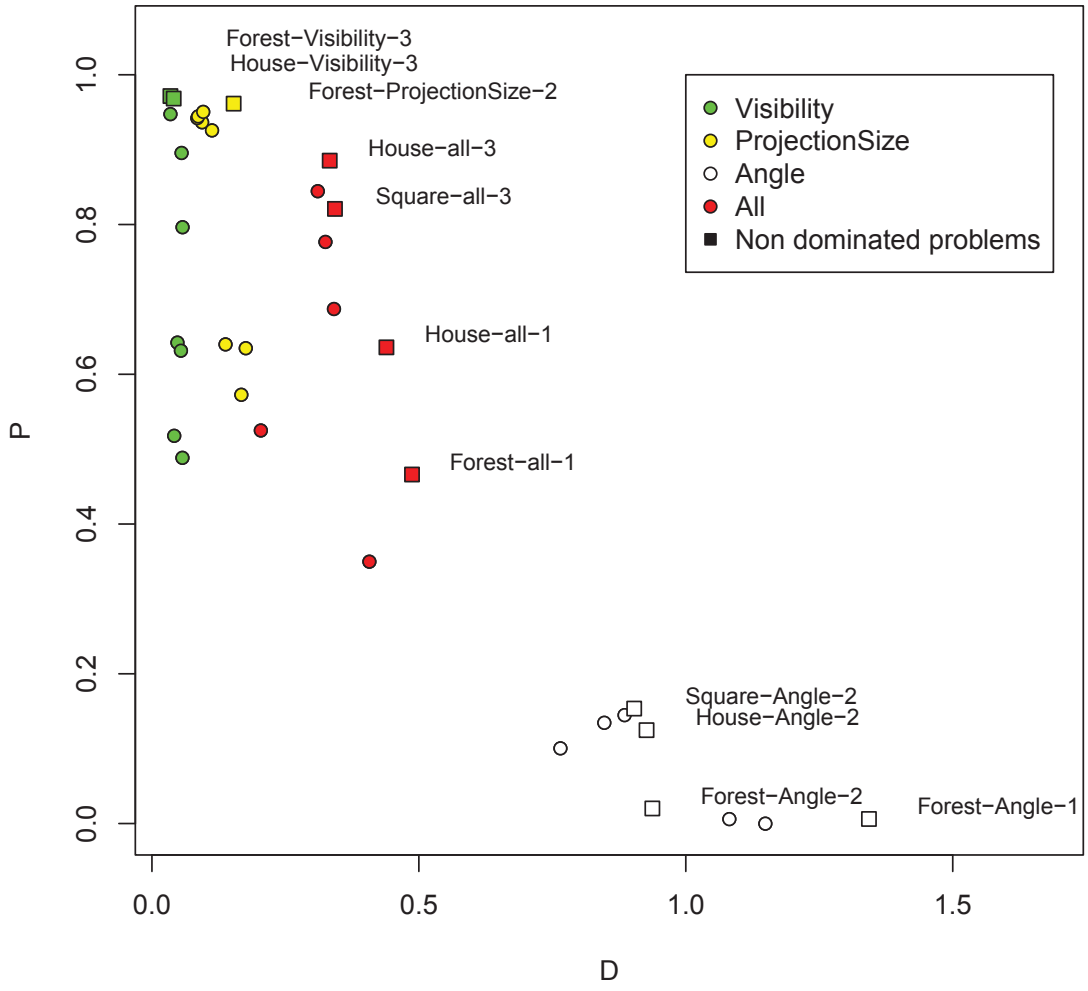


Figure 6.1: Test problems sorted in a scatter plot according to their dynamism factor D and their landscape complexity P. The Pareto front identified by the squares contains all the non dominated problems in terms of complexity and dynamism.

stand as the most complex ones; on the contrary, they exhibit a complexity that varies from *easy* to *moderate*.

When analysing the problems according to their dynamism, a different picture emerges: the visibility and projection size problems have all a significantly lower D value than the angle problems, revealing that the latter are the most complex set of problems from a dynamic optimisation perspective. The reason for such a difference in terms of dynamism is expected, as the angle objective function, depends on the subjects orientation; therefore, even a little orientation change in terms of degrees, has an amplified effect on the objective

function landscape proportional to the distance to the subject.

Due to the contradiction between static and dynamic complexity, we need to sort the problems in a two dimensional space, defined by these two measures. Figure 6.1 shows the problems sorted according to P on the Y axis and D on the X axis; it is apparent how the combined problems — i.e. the ones including all the three frame constraints — stand in the middle of the graph, demonstrating to have an average complexity in terms of both dynamism and static optimisation complexity. The problems identified by a square symbol in Fig. 6.1 belong to the Pareto front; a solution belongs to the Pareto front if it is not dominated by any other solution in the solution space. A Pareto optimal solution cannot be improved with respect to any objective without worsening at least one other objective; thus, this group of problems is the one that can be identified as the most complex both in terms of dynamism and static optimisation complexity. We identify this group as the most representative sub set of problems and, therefore, we concentrate the evaluation of the algorithms on these 11 problems:

1. A visibility constraint on 3 subjects in the forest
2. A visibility constraint on 3 subjects in the house
3. A projection size constraint on 2 subjects in the forest
4. All frame constraints on 3 subjects in the house
5. All frame constraints on 3 subjects in the square
6. All frame constraints on 1 subject in the house
7. All frame constraints on 1 subject in the forest
8. A vantage angle constraint on 2 subjects in the square
9. A vantage angle constraint on 2 subjects in the house
10. A vantage angle constraint on 2 subjects in the forest
11. A vantage angle constraint on 1 subject in the forest

The first three problems (1,2,3) are among the ones categorised as *difficult* in terms of static optimisation complexity, while they have extremely low dynamism. We name these problems as *difficult-slow*. Following the same principles, problems 4,5,6,7 are labelled as *average*, problems 8,9,10 as *easy-fast* and problem 11 as *easy-faster*.

6.3 Algorithms Configurations

The algorithms employed in the comparative study have all been configured based on extensive experimentation and know successful settings.

Differential Evolution is extremely robust to the parameters choice (Storn and Lampinen, 2004; Ali and Törn, 2004); therefore, the algorithm has been configured for this study following the parameter settings suggested by Storn (1996) as the ideal ones for a 6-dimensional optimisation problem: the population consists of one 60 individuals, the crossover probability C is 0.8 and the differential weight W_d is 0.9.

Particle Swarm Optimisation has been employed in camera control for off-line camera optimisation combined with a space pruning technique (Burelli et al., 2008). Based on the results of that study, the algorithm has been implemented using a population of one 64 particles, a cognitive factor of 0.75, a social factor of 0.5 and an inertia linearly decreases from an initial value of 1.2 to 0.2. The cognitive factor has been increased to 0.75 to increase the diversity in the population and improve the algorithm's ability to deal with a dynamic objective function.

The settings of both Genetic Algorithm and hybrid genetic algorithm have been chosen based on the successful parameter settings of our previous study on occlusion minimisation (Burelli and Yannakakis, 2010b): the population consists of 80 individuals, the crossover probability is 100%, the mutation probability is 30% and the selection scheme is ranking with all individuals selected for breeding. The main difference lies in the replacement policy, which is generational as it is better suited for dynamic optimisation.

The configuration of the Artificial Potential Field follows the settings described in the authors' previous work on automatic camera control using Artificial Potential Field (Burelli and Jhala, 2009b). The configuration of the Sliding Octree Algorithm follows the suggestions described in (Bourne et al., 2008); therefore, the scaling factor at each pass is equal to 0.75 and the number of passes is equal to 15 and the maximum movement distance is equal to 1.

6.4 Results

The experiment consists of a comparative analysis of the performance of the six optimisation algorithms. Each algorithm is run to find and track the optimal solution on the 11 dynamic virtual camera composition problems described in Sec. 6.2 All algorithms are evaluated on each problem for 10 seconds, during which, the subjects move in the environment as described in Section 6.2. The best solution found by the algorithms is recorded at each

	Hyb.	GA	DE	PSO	APF	SO
Difficult-Slow						
F. Visib. 3	**0.61**	0.27	0.56	0.43	0.14	0.25
H. Visib. 3	**0.51**	0.23	0.35	0.30	0.07	0.14
F. Size 2	0.10	**0.32**	**0.31**	0.24	0.16	0.22
Average						
H. All 3	**0.38**	0.28	0.35	0.35	0.21	0.30
S. All 3	**0.43**	0.28	0.41	0.39	0.23	0.32
H. All 1	**0.72**	0.34	0.40	0.31	0.28	0.23
F. All 1	**0.89**	0.35	0.38	0.32	0.43	0.25
Easy-Fast						
S. Angle 2	**0.65**	0.46	0.47	0.51	**0.66**	0.46
H. Angle 2	**0.65**	0.53	0.49	0.48	**0.66**	0.49
F. Angle 2	**0.66**	0.51	0.50	0.52	**0.63**	0.48
Easy-Faster						
F. Angle 1	0.49	**0.52**	0.49	0.47	0.49	0.48

Table 6.2: Average best function values (ABFV) reached by each search algorithm. The first column indicates the test environment. The best values are highlighted.

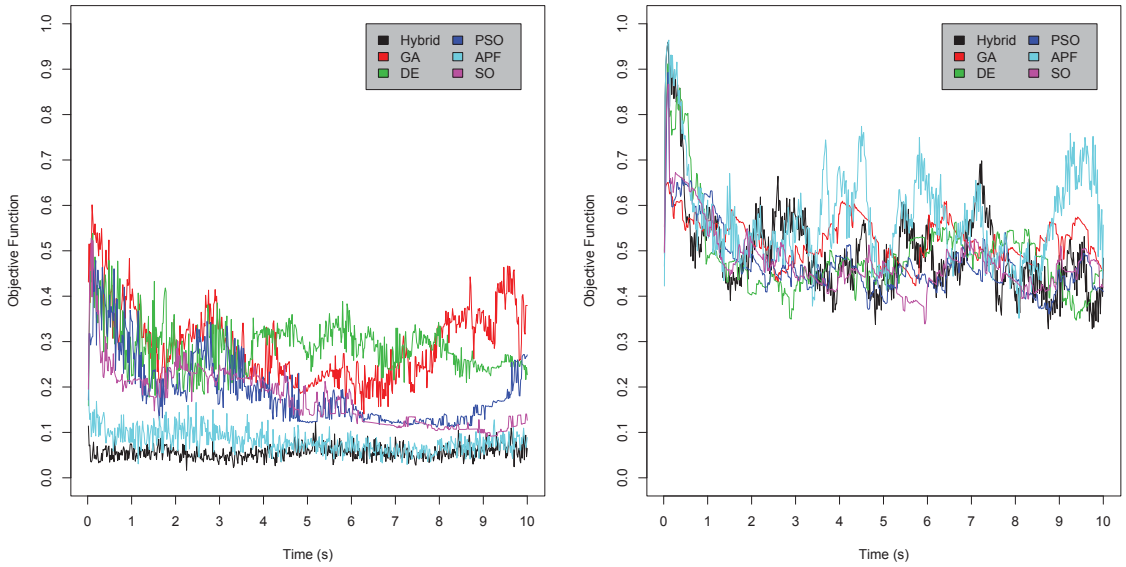
algorithm iteration. All the evaluations are repeated 50 times to to minimise the effects of the initial random placement and the random movements of the subjects.

In this section we showcase the performance of the proposed optimisation method in comparison with the other five search algorithms. The algorithms are compared with respect to the complexity of the task to be solved and the time taken for the algorithm to reach a solution. All experiments are executed on a Intel Core i7 950 3.06 GHz (the implemented algorithms use only one core) with 4 GB of RAM at 1333 MHz. The performance of the algorithms is compared for *accuracy* and *reliability*. Accuracy is measured using the average best function value measure suggested by Schönemann (2007), while reliability is measured as the average amount of time in which the algorithm is unable to find a solution with an objective function value higher than 0.25.

6.4.1 Accuracy

The average best function value (ABFV), obtained by each algorithm across 50 runs on each test problem is used to measure how accurately the algorithm finds and tracks a global optimum. This value is calculated as the mean of the *median run*, which is defined as the list of median values between all the runs at each moment of the computation.

Table 6.2 contains the accuracy values of the hybrid GA opposed to the accuracy values of the other 5 algorithms on all test problems. The values highlighted in bold in each table row have no significant difference in accuracy between them — i.e. the result of the pairwise



(a) Forest - Projection Size - 2

(b) Forest - Angle - 1

Figure 6.2: Median best solution value over time (median run) for each algorithm on the problems in which the proposed hybrid GA fails to achieve the best results.

t-test with the other highlighted algorithms yields a p value greater than 0.01 — but they show a significantly higher accuracy when compared to the non-highlighted values.

From the presented results, it is evident that the hybrid GA is capable of finding and tracking the best solution across the vast majority of the test problems. In particular, the algorithm appears to constantly achieve the best accuracy in the visibility tasks in all virtual environments. Moreover, it also demonstrates the best accuracy in the optimisation of the problems including all the frame constraints, independently of the number of subjects and the type of virtual environment.

If we focus the analysis on a comparison between the hybrid GA and the two algorithms on which this algorithm is based (GA and APF), we can observe two facts: in the *average* and *difficult-slow* problems, the hybrid GA is able to achieve accuracy often more than two times higher than the classical GA and the APF, while in the *easy-fast* and *easy-faster* the hybrid GA and the APF achieve similar results. This behaviour reveals that the scheduler and the early convergence detection method described in Section 3.3 are effectively able to accelerate the convergence of a classical GA while maintaining a sufficient population diversity to adapt to the changes of the objective function. However, in one of the *difficult-slow* optimisation problems and in the *easy-faster* problems, the classical GA achieves higher accuracy. This happens for two distinct reasons: inaccuracy of the approximated projection size derivative and the high dynamism of the problems.

	Hyb.	GA	DE	PSO	APF	SO
Difficult-Slow						
F. Visib. 3	**1.07**	4.48	**1.03**	2.38	7.96	5.02
H. Visib. 3	**0.77**	4.63	3.49	3.62	8.61	6.85
F. Size 2	9.14	**4.43**	**4.33**	6.11	8.15	6.52
Average						
H. All 3	**2.27**	4.66	2.88	2.63	6.95	3.75
S. All 3	**1.85**	4.67	2.01	2.41	6.37	3.89
H. All 1	**1.28**	4.12	3.84	4.99	4.66	6.36
F. All 1	**0.17**	4.38	4.05	5.62	4.05	6.11
Easy-Fast						
S. Angle 2	**0.78**	1.48	2.12	1.52	**0.72**	1.92
H. Angle 2	**0.60**	1.42	1.89	1.73	0.73	1.07
F. Angle 2	**0.78**	1.25	1.88	1.38	0.88	1.82
Easy-Faster						
F. Angle 1	3.66	**2.45**	2.74	2.85	2.66	3.15

Table 6.3: Average time in which the algorithms produces a solution with an objective function value lower than 0.25. The time is expressed in seconds, the duration of each run is 10 seconds. The best values — i.e. the lowest times — are highlighted.

In the projection size optimisation problems, the algorithm fails to accurately optimise the function when more than subject is included in the problem. The approximated derivative becomes inaccurate when dealing with more than one subject, misleading the search algorithms, as it is possible to observe in the median run of both GA and APF depicted in Fig.6.2a. This is even more evident if we consider that the same algorithm with only one subject achieves the best accuracy in all environments (these problems are not part of the Pareto front in Fig. 6.1; therefore, they have not been included in the tables).

In the vantage angle problem with one subject (*easy-faster*), the hybrid algorithm is unable to track optimum due to the high dynamism of the problem and the scheduler is unable to effectively detect the early convergence of the algorithm. This results in the oscillating behaviour of both the hybrid GA and APF visibile in Fig. 6.2b. The latter aspect, in particular, requires future investigation, as the detection of early convergence, in a highly dynamic environment in which it is impossible to rely on value stagnation, is still an open research problem.

6.4.2 Reliability

We measure reliability using the average amount of time in which the algorithms are unable to provide and acceptable solution during each run of the dynamic experiment; the lower this time is the more reliable is the algorithm. A solution is considered acceptable if its

fitness is above a certain fraction of the ideal value of the objective function ($f = 1$); for this experiment, we compare the algorithms' reliability with the following threshold: $F \geq 0.25$. To measure this value, we run each algorithm 50 times, measure the amount of time each algorithm provides a solution with a value lower than 0.25 for each run and we calculate the average of all the runs on each test problem. Table 6.3 shows the values of the reliability measure for each algorithm across the different test problems.

The reliability findings exhibit a pattern very similar to the accuracy results, in which the hybrid GA appears to perform best on all *average* and *easy-fast* problems, with fail times down to 1.7% of the total experiment time. Also similarly to the accuracy results, the performance decrease drastically in two problems: the projection size problem with two subjects and the vantage angle problem with one subject. It is clear that also reliability is affected by the imprecise approximation of the projection size objective function derivative and by the failure of the early convergence detection method.

6.5 Summary

This chapter presented an experimental evaluation of the optimisation algorithm at the core of the automatic camera control architecture described in Chapter 3.

To test this algorithm, we identify the building blocks of the camera control problem and we study the complexity of each block. In particular, we identify three principal objective functions, we relate these to the state-of-the-art definition of frame constraints and we systematically analyse the complexity of each objective function and their combination. As a result of this analysis we identify 11 test problems which represent a wide range of problem complexity values in terms of both static and dynamic optimisation.

These problems compose a benchmark on which we evaluate the performance of the proposed hybrid meta-heuristic algorithm and compare it with other five search algorithms widely used in automatic camera control. The algorithms are compared on two aspects: accuracy and reliability. Accuracy is evaluated using the average best function value (ABFV) measure suggested by Schönemann (2007), while reliability is measured as the average amount of time in which the algorithm is unable to find a solution with an objective function value higher than 0.25.

The proposed search algorithm is able to find and track solutions with the highest accuracy and reliability on most test problems; however the results achieved in two test problems reveal the main limitations of the proposed algorithm. The accuracy and reliability demonstrated by the algorithm in the optimisation of projection size for more than one subject are low as the algorithm is unable to provide an acceptable solution, on average, for 91% of the time. Similar performance is manifested also by the standalone APF algorithm,

suggesting that the approximated projection size objective function derivative used by the APF operator misleads the search process. The hybrid GA appears also unable to accurately and reliably optimise the most dynamic of the problems considered: a vantage angle constraint on one subject in the forest environment. We believe that such a behaviour is the consequence of an imprecise detection of early convergence, which induces the scheduler to constantly switching between the two policies. problem and the scheduler is unable to effectively detect the early convergence of the algorithm, this results in the oscillating behaviour of both the APF and the hybrid GA.

However, the results emerging from the optimisation of the problems including all the frame constraints are very positive: the algorithm shows the best accuracy and reliability independently of the number of subjects and the type of virtual environment. This kind of problem is the most realistic and the most common encountered in interactive virtual environments, where most of the times the camera needs to follow an avatar and a number of other objects.

Chapter 7

Controller Evaluation

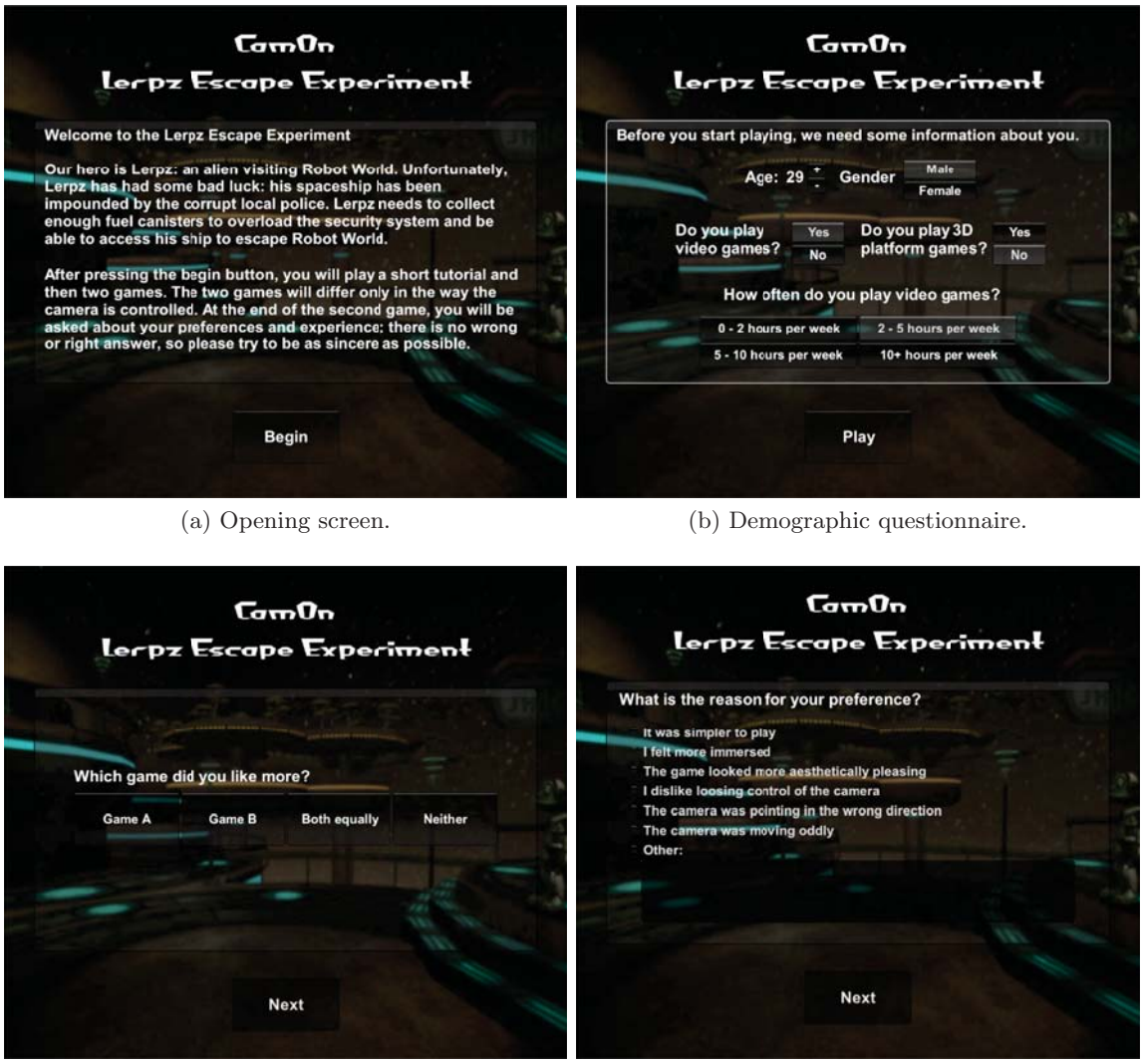
In this chapter[1], we intend to evaluate the automatic camera control architecture described in Chapter 3 and, in general, test the suitability of automatic camera control to three dimensional computer games with respect to its impact on player experience. For this purpose, an implementation of the automatic camera control architecture has been integrated into a custom built three-dimensional platform game including all the stereotypical aspects of the game genre. The game has been published as an on-line experiment on the author's web page[2] attracting 22 subjects as participants of the experiment, among which 16 were males, 13 declared to be regularly playing games and 9 declared to be regularly playing three-dimensional platform games. The age of the subjects is between 26 and 37 years. Each participant was asked to play a custom version of the game presented in Section 5.2. The players initially plays a tutorial level (see Fig. 5.6a) to familiarize with the game, the tutorial level was followed by two instances of the main level (see Fig. 5.6b) alternatively with automatic camera control and manual camera control. At the end of each pair of games we collect information about the player's preferences and their performance in the games. The remainder of the chapter describes the experimental protocol followed (see Section 7.1), the experimental setup (see Section 7.2), the extracted features (see Section 7.3) and the analysis of the data collected (see Section 7.4).

7.1 Experimental Methodology

Our experimental hypotheses are that players prefer to play without controlling the camera and that players who play with an automatically controlled camera will achieve better results in the game (e.g. more collected items, shorter time). To test these hypotheses we have conducted a within subject experimental evaluation, in which each subject plays the

[1] The content of this chapter has been submitted to the ACM Transactions On Graphics journal (Burelli and Yannakakis, 2012a)

[2] http://www.paoloburelli.com/experiments/CamOnLerpzEscapeEvaluation/



(a) Opening screen.

(b) Demographic questionnaire.

(c) 4-AFC questionaire of preference.

(d) Motivations questionaire.

Figure 7.1: Screen-shots of the game menus introducing the experiment and gathering informations about the subject and her experience.

main level with and without automatic camera control and, at the end of the second level played, she or he expresses her or his preference between the two. The order of the levels presented is randomised to eliminate potential order effects.

The participant is initially presented with a short description of the experiment and the game objectives (see Fig. 7.1a). The subject chooses to start the game, by clicking on the start button on the screen. The game then goes in full screen mode and the subject is asked to fill in the following information (see Fig. 7.1b): age, gender, whether she normally plays computer games, whether she normally plays 3D platform games and how often she plays games in a week (e.g. from 0 to 2 hours per week or from 2 to 5 hours per week).

At completion of the questionnaire, the subject starts to play the tutorial level (see Fig. 5.6a). During this phase of the experiment the player plays a short version of the main game level, in which she can familiarise with the controls. To minimise the impact of the learning effect the camera control paradigm is randomised during the tutorial stage. After the tutorial, the participant proceeds to the main part of the experiment, in which she plays two times through the main game level (see Fig. 5.6b) with or without automatic camera control.

At the end of this phase the subject is asked to express a preference between the two main game levels (Game A and Game B) just played (see Fig. 7.1c) and subsequently to provide a motivation for the preference (see Fig. 7.1d). The preference is expressed through 4-alternative forced choice (4-AFC) questionnaire scheme. The preference questionnaire includes four alternative choices: *Game A*, *Game B*, *Neither* or *Both Equally*. This scheme has been chosen over a rating scheme for a number of advantages including the absence of scaling, personality, and cultural biases as well as the lower order and inconsistency effects (Yannakakis and Hallam, 2011). Moreover, a 4-AFC scheme, opposed to a 2-AFC scheme, accounts also for cases of non-preference.

The motivations for preference can be picked out of six predefined options (labels) which are as follows:

- It was simpler to play.
- I felt more immersed.
- I dislike loosing control of the camera.
- The game looked more aesthetically pleasing.
- The camera was pointing in the wrong direction.
- The camera was moving oddly.

The first three motivations have been included to allow us to analyse the magnitude of the impact of the camera control scheme on gameplay. The fourth motivation has been included to gather more information about the role of aesthetics in camera control. The last two motivations have been included to allow the participant to report for any undesired behaviour of the automatic camera controller.

7.2 Experimental Setup

Since subjects participated in the experiment directly on their computers by loading and running the game in a web browser, there is no obvious physical experimental setup. However, we can identify some common conditions among the subjects: all of them played the game through a web browser and the game was played in full screen. All subjects played a custom version of the game presented in Section 5.2: the game features one tutorial level and two main levels. The two main stages differ in terms of the camera control paradigm: in one stage, the player controls the camera manually and, in the other one, the camera is controlled automatically.

In the stage featuring automatic camera control, the controller is instructed to position and animates the camera so to the generated images display the following properties for the avatar:

- The the avatar should be fully visible; to achieve this, the *Object Visibility* property should be equal to 1.0.
- The projection image of the avatar, either horizontally or vertically, should cover approximately one third of the screen length; that means the *Object Projection Size* property should be equal to 0.3.
- The avatar should be shot from above is right shoulder; that means the *Object View Angle* property should be equal to 170 degrees horizontally and 25 degrees vertically.
- The avatar should appear at the crossing point of the lower horizontal line and the left vertical line according to the *rule of the thirds* in photography; that means the *Object Frame Position* property coordinates should be equal to (0.33,0.66).

All together, these properties describe an over-the-shoulder shot, which places the avatar in the lower left quadrant of the screen. This quadrant has been chosen to balance the image, since the controls are depicted on the right side of the screen, and to grant to the player a good view of the horizon. Figure 5.8b displays and example of how the game is framed by the automatic camera controller. The numerical constraint parameters have been selected after a testing phase in which we have searched the values that matched the desired composition principles.

7.3 Collected Features

As already mentioned, the player preference and the motivation for such a preference are asked to the player upon the completion of the two gaming sessions. Given that the two

games differ only in the way the camera is controlled, the self-reported preferences provide a valid indicator of the preferred camera control scheme and a set of arguments for justifying this preference. Moreover, we can evaluate whether the implementation of the camera control architecture presented in this thesis is able to provide a satisfactory camera experience to the player in a real game context. To evaluate this aspect we triangulate the preference information with the average frame-rate and the number of algorithm iterations per second. Statistical features of the in-game performance of the experiment participants are also recorded and include the following:

- Level completion time.
- Number of collected canisters.
- Amount of damage inflicted.
- Amount of damage received.
- Number of falls from the platforms.
- Number of jumps performed.
- Number of respawn points activated.

The performance holds important information about which aspect of the gameplay is affected more by the camera control paradigm and what is the magnitude of such impact.

7.4 Results

As a first step in the analysis of the results we test if the order of play affects the reported preferences and a number of key gameplay features. The section follows with an analysis of the effect of the order of play on the player performance and preferences (see Section 7.4.1) and analysis of the impact of the proposed camera controller on player preferences (see Section 7.4.2) and performance (see Section 7.4.3).

7.4.1 Order Of Play

To check whether the order of playing game levels affects the user's performances, we test the statistical significance of the difference between the features collected in the first game level and in the second game level played. Table 7.1 contains the results of a paired two-sample t-test between each feature collected in the first and in the second game. The statistical test shows that the order of play does not significantly affect any of the features

Feature (F)	First (F_1)	Second (F_2)	p-value
Completion time	187.47	170.81	0.100
Canisters collect.	6.86	7.32	0.094
Damage inflicted	0.54	0.68	0.164
Damage received	0.59	0.86	0.233
Falls	1.22	1.45	0.277
Jumps	57.18	55.54	0.278
Respawns activ.	2.86	2.77	0.164

Table 7.1: Average features of the players with respect to the order of play and test of order of play effect on the features. The p-value is the result of a paired-sample t-test between the values recorded in the first game level and the values recorded in the second game level. Features for which the null hypothesis can be rejected with a 5% threshold (i.e. there is a significant effect of order of play on the feature) are highlighted.

describing the player user's performance as the null hypothesis can be rejected in all cases ($p\text{-}value < 0.05$).

The p-value for the number of collected canisters is very close to the 5% threshold, suggesting the presence of a weak influence of the order of play. Such influence can also be seen in Fig. 7.3b where, in most cases, the fuel canisters difference is negative when the level featuring the automatic camera controller is played as the second game. Such a difference is most likely a learning effect caused by the canister positions which are the same in both games; players appear to be more efficient in collecting them in the second game.

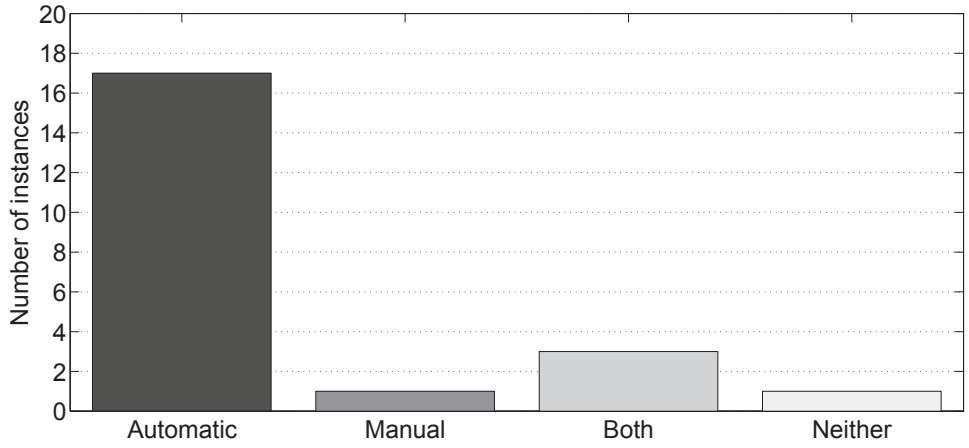
To check the effect of the order of play on the user's preferences, we follow the procedure suggested in (Yannakakis et al., 2008) and we calculate the correlation r_o as follows:

$$r_o = \frac{K - J}{N} \tag{7.1}$$

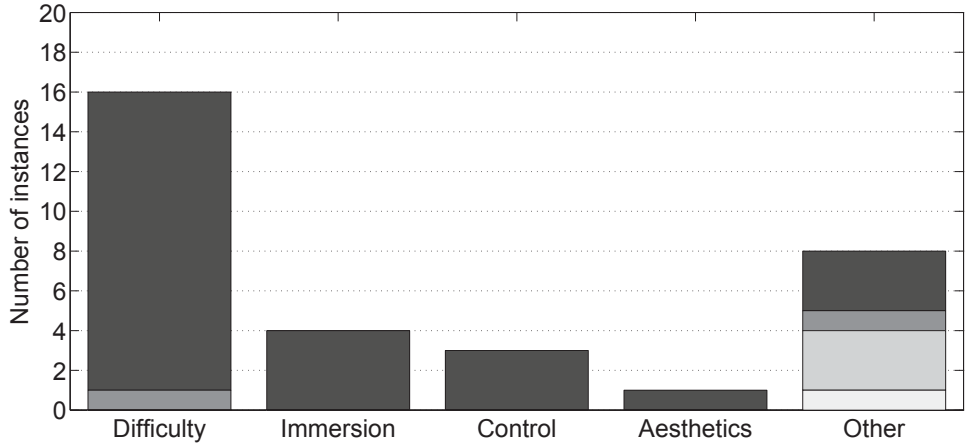
where K is the number of times the users prefer the first game level, J is the number of times the users prefer the second game level and N is the number of games. The statistical test shows that no significant order effect emerges from the reported preferences ($r_o = 0.000$, $p\text{-}value = 0.500$).

7.4.2 Preferences

Figure 7.2a displays the total number of selections for each of the four options of the 4-AFC questionnaire. The first noticeable fact is that 18 subjects (81.8%) have expressed a clear preference between the two camera types. Nearly all of these subjects (17 out of 18) have stated that they prefer playing the level in which the camera was automatically controlled by our system. Therefore, the vast majority of the players prefer the automatic camera scheme over the manual one, confirming our fist initial hypothesis. The significance of this choice is



(a) Preferences



(b) Motivations

Figure 7.2: Expressed preferences (7.2a) and corresponding motivations (7.2b). The bar colours in the motivations chart describe which preference the motivations have been given for.

tested in using the same formula employed to test the order effect with different K, J and N values: K is the number of times the level featuring the automatic camera controller was picked, whereas J is the number of times the level with manual camera control was picked; N is the number of of games where subjects expressed a clear preference (either $A < B$ or $A > B$; 2-AFC). The test reveals a strong positive correlation ($r_o = 0.889$, $p\text{-}value = 0.001$) confirming the presence of a significant influence of the camera control scheme on the users' preference.

The most common motivation, given by 15 subjects out of 17, is that the level with automatic camera control was "simpler to play", two of these subjects have explicitly stated that

Feature (F)	Automatic (F_a)	Manual (F_m)	p-value
Completion time	167.70	191.05	**0.032**
Canisters collect.	7.09	7.09	0.500
Damage inflicted	0.55	0.68	0.164
Damage received	0.82	0.64	0.314
Falls	1.59	1.09	0.093
Jumps	58.27	54.45	0.081
Respawns activ.	2.82	2.82	0.500

Table 7.2: Average features of the players with respect to camera control type. The third column contains the p-values of a paired two sample t-test between the two measurements across camera control type. The features for which the null hypothesis can be rejected with a 5% of significance (i.e. there is a significant effect of the camera control scheme on the feature) appear in bold.

the game was made simpler by the fact that the camera was controlled automatically. Such results validate our initial hypothesis that players would prefer an automatically controlled camera to a manually controlled camera in a virtual environment such as a 3D platform game.

A very interesting fact that emerges from the subjects' motivations (see Fig. 7.2b) is that, although the automatic camera controller provided a well composed shot according to the *rule of the thirds*, only one subject stated that she perceived the level with automatic camera as "more aesthetically pleasing" and only four "felt more immersed". Considering that the automatic camera controller was explicitly instructed to provide properly composed shots, such results might question the applicability of classical photography and cinematography rules to computer games.

The four subjects (18.2%) that have expressed no preference between the two games motivated the fact by stating that they did not like the control in general and found both games frustrating. It is worth noticing that one of these four subjects, despite having expressed no preference between the two levels, she stated in the motivations that she preferred the camera to be automatically controlled

The subject which stated that prefers the level featuring manual camera control, motivated her choice by saying that she personally "prefers a mouse+keyboard control for PC games". This answer, along with the fact that no subject reported any kind of unexpected or annoying behaviour of the camera, allows us to state that the camera control system is able to successfully position and animate the camera in a real game environment with dynamic geometry and interactive events.

Feature (F)	$c(\vec{z})$	p-value
Completion time	-0.389	**0.037**
Canisters collect.	-0.111	0.311
Damage inflicted	0.056	0.403
Damage received	0.111	0.311
Falls	0.167	0.229
Jumps	0.222	0.160
Respawns activ.	0.000	0.500

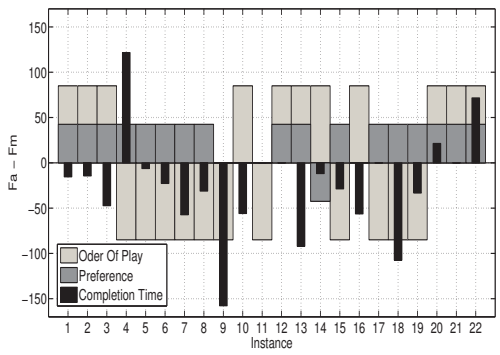
Table 7.3: Correlation between collected features and reported preferences. Features for which the null hypothesis can be rejected with a 5% threshold (i.e. there is a significant correlation between preference and the feature) are highlighted.

7.4.3 Gameplay

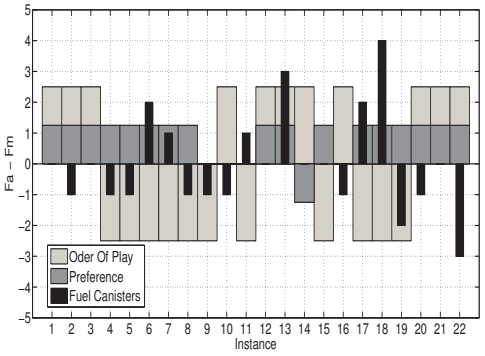
Our second hypothesis is that, when players play with the automatic camera controller their performance in the game will increase. As described earlier in Section 7.3, we have extracted and collected seven features to describe the players' performance during the game. The collected features have been sorted according to the camera control scheme (see Table 7.2) and the two groups of measurements have been tested to assess whether their means differ significantly. The first two columns of Table 7.2 contain the mean values of each feature in the levels featuring the automatic camera controller and the ones in which the player is requested to manually control the camera. The last column displays the p-values of a two-sample paired t-test between the two groups of features.

These statistics reveal that only completion time is affected by the camera control scheme as it is the only feature for which it is possible to reject the null hypothesis. In other words, between the games played with the automatic camera controller and without it, only one feature significantly differs (with a 5% threshold), while no significant difference emerges for the other features; this feature is completion time. Figure 7.3a shows how the difference of completion time between the automatically controlled and the manually controlled stages is constantly equal or below 0, except from 3 instances, further confirming that the automatic camera controller helps reducing the time required to complete the game.

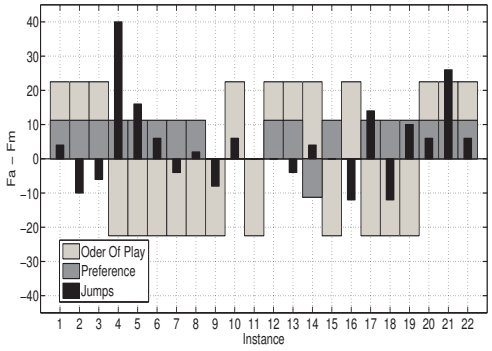
Other two features worth observing are the number of falls and the number of jumps. In both tests (see Tables 7.1 and 7.2) the p-values are slightly above the 5% significance level, meaning that these features are probably also mildly affected by the camera control scheme. The average number of jumps is about 9% lower in the games played with automatic camera control and the number of falls is about 32% lower. Moreover, the trend depicted in Figure 7.3 shows that, for both features, the difference between the level with the automatic camera controller and the one with a manually controlled camera is positive more than half



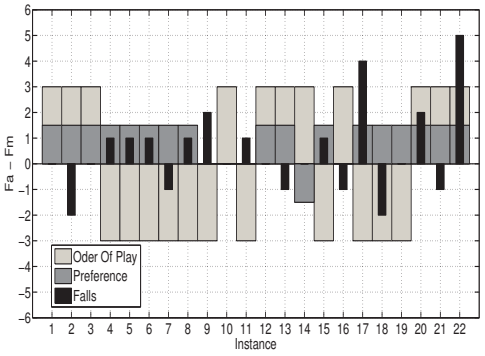
(a) Completion time



(b) Fuel canisters



(c) Jumps



(d) Falls

Figure 7.3: Differences $\Delta F = F_a - F_m$ of completion time (7.3a), number of canisters collected (7.3b), number of jumps (7.3c) and number of falls (7.3d) between the games played with the automatic camera controller and the ones with manual camera control. The background color pattern shows which level was preferred and which level was played first for each game played. If the dark grey bar is in the upper half of the plot the level featuring the automatic camera controller was preferred and vice versa. If the light grey bar is in the upper half of the plot the level featuring the automatic camera controller was played first and vice versa. The four features displayed have a significant or close-to-significant (i.e. *p-value* < 0.10) correlation with either the order of play or the camera control paradigm.

of the times. An explanation for these findings could be that the automatic camera controller did not support adequately the jump action and, therefore, it did not provide the optimal view point for the player to jump between platforms, resulting in more attempts and more falls.

Finally, an analysis of statistically significant correlations between subject expressed preferences and collected gameplay features has been carried out. Correlation coefficients $c(\vec{z})$ are calculated according to the following equation:

$$c(\vec{z}) = \sum_{i=1}^{N_p} (z_i / N_p) \tag{7.2}$$

where N_p is the total number of games where subjects expressed a clear preference and $z_i = 1$, if the subject preferred the level with the larger value of the examined feature and $z_i = -1$, if the subject chose the other level. A significant negative correlation is observed between average completion time and reported preference (see Table 7.3) further validating the results of the survey; since shorter completion times can be seen as signs of lower gameplay difficulty; decreased difficulty was widely reported as a motivation for preferring the automatic camera over the manual one.

7.5 Summary

The chapter presented a user evaluation of the automatic camera control architecture described in Chapter 3 in a commercial-standard three-dimensional computer game. Through this experiment, the performance of the camera control system and the impact of automatic camera control in third-person computer games are evaluated.

The pairwise preferences reported by the participants indicated that automatic camera control is perceived as an enhancement to the game by simplifying the interaction. Furthermore, the subjects reported no complaints on the camera behaviour, which, along with the dominating preference for automatic camera control, suggests that the camera control system was able to elicit a satisfactory experience without noticeable issues on the quality of the generated images and camera motion. The analysis of the features describing player performance in the game revealed that when players did not control the camera, they completed the level in less time, confirming our hypothesis that the automatic camera controller would positively influence their performance.

However, when playing the level with the automatic camera controller, a large amount of players had more difficulty to perform jumps. The reason for such a negative impact is most likely due to the camera profile chosen which did not support all the types of actions the player had to perform.

Chapter 8

Evaluation of Adaptive Camera

In this chapter[1] we investigate player preferences concerning virtual camera placement and animation, by applying the methodology described in Chapter 4 to a 3D platform game. We build a model of the relationship between camera behaviour, player behaviour and gameplay and we evaluate the performance of this model. For this purpose, data from player gaze and the virtual camera motion is collected in a data-collection experiment and used to identify and describe the players' camera behaviours.

The obtained model is then employed to adjust in real-time the behaviour of an automatic camera controller at the same 3D platform game. This camera adaptation process is tested through a user evaluation experiment in which a set of participants are asked to express their preference between a level of the game played featuring automatic camera control with a static camera profile (identical to the controller employed in the evaluation study of Chapter 7) and a level that features the adaptive camera profile approach.

The reminder of this chapter presents the models obtained from the first experiment, discusses their accuracy and finally presents the results of the user evaluation of the adaptation mechanism.

8.1 Camera Behaviour Models

Our hypothesis is that camera behaviour depends on player's actions and the game context and that it is possible to predict the current camera behaviour given a description of the current game situation and a track of the previous player actions. Therefore, we expect that different players will focus on different object types in the same game situation and that the same player will observe different types of object in different game situations. Moreover, we expect to find a function that relates these differences to the game situation and the previous actions of the players..

[1] Parts of this chapter have been published in Burelli and Yannakakis (2011) and Picardi et al. (2011)

To evaluate this hypothesis, we conducted an experiment in which we collected eye gaze, camera and game-play data from subjects playing a 3D platform game. In this experiment the participants play a three-dimensional platform game featuring all the stereotypical aspects of the genre's mechanics.

The player controls, using keyboard and mouse, a humanoid avatar and the camera through the game while jumping on platforms, fighting with enemies and collecting objects (see Section 5.2). Camera behaviour is modelled using the combination of gaze and camera position at each frame. Combining gaze data with camera data allows a finer analysis of the player's visual behaviour permitting, not only to understand what objects are visualised by the player, but also which ones are actually observed. This information permits to filter exactly which object is relevant for the player among the ones visualised by the player through her control of the virtual camera. A cluster analysis of the gaze data collected is run to investigate the existence of different virtual camera motion patterns among different players and different areas of the game. Moreover, the relationship between game-play and camera behaviour is investigated through a linear analysis but also modelled using artificial neural networks. Results obtained support the initial hypothesis: relevant differences in the camera behaviour were found among groups of players and among game situations, and the models built are accurate predictors of the camera behaviour.

8.1.1 Experiment

A custom version of the three-dimensional platform game presented in Section 5.2 has been developed as a testbed for this study. Similarly to the original game it features an alien-like avatar in a futuristic environment floating in the open space. The player controls the avatar and the camera using keyboard and mouse. Avatar movements, defined in camera-relative space, are controlled using the arrow keys, and jump and hit actions are activated by pressing the left and right mouse buttons, respectively. The camera orbits around the avatar at a fixed distance; the player can change the distance using the mouse wheel and can rotate the camera around the avatar by moving the mouse. The only difference between the game used in this experiment and the original test-bed environment lies in the structure of the main game level. For the purpose of this study, the main game comprises 34 areas containing 14 *collection* areas, 11 *fight* areas and 9 *jump* areas. The length of the main level is longer than the original game to maximise the amount of data collected from each participant. This is possible for two reasons: first, in this experiment the participants are expected to play only one instance of the main level, so the total duration of the experiment is shorter; and, second, the participants play the game in a controlled environment, so it is much less likely for them to drop the experiment before its conclusion.

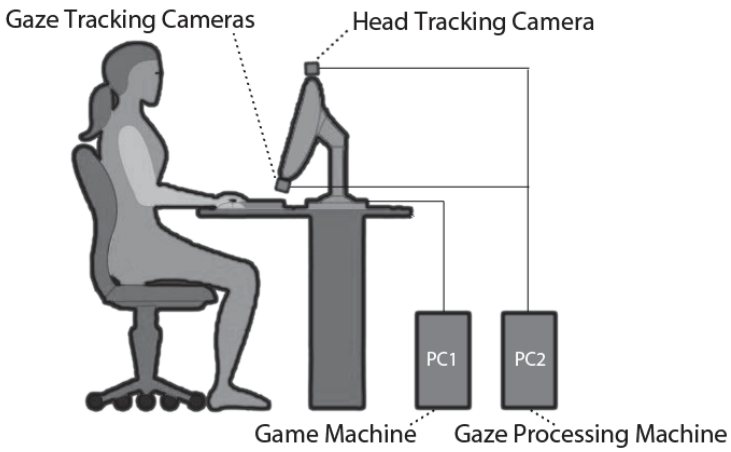


Figure 8.1: Experimental setup used for the data collection experiment.

We represent the virtual camera behaviour as the amount of time the player spends framing and observing different objects in the game environment while playing the game. This representation of behaviour is chosen over a direct model of the camera position and motion as it describes the behaviour in terms of the content visualised by the camera and, therefore, it is independent of the absolute position of the avatar, the camera and other objects. To get information about the objects observed by the player during the experiment, we used the *Eyefollower*[2] gaze tracker The Eyefollower is a device able to locate the player's gaze position on a computer screen through a combination of three cameras. The first camera, placed on top of the screen, tracks the head position while two other motorised infra-red cameras, placed at the bottom of the screen, follow the player's eyes and track the point of regard (Fig. 8.1). which samples the player's gaze at a 120 Hz frequency (60 Hz per eye).

The Eyefollower necessitates an environment with fluorescent lighting and no daylight or incandescent lights as ambient infra-red light or incandescent lights would interfere with the infra-red light emitted by the gaze tracker. In order to provide the best possible operational environment for the gaze tracker, the experiment is set in a room without windows.

The room contains a desk, a chair and two computers (see Fig. 8.1): the first computer runs the computer game while the second one collects the data from the game and the player's gaze and synchronises the two streams into one log. The first computer is connected to a 17" screen, a keyboard and a mouse; the second computer is only connected to the gaze tracker and, through a network cable to the first computer. Running the gaze tracking process and the game on different machines guarantees the necessary resources for both and minimizes the risk of data loss due to heavy computation.

[2]developed by LC Technologies, Inc. - www.eyegaze.com

Twenty-nine subjects participated in the experiment. Twenty-four were male and the age of the participants ranged between 23 and 34 years (mean=26.7, standard deviation=2.41). Statistical data about game-play behaviour, virtual camera movement and gaze position is collected for each participant. The collection of the data is synchronised to the Eyefollower sampling rate, therefore, both data from the game and from the gaze tracker are sampled 120 times per second. Each data sample contains: information about the game-play, position and orientation of the camera, coordinates of the gaze position on the screen and the number and the type of objects around the avatar. The objects are classified into two categories: close and far. All the objects reachable by the avatar within the next action are labelled as close, otherwise as far.

The data is logged only during the time the participants play the game; this phase is preceded by the calibration of the gaze tracking system, a tutorial level and a demographics questionnaire. However, this is only a step of the whole experimental procedure, which is structured as follows:

1. The participant enters the experiment room and is asked to sit at the desk in front of the screen.
2. The screen and the cameras are placed at 60 cm away from the participants' head and just below her line of sight.
3. The experimenter guides the participant through the gaze tracker calibration process.
4. The experimenter leaves the room and the participant starts the experiment by clicking on the start button on the screen.
5. The participant is presented on-screen instructions about the game controls and proceeds to the tutorial level of the game.
6. After the tutorial, the participant is asked to answer a few demographic questions.
7. The participant plays through the game during which data is logged.
8. Once the game is over, the player exits the experiment room.

The total amount of time needed to complete the experiment is, on average, 13 minutes. A time limitation is intentionally not considered in order to give the players the possibility to familiarise well with the game and choose the game style they prefer.

8.1.2 Collected Data Types and Feature Extraction

The data collected for each game is sorted into three datasets according to the three area types described in Section 5.2. Each time a player completes an area two types of statistical features are extracted from that area: game-play and camera related features. The features of the first type are the experiment's independent variables and encapsulate elements of the player's behaviour in the area. The features of the second type describe the camera behaviour for each platform, therefore, they define the experiment's dependent variables.

The player's behaviour is defined by the actions the player performs in each area or, more precisely, by the consequences of these actions. Hence, the features extracted describe the interaction between the player and the game though the avatar's actions, rather than the sequences of pressed keys. For each area the following features have been extracted: the numbers of fuel cells collected, damage given, damage received, enemies killed, re-spawn points visited and jumps. The features are normalised to a range between 0 and 1 using a standard min-max normalisation.

To model the camera behaviour, we analyse the content visualised by the camera instead of the camera absolute position and rotation. The presence of a certain object on the screen, however, does not necessarily imply an intentionality of the player; e.g. the object might be on the screen only because it is close to an object the player is interested to. The gaze data available permits to overcome this limitation since, using the gaze position, it is possible to understand which object is currently observed among the ones framed by the player. Therefore, we combine camera movements and gaze coordinates to identify the objects observed by the player at each frame and we extract the following statistical features: the relative camera speed as well as the time spent observing the avatar, the fuel cells close to the avatar, the enemies close to the avatar, the re-spawn points close to the avatar, the jump pads close to the avatar, the platforms close to the avatar and the far objects.

The seven features related to the time spent observing objects are calculated as the sum of the durations of the smooth pursuit and fixation movements of the eyes (Yarbus, 1967) during which the gaze position falls within an object's projected image. These values are normalised to $[0, 1]$ via the completion time of each area. The first feature is the average speed S of the camera relative to the avatar and it is defined as $S = (D_c - D_a)/T$, where D_c is the distance covered by the camera during the completion of an area, D_a is the distance covered by the avatar and T is the time spent to complete the area.

Each time the avatar leaves an area, the aforementioned features are logged for that area. The data is then, cleaned from experimental noise by removing all the records with a duration inferior to 3 seconds and the ones with no platforms or no enemies and fuel cells left. The remaining records are sorted into three separate groups according to the area type

	Area Type		
Index	Collection	Fight	Jump
Davies-Bouldin	3	**3**	8
Krzanowski-Lai	**2**	**3**	**3**
Calinski-Harabasz	**2**	**3**	**3**
Dunn	**2**	2	2
Hubert-Levin	3	**3**	**3**

Table 8.1: Optimal K-values for the five validity indexes used to chose the number of clusters. The K-value is selected according to a majority voting scheme; the selected value is highlighted in the table for each area type.

and stored into three datasets, containing 239 records for the collection areas, 378 records for the fight areas and 142 records for the jump areas.

8.1.3 Behaviour Models

The number of distinct camera behaviours as well as their internal characteristics can only be based, in part, on domain knowledge. One can infer camera behaviour profiles inspired by a theoretical framework of virtual cinematography (Jhala and Young, 2005) or alternatively follow an empirical approach — as the one suggested here — to derive camera behaviour profiles directly from data. The few existing frameworks focus primarily on story-driven experiences with little or no interaction, thus are not applicable in our context. Therefore, we adopt a data-driven approach and we employ the k-means clustering algorithm on the gaze-based extracted features for the purpose of retrieving the number and type of different camera behaviours.

Unsupervised learning allows us to isolate the most significant groups of samples from each dataset. However, k-means requires the number of clusters k existent in the data to minimise the intra-cluster variance. To overcome this limitation, the algorithm runs with a progressively higher k value — from 2 to 10 — and the clusters generated at each run are evaluated using a set of five cluster validity indexes. The algorithm runs 50 times for each k and the run with the smallest within cluster sum of squared errors is picked. Each selected partition is evaluated against 5 validity indices: Davies-Bouldin (1979), Krzanowski-Lai (1988), Calinski-Harabasz (1974), Dunn (1973) and Hubert-Levin (1976). As displayed in Table 8.1, the smallest k value that optimises at least 3 validity measures out of five is used for the clustering; the chosen k value is 2 for the collection type areas and 3 for the fight and jump type areas.

Collection Areas ($k = 2$)						
Records	Avatar	Fuel Cells	Jump Pads	Re-spawn Points	Far Objects	Speed
150	0.595	0.108	0.034	0.113	0.021	3.338
89	0.361	0.125	0.056	0.072	0.012	8.852

Fight Areas ($k = 3$)							
Records	Avatar	Fuel Cells	Coppers	Jump Pads	Re-spawn Points	Far Objects	Speed
137	0.674	0.042	0.095	0.049	0.034	0.036	3.283
99	0.676	0.032	0.478	0.044	0.056	0.025	5.293
142	0.250	0.029	0.069	0.030	0.052	0.013	5.927

Jump Areas ($k = 3$)					
Records	Avatar	Fuel Cells	Platforms	Far Objects	Speed
33	0.759	0.464	0.795	0.202	2.1293
80	0.736	0.166	0.658	0.059	2.7593
29	0.450	0.113	0.559	0.012	5.5854

Table 8.2: Camera behaviour features of the centroid of each cluster and the number of records in each cluster. Speed indicates the average camera speed with respect to the avatar. The remaining features express the time spent observing each object of a type in an area divided by the time spent completing the area. Highlighted in dark grey colour is the feature related to the main task of the area type. The features related to the other objects close to the avatar are highlighted in light grey colour.

Clusters Characteristics

As seen in Table 8.2, the camera behaviour is described with a different feature set for each area type. The features are selected to match the visual stimuli offered by each area, thus only the features describing observation of objects which are present in the area type are included in the set. Moreover, for each area type the features are sorted into 5 categories: camera speed and time spent observing the avatar, the primary task objects, other close objects and far objects. The primary task objects highlighted in dark grey colour in Table 8.2, represent the time spent observing objects relative to the main task of each area type; all the other objects are categorised as secondary. The time spent observing the avatar, the far objects and the camera speed are sorted separately since they represent type-independent features. According to this feature categorisation it is possible to observe four main behaviour types: *task focused* including the clusters spending more time observing the task related objects, *avatar focused* including the clusters mostly observing the avatar, *overview* which includes the clusters evenly observing all the objects in each area and *look ahead* which includes the clusters that reveal a higher time spent observing the far objects. Each of these categories is divided according to the camera speed as *slow* of *fast* according to whether the speed is higher or lower than the overall average speed, 4.667.

The two clusters emerging from the gaze behaviour data of the collection areas show two very distinct behaviours in terms of camera speed and time spent observing the avatar and

the fuel cells. The first cluster of players spend, on average, almost twice as much time as the players of the second cluster observing the avatar. Moreover, the players belonging to the first cluster spend on average 20% less time observing fuel cells and move the camera at less then half of the speed, compared to the players belonging to the second cluster. These values suggest that the camera for the first cluster should move slowly and focus primarily on the avatar, while the behaviour exhibited by the second cluster would be better modelled with a fast camera with an equal focus on the avatar and the other relevant items in the area. We label these behaviour patterns, respectively, as *slow avatar focused* and *fast task focused*.

The first and the second cluster of behaviours in the fight areas exhibit a pattern matching, respectively, the *slow avatar focused* and the *fast task focused* types. The third cluster of behaviours, on the other hand, exhibits a very short time for all the objects, suggesting a far camera moving at high speed. We label this behaviour as *fast overview*.

In the jump areas, as expected, all the players spend a large amount of time observing the platforms; however, their camera behaviour differs on all the other parameters. The centroid of the first cluster evenly spends a large amount of time focusing on all the objects in the area and moves the camera at very low speed (2.1293). Moreover, the time spent looking at far objects is more than four times larger than all the other clusters. Such values suggest that the camera for this cluster should stay at a longer distance in order to frame all the objects of each platform and should move at a low speed. We label this behaviour as *slow look ahead*. Both the second and the third clusters of players focus primarily on the avatar and the platforms, but differ on the camera speed. Since the platforms are the primary task objects in these areas, we label the two clusters as *slow task focused* and *fast task focused*, respectively.

Behaviour Prediction

Once the relevant camera behaviour types are identified, we proceed by modeling the relationship between game-play and camera behaviour types. More precisely, since the model is intended to select the most appropriate camera behaviour that fits the player's preferences in the game, we attempt to approximate the function that maps the game-play behaviour of the player to the camera behaviour. For this purpose, we use Artificial Neural Networks (ANNs) which are chosen as a function known for its universal approximation capacity. In particular, we employ a different ANN for each area type, instead of one for the whole dataset, to be able to base each model on the best features necessary to describe the game-play in that area.

The three fully connected feed-forward ANNs are trained using Resilient Backpropagation (Riedmiller and Braun, 1993) on the game-play data (ANN input) and the camera behavior clusters (ANN output) using early stopping for over-fitting avoidance. The networks employ the logistic (sigmoid) function at all their neurons. The performance of the ANNs is obtained as the best classification accuracy in 100 independent runs using 3-fold cross-validation. While the inputs of the ANN are selected algorithmically from the set of game-play features the ANN outputs are a set of binary values corresponding to each cluster of the dataset.

The exact ANN input features, the number of hidden neurons and the number of previous areas considered in the ANN input are found empirically through automatic feature selection and exhaustive experimentation. Sequential Forward Selection (SFS) (Kittler, 1978) is employed to find the feature subset that yields the most accurate ANN model. SFS is a local-search procedure in which a feature is added at a time to the current feature (ANN input) set until the accuracy of the prediction increases. The feature to be added is selected from the subset of the remaining features such that the new feature set generates the maximum value of the performance function over all candidate features for addition. The performance of a feature set is calculated as the best classification accuracy of the model in 100 independent runs using 3-fold cross-validation. The network used for the feature selection process has one hidden layer containing a number of neurons equal to two times the number of inputs. Once the best feature set is selected, the best ANN topology is calculated through an exhaustive search of all possible combinations of neurons in two hidden layers with a maximum of 30 neurons per layer.

The combination of automatic feature and topology selection is tested on three different feature sets representing different time horizons in the past: input (game-play features) from one (*one step*) previous area visited in the past, input from the previous two areas visited (*two step*) and the combination of one previous area in the past with the average features of the rest of the previous areas visited (*one step + average*). Each area is represented by the 6 features describing player behaviour introduced in section 8.1.3 and other 3 values describing the number of fuel cells, coppers and re-spawn points present in the area.

8.2 Models Evaluation

In this section we present and discuss the results in terms of prediction accuracy of the camera behaviour models created. First, a statistical analysis of the data is performed to check the existence of a relationship between camera behaviours and game-play features and to identify the significant ones. Then, the prediction accuracy of the models is evaluated

Area Type	Fuel Cells	Damage Given	Damage Received	Enemies Killed	Re-spawn Points	Jumps
Collection	**5.02**	-	-	-	1.23	0.76
Fight	1.63	**12.42**	**10.89**	**24.03**	0.49	**9.64**
Jump	**11.98**	-	-	-	-	0.53

Table 8.3: F-distribution values of the inter-cluster ANOVA test on the game-play features. The threshold for a 5% significance level is $F > 3.85$ for the collection areas and $F > 2.99$ for the jump areas. Values above the threshold appear in bold.

with respect to the length of the time-series expressing the past which is considered in the ANN input, the selected feature set and the network topology.

To isolate the significant features among the ones logged, we perform an inter-cluster one-way ANOVA (Snedecor and Cochran, 1989) for each game-play feature to identify for which features we can reject the null hypothesis (no statistical difference exists).

As it is possible to see in Table 8.3, for each area type, at least one feature demonstrates a significant difference revealing the existence of significant linear relationships. In the fight areas dataset there is a significant difference in terms of damage (both given and taken), number of killed enemies and number of jumps. In the other two area datasets the clusters differ significantly in the number of fuel cells collected.

The features highlighted by the ANOVA test reveal the existence of a linear relationship between the current camera behaviour and those features. However variable relationships, in general, are most likely more complex given that linearly separable problems are extremely rare. Thus, the aim of the analysis presented below is the construction of non-linear computational models of camera behaviour via the use of ANNs described in Section 8.1.3. Figure 8.2 depicts the best performance (3-fold cross validation) of 100 runs for each feature set on the three representations of the past events described in Section 8.1.3. The best value over 100 runs is picked to minimise the impact of the initial randomisation of the network weights on the backpropagation performance.

Each value shown in Figure 8.2 corresponds to the best topology found. It is noteworthy that all the selected topologies have at least one hidden layer, confirming the non linear nature of the relationship. This aspect is also highlighted, in Figure 8.2, by the difference in prediction accuracy between the ANNs that use the subset of significant features identified through the ANOVA test and the ANNs using the subset identified through SFS. The latter type of ANNs, yield a better accuracy regardless of the past representation and game areas. The best 3-fold cross-validation performances achieved for the fight, the jump, and the collection areas are, respectively, 70.04%, 76.43% and 82.29%. It is worth noting that in the collection areas, while the first type of ANNs, built solely on the features that are

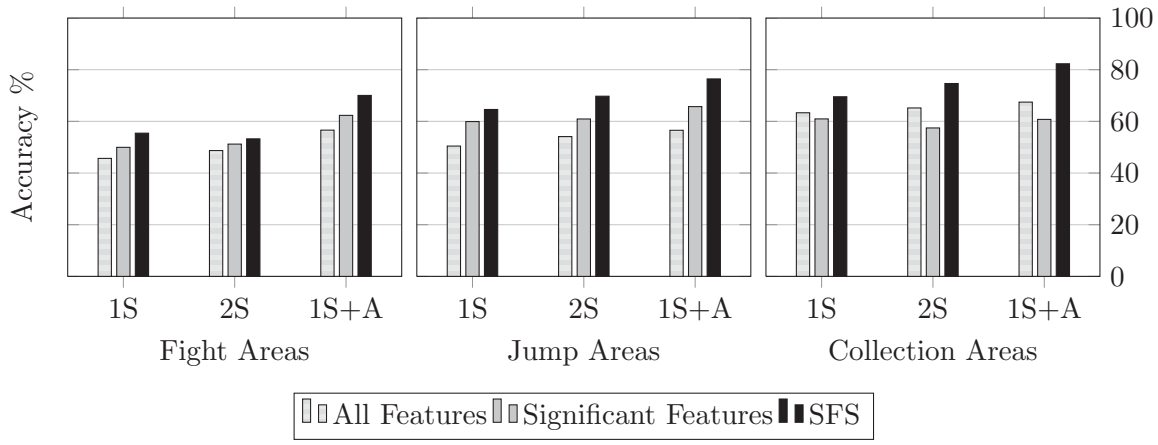


Figure 8.2: Best 3-fold cross-validation performance obtain by the three ANNs across different input feature sets and past representations. The bars labelled 1S refer to the one step representation of the past trace, the ones labelled 2S refer to the two step representation and 1S+A to the representation combining one previous step and the average of the whole past trace.

found significant by ANOVA, perform even worse than the ones using the full feature set, indicating that the linear analysis does not capture the relationship between game-play and camera behaviour accurately.

While, in the collection and jump areas, the ANOVA test indicates the number of fuel cells collected as the only significant feature, SFS selects the number of jumps and the number of re-spawn points activated as additional features for the ANN input. On the other hand, in the collection areas, SFS does not only select features not indicated by ANOVA (the number of re-spawn points activated), but it also discards the number of jumps performed.

The results shown in Figure 8.2 confirm also our supposition that a more extensive representation of the past events would lead to a better accuracy. In fact, the best accuracies are achieved when the ANNs use the most extensive information about the past game-play events. All ANN models predicting camera behaviour in the collection type areas have two hidden layers suggesting a more complex function between game-play and camera behaviour in the collection areas. The non-linearity of the models emerges also from the difference, in terms of performance, between the models based on significant features and the ones based on the features selected with SFS. The first models exhibit always an accuracy lower than the corresponding models assisted by SFS and, for the collection type areas, they even perform worse than the models based on the full feature set. This confirms further our conjecture on the complexity of the function between game-play and camera behaviour in the collection type areas, as a selection based on linear relationships discards very important

information.

8.3 User Evaluation

The second step towards adaptive camera control requires the adaptation of the camera behaviour based on the models just evaluated. Following the methodology described in Chapter 3, we need to define a camera profile corresponding to each behaviour identified and to employ the predictive models to choose the right behaviour at the right moment in the game. In this section we present an experiment in which we apply such a methodology to the 3D platform game described in Chapter 5 and we test the effects of the adaptive camera control on human players. Through this experiment we evaluate the effectiveness of the proposed methodology beyond the numerical accuracy of the models and we analyse the impact of adaptivity on player preferences and in-game performance.

The remainder of the section describes the experiment, the mechanism used to select and activate the behaviours, the profiles associated to each camera behaviour identified in Section 8.1.3 and it concludes with a discussion on the evaluation results obtained.

8.3.1 Experiment

Our experimental hypothesis is that the camera behaviour models built on the combination of gameplay, gaze and camera information can be successfully used to adapt the camera behaviour to the player's preferences. To test this hypothesis we have conducted a within-subject evaluation in which each subject plays the same level with two camera control schemes (conditions): (a) the camera is controlled automatically with a static profile or (b) the camera is controlled by the player profile that is influenced by the player behaviour. At the end of the second experiment each subject expresses her or his preferences between the two camera control schemes.

This experiment has been conducted in parallel with the evaluation presented in Chapter 7 and it shares the same protocol structure and the same game mechanics; therefore, the rest of the section will only highlight the main differences with that experiment. For details about the game and the experimental setup please refer to Chapter 7.

More precisely, the two experiments share the following aspects:

- The experiments have been conducted on-line; therefore the players have participated to the experiment on a web browser though the Unity 3D web player.
- The game used in the evaluation is the one described in Section 5.2.

- In both experiments the players are asked to answer to a demographic questionnaire, play a tutorial level followed by two main levels with different experimental conditions and they finally are asked for their preference between the two last games and the motivations for their choice.
- The order of the two camera control conditions is randomised to minimise learning effect.
- The preference is expressed, in both experiments, using a 4-AFC scheme. The motivations can be chosen from a list of suggestions or expressed freely using a text box.
- The data collected includes both the player preferences and the following seven features from the gameplay: *level completion time, number of collected canisters, amount of damage inflicted, amount of damage received, number of falls from the platforms, number of jumps performed and number of respawn points activated.*

The primary difference between the two experiments lies in the experimental conditions as this experiment aims to evaluate the improvements given to automatic camera control by adapting the camera profiles to the player behaviour. Therefore, in both conditions the player controls only the avatar movements while the automatic camera controller animates the camera according to the camera profile: in one condition the camera profile is static, while in the other condition the camera profile is selected from a list of behaviours according to the way the player plays. Furthermore, as no experimental condition features direct manual camera control, during the tutorial, the player controls manually the camera to eliminate any possible learning effect on the camera control scheme.

Adaptation

In the level featuring adaptive camera control the camera profile depends on the way the player performed in the game and on the type of area in which the player is currently playing. At the instant in which the player enters an area, the game takes the gameplay data collected up until that moment, selects the most appropriate prediction model and activates it using the collected data as input. The output of the model indicates which camera behaviour should be selected from that moment until the player enters another area. Figure 8.3 displays an example of a camera profile transition that happens between a *collection* area and a *fight* area. The new profile is activated two seconds after the player enters the new area, to avoid multiple repetitive changes if the player moves in and out of the area.

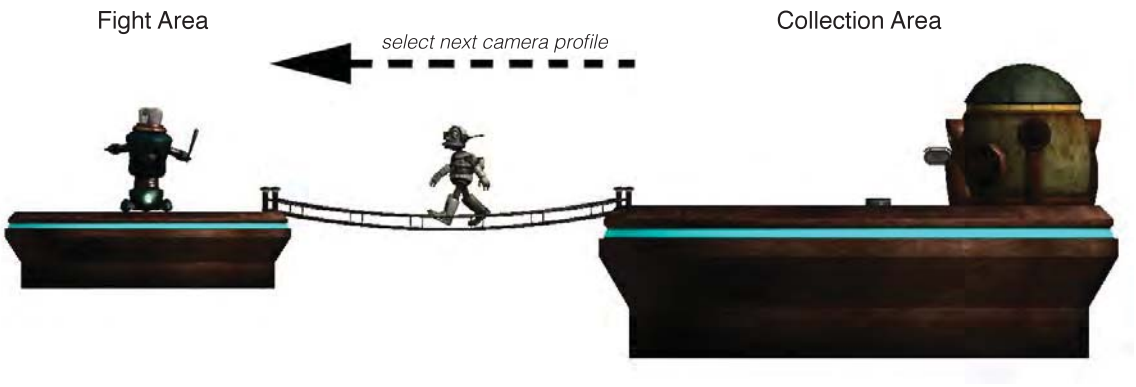


Figure 8.3: Example of a transition from one camera profile to another. As soon as the avatar enters the bridge, the neural network relative to the fight area is activated using the player's gameplay features, recorded in all the previously visited fight areas, as its input. The camera profile associated to the behaviour cluster selected by the network is activated until the avatar moves to a new area.

A new camera profile can be activated also in the case the current area changes category, and this could happen in two cases: the current area is a *collection* area and the last collectible item is taken or the area is a *fight* area and the last enemy present is killed. This event is considered as an area transition by the game, therefore the area is considered completed and the player enters a new area. The new area type depends on the type of elements present in the area and it is categorised according to the most threatening challenge. The challenges are sorted in decreasing level of threat as follows: fight, jump and collect. If the area offers none of these challenges, it will have no category; therefore no prediction model is activated to identify the behaviour and a neutral camera profile is selected.

The models of camera behaviour presented in Section 8.1.3 are able to predict the camera behaviour cluster each player belongs to in each area based on two inputs: the area type and a set of features describing the player behaviour in the previous platform of the corresponding type. Therefore, before the right camera profile can be selected for the player it is necessary that she plays at least one area for each area type. Moreover, as it emerges from the results presented in Section 8.2, the more areas are used by the model to predict the behaviour, the more accurate the model will be.

For this reason, in this experiment, the data about the player in-game behaviour is collected starting from the tutorial game. The tutorial includes at least one area of each type (2 *collection*, 1 *jump*, 1 *fight*), allowing the model to predict the camera profile starting from the first area of the level played with adaptive camera control The in-game player behaviour is recorded throughout the game so that, each time the model is requested to predict which camera behaviour cluster to select, the inputs are updated to the way the

player plays up until that moment. This means that a player playing twice the same area might experience two different camera behaviours, either because its behaviour changed during the game or because a more accurate model (with more inputs) can be used.

After the tutorial game, the player has crossed at least two collection areas, one fight area and one jump area. The next time the player enters an area, the game selects the most appropriate prediction model for the according to area type and the number of areas previously explored. For instance, when the player enters a jump area for the first time after the tutorial, the game will activate the behaviour prediction model that uses only one previous area in its inputs. As displayed in Figure 8.2, the behaviour will be predicted with approximately 62% accuracy. The following time a jump area will be visited, two previous areas will be used for the prediction and the accuracy will increase up to the maximum prediction accuracy of 76.43%.

Camera Profiles

In the level featuring no adaptivity, the controller is instructed to position and animate the camera so that the generated images display the following properties for the avatar:

- The *Object Visibility* property should be equal to 1.0, meaning that the avatar should be fully visible.
- The *Object Projection Size* property should be equal to 0.3, meaning that the projection image of the avatar, either horizontally or vertically, should cover approximately one third of the screen along.
- The *Vantage Angle* property should be equal to 170 degrees horizontally and 25 degrees vertically.
- The *Object Frame Position* property coordinates should be equal to (0.33,0.66), meaning that the avatar should appear at the crossing point of the lower horizontal line and the left vertical line according to the *rule of the thirds* in photography.

All together, these properties describe an over the shoulder shot, which places the avatar in the lower left quadrant of the screen. This quadrant has been chosen to balance the image, since the controls are depicted on the right side of the screen, and to grant to the player a good view of the horizon. Figure 5.8b displays and example how the game is framed by the automatic camera controller. The numerical constraint parameters have been selected after a testing phase in which we have searched the values that matched the desired composition principles.

The profiles used in the level featuring adaptive camera control are built on top of this profile. Each profile differs on three aspects: the weight given to the frame constraints on the avatar, the desired angle and projection size of the avatar and the other objects on which a frame constraint is imposed. The basic profile is also used in the level featuring adaptive camera control, when the current area cannot be placed in any category.

A different camera profile corresponds to each behaviour identified in Section 8.1.3. Each camera profile includes the frame constraints of the basic profile plus one *Object Visibility* constraint for each object present in the platform that belongs to a category observed for more than 5% of the time (e.g. the profile corresponding to the first behaviour cluster of the collection areas will contain an *Object Visibility* frame constraint for each re-spawn point and one for each fuel cell). The 5% threshold has been introduced to deal with possible noise in the collected data; the objects observed for a time windows shorter than the threshold are considered not important.

The time spent observing the far objects is considered separately and it affects the vertical angle value of the *Object View Angle* constraint imposed on the avatar. The vertical angle β will be equal to 45° if the time spent far objects is equal to 0 and it will decrease down to 0 as the time increases according to the following formula:

$$\beta = 45 * \frac{(t_{max} - t_{fo})}{t_{max}} \tag{8.1}$$

where t_{max} is the maximum fraction of time spent observing far objects in all behaviour clusters and equals 0.202 and t_{fo} is the fraction of time spent observing the far objects in the current cluster.

The weight of the frame constraints imposed on the avatar is equal to the fraction of time the avatar is observed, while the weight of the *Object Visibility* constraints for the other objects is equal to the fraction of time their type of object is observed in the cluster divided by the number of objects present in the area (e.g. if there are 3 fuel cells in the area and the current active camera profile corresponds to the second collection behaviour, each *Object Visibility* constraint will have a weight equal to 0.125/3). The weight of each object is assigned according to this scheme because the observation time of each cluster is independent of the number of objects present in each area.

At the time the player enters a new area, the game queries the most appropriate behaviour prediction model (given the number of previous areas played and the type of area) and builds an initial camera profile; when the player collects an item or kills an enemy, the profile is updated. If such an action triggers an area type change, a complete new profile is selected, otherwise the current profile is updated according to the remaining number of

- *Object Visibility* on Avatar: visibility=1, weight=0.36
- *Object Projection Size* on Avatar: size=0.3, weight=0.36
- *Vantage Angle* on Avatar: $\alpha = 170$, $\beta = 42$, weight=0.36
- *Object Frame Position* on Avatar: $x = 0.33$, $y = 0.66$, weight=0.36
- *Object Visibility* on Avatar: visibility=1, weight=0.36
- *Object Visibility* on Fuel Canister: visibility=1, weight=0.06
- *Object Visibility* on Fuel Canister: visibility=1, weight=0.06
- *Object Visibility* on Re-spawn: visibility=1, weight=0.07

a: Profile for two fuel canisters and one active re-spawn point.

- *Object Visibility* on Avatar: visibility=1, weight=0.36
- *Object Projection Size* on Avatar: size=0.3, weight=0.36
- *Vantage Angle* on Avatar: $\alpha = 170$, $\beta = 42$, weight=0.36
- *Object Frame Position* on Avatar: $x = 0.33$, $y = 0.66$, weight=0.36
- *Object Visibility* on Avatar: visibility=1, weight=0.36
- *Object Visibility* on Fuel Canister: visibility=1, weight=0.12

b: Profile for one fuel canister.

Figure 8.4: Changes in a camera profile before and after the collection of one fuel canister and the activation of one re-spawn point. The screen-shots above each profile depict the camera configuration produced by the two profiles for the same avatar position.

objects in the area. Figure 8.4 shows an example of the update of a camera profile in a *collection* area. The camera behaviour remains unaltered as the type of area does not change, but two frame constraints are dropped from the camera profile and the weight of the *Object Visibility* constraint on the remaining fuel canister is doubled.

8.3.2 Results

Twenty eight subjects participated to the experiment, among which 21 were males, 20 declared to be regularly playing games and 14 declared to be regularly playing three-dimensional platform games. The age of the participants ranged between 18 and 40 years (mean=27.04, standard deviation=4.63). Before the analysis, the data has been filtered to remove the logs of the experimental sessions in which the average frame-rate was lower than 30 frames per second. The data about these subjects have been removed to guarantee that all the participants included in the study have been exposed to the same experience. After this selection, the number of subjects considered is 23 in this experiment.

Due to the similarities in the structure between the experiment presented in this section and the experiment protocol presented in Chapter 7, the analysis of the results follows the same steps. We first control the effect of the order of play on the participants performance and on their preferences. The section ends with an analysis of the impact of the adaptive camera behaviours on the participants' preferences and performance.

Order Of Play

To check whether the order of playing the levels affects the user's performances, we test for the statistical significance of the difference between the features collected in the first level and in the second level played. Table 8.4 contains the results of a paired two-sample t-test between each feature collected in the first and in the second game.

The statistical test shows that the order of play affects one of the features describing the players' performance: completion time. Moreover, similarly to what happened in the evaluation of the automatic camera controller, the p-value for the number of collected canisters is very close to the 5% significance level, suggesting the presence of a weak influence of the order of play on this feature. Both influences can be observed also in Figure 8.7a and in Figure 8.7b. In the first case, the completion time difference between the level featuring an adaptive camera behaviour is almost always positive when the adaptive level is played first. The opposite behaviour can be seen for the fuel canisters where, in several cases, the difference is positive when the level featuring the adaptivity is played as the second game. Similarly to what emerged in the automatic camera controller evaluation experiment, such

Feature (F)	First (F_1)	Second (F_2)	p-value
Completion time	172.00	148.24	**0.01**
Canisters collected	7.09	7.78	0.07
Damage inflicted	0.57	0.57	0.50
Damage received	0.30	0.35	0.61
Falls	2.83	2.09	0.20
Jumps	58.87	53.04	0.15
Respawns activated	2.65	2.74	0.29

Table 8.4: Average performance values of the players with respect to the order of play and test of order of play effect on the features. The p-value is the result of a paired-sample t-test between the values recorded in the first level and the values recorded in the second game. Features for which the null hypothesis can be rejected with a 5% threshold (i.e. there is a significant effect of order of play on the feature) are highlighted.

an influence most likely occurs since players can remember the positions of the objects and the directions to reach the end of the level.

To check the effect of the order of play on the user's preferences, we follow the procedure suggested by Yannakakis et al. (2008) and we calculate the correlation r_o as follows:

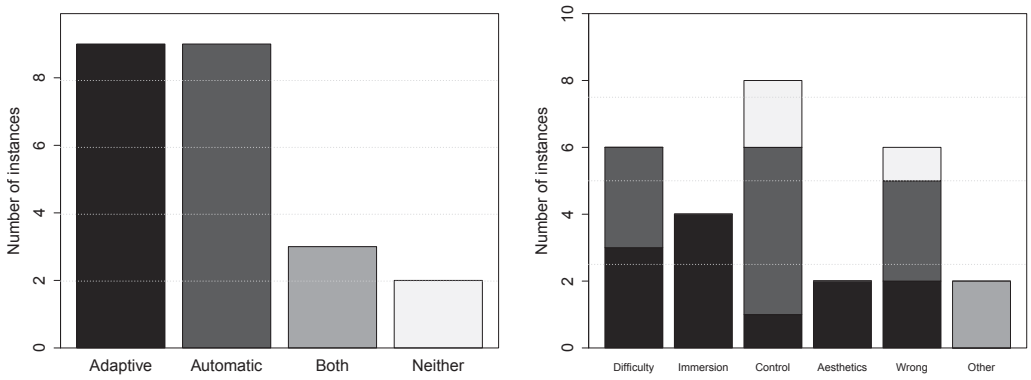
$$r_o = \frac{K - J}{N} \tag{8.2}$$

where K is the number of times the users prefer the first level, J is the number of times the users prefer the second level and N is the number of games. The greater the absolute value of r_o the more the order of play tends to affect the subjects' preference; r_o is trinomially-distributed under the null hypothesis. The statistical test shows that no significant order effect emerges from the reported preferences ($r_o = -0.22$, $p\text{-}value = 0.46$).

Preferences

Figure 8.5a displays the total number of selections for each of the four options of the 4-AFC questionnaire. The first noticeable fact is that 18 subjects (78.3%) have expressed a clear preference between the two camera types. Among these, half of the subjects (9 out of 18) have stated that they prefer playing the level in which the camera profile was adapted and the same amount preferred the level without adaptation.

The significance and the magnitude of the correlation between preference and camera control paradigm can be tested using the same formula employed to test the order effect with different K, J and N values: K is the number of times the level featuring the adaptive camera controller is preferred plus the number of times both level have been preferred, J is the number of times the level without adaptivity is preferred and N is the number of games. A positive correlation value expresses a stronger preference for the levels featuring



(a) Preferences

(b) Motivations

Figure 8.5: Expressed preferences (a) and corresponding motivations (b). The bar colours in the motivations chart describe which preference the motivations have been given for.

the adaptive camera behaviour, while a negative correlation expresses a stronger preference for the levels without adaptation.

Even by considering the *Both equally* preferences as positive (assuming that the adaptation process is successful also when not noticeable), the test reveals only a very mild correlation ($r_o = 0.14$) with no statistical significance ($p\text{-}value = 0.51$). These results indicate that the adaptivity has not been clearly preferred by all the subjects; however, if we sort the subjects according to their performances, it appears that there are two distinct groups of players among the participants with different preferences about the camera control paradigm.

To identify such groups we clustered the subjects, according to the completion time and the number of collected canisters, into three groups using k-means: the *expert players*, the *intermediate players* and the *novice players* (see Figure 8.6). The first group contains the 9 players which, on average, have completed each level in approximately 106 seconds and have collected approximately 8.1 canisters. The second group contains the 9 players which, on average, have completed each level in approximately 154 seconds and have collected approximately 8.3 canisters, while the last group contains 5 subjects which completed the level in approximately 276 seconds and have collected approximately 4.6 canisters on average. The main difference between the expert players and the intermediate player groups is the completion time. In addition, the novices group differs from the other two in both features considered (completion time and canisters collected). The novice players were often unable to complete the level and, when they did complete it, it took them significantly more time than the other players.

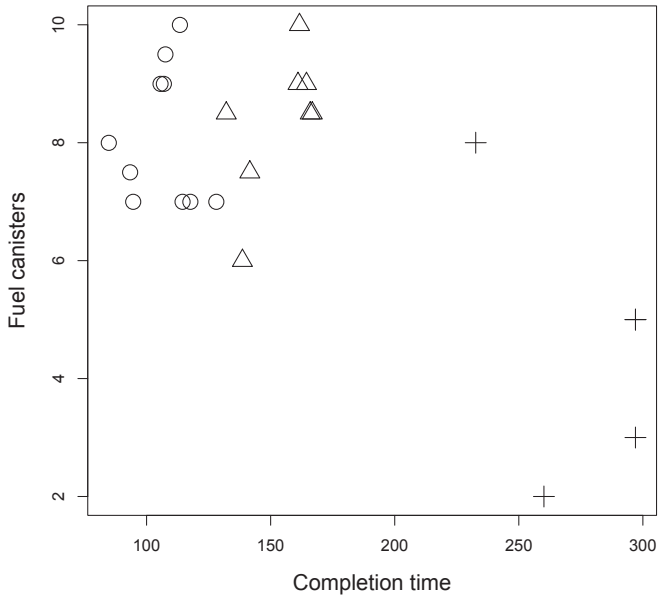


Figure 8.6: Subjects sorted according to their average completion time and average number of collected canisters. The subjects indicated with a *cross* symbol belong to the expert players cluster, the ones indicated with a *triangle* symbol belong to the average players cluster, while the subjects indicated with a *circle* symbol belong to the novices cluster.

As displayed in Table 8.5, the correlation coefficient r_o for the first group of subjects equals -0.55 and it clearly indicates that the expert players preferred the level without adaptation. The same analysis carried out on the two non-expert groups reveals, instead, a significant positive correlation between preference and adaptivity ($r_o = 0.71$ for the intermediate players and $r_o = 0.6$ for the novice players), indicating that the non expert-players preferred the level played with adaptation.

The motivations given by the players for their preferences (see Figure 8.5b) can partially motivate these results; in fact, the main motivation given when the level without adaptation is preferred is the lack of sense of control. The answer *"I dislike loosing control of the*

Player group	r_o	*p-value*
All players	0.14	0.51
Experts	-0.55	0.04
Intermediate	0.71	0.01
Novices	0.6	0.05

Table 8.5: Correlation between expressed preference and camera type for all players and for the players sorted according to their skill. A positive correlation expresses a positive preference towards the adaptive camera control experience.

Feature (F)	Adap. (F_{ad})	Auto. (F_{au})	p
Completion time	156.54	163.70	0.25
Canisters collected	7.96	6.91	**0.01**
Damage inflicted	0.61	0.52	0.08
Damage received	0.35	0.30	0.61
Falls	1.83	3.09	**0.05**
Jumps	54.70	57.22	0.54
Respawns activated	2.78	2.61	0.13

a: All players

Feature (F)	Adap. (F_{ad})	Auto. (F_{au})	p
Completion time	115.07	98.28	0.90
Canisters collected	8.20	8.00	0.31
Damage inflicted	0.70	0.60	0.17
Damage received	0.40	0.10	0.96
Falls	0.80	0.30	0.84
Jumps	53.20	46.80	0.11
Respawns activated	2.80	2.90	0.70

b: Expert players

Feature (F)	Adap. (F_{ad})	Auto. (F_{au})	p
Completion time	146.12	161.92	0.19
Canisters collected	8.50	8.25	0.26
Damage inflicted	0.75	0.75	1.00
Damage received	0.25	0.75	**0.05**
Falls	1.50	1.88	0.29
Jumps	57.50	62.25	0.45
Respawns activated	3.00	3.00	1.00

c: Intermediate players

Feature (F)	Adap. (F_{ad})	Auto. (F_{au})	p
Completion time	256.15	297.38	0.10
Canisters collected	6.60	2.60	**0.01**
Damage inflicted	0.20	0.00	0.19
Damage received	0.40	0.00	0.91
Falls	4.40	10.60	**0.04**
Jumps	53.20	70.00	0.27
Respawns activated	2.40	1.40	**0.04**

d: Novice players

Table 8.6: Average performance values of the players with respect to camera control type. The third column contains the p-values of a paired two sample t-test between the two measurements across camera control type. The features for which the null hypothesis can be rejected with a 5% of significance (i.e. there is a significant effect of the camera control scheme on the feature) appear in bold.

camera" has been given 5 times in total (4 times by the expert players). Moreover 3 subjects reported that the camera pointed in the wrong direction while playing the level with adaptive camera control, suggesting that the camera behaviour models are either not accurate enough or not general enough. The two subjects that have expressed a dislike for both levels motivated the fact by stating that they did not like automatic camera control in general and found both controls frustrating.

Gameplay

Our second hypothesis is that adaptive camera control will positively affect the player. As described earlier, we have extracted and collected seven features to describe the players' performance during the game. The collected features have been sorted according to the camera control scheme (see Table 8.6) and the two groups of measurements have been tested to assess whether their means differ significantly. Moreover, the same analysis has been separately applied to the three groups of subjects introduced in the previous section.

Feature (F)	Pref.	Other	p
Completion time	149.57	170.66	**0.02**
Canisters collected	7.78	7.09	0.07
Damage inflicted	0.57	0.57	0.50
Damage received	0.39	0.26	0.81
Falls	2.04	2.87	0.17
Jumps	53.13	58.78	0.16
Respawns activated	2.83	2.57	0.04

a: All players

Feature (F)	Pref.	Other	p
Completion time	93.18	120.17	**0.01**
Canisters collected	8.10	8.10	0.50
Damage inflicted	0.60	0.70	0.83
Damage received	0.20	0.30	0.30
Falls	0.20	0.90	0.08
Jumps	45.20	54.80	0.01
Respawns activated	2.90	2.80	0.30

b: Expert players

Feature (F)	Pref.	Other	p
Completion time	137.43	170.61	**0.02**
Canisters collected	8.38	8.38	0.50
Damage inflicted	0.75	0.75	1.00
Damage received	0.62	0.38	0.77
Falls	1.38	2.00	0.18
Jumps	55.50	64.25	0.14
Respawns activated	3.00	3.00	1.00

c: Intermediate players

Feature (F)	Pref.	Other	p
Completion time	281.79	271.74	0.61
Canisters collected	6.20	3.00	**0.05**
Damage inflicted	0.20	0.00	0.19
Damage received	0.40	0.00	0.91
Falls	6.80	8.20	0.37
Jumps	65.20	58.00	0.66
Respawns activated	2.40	1.40	**0.04**

d: Novice players

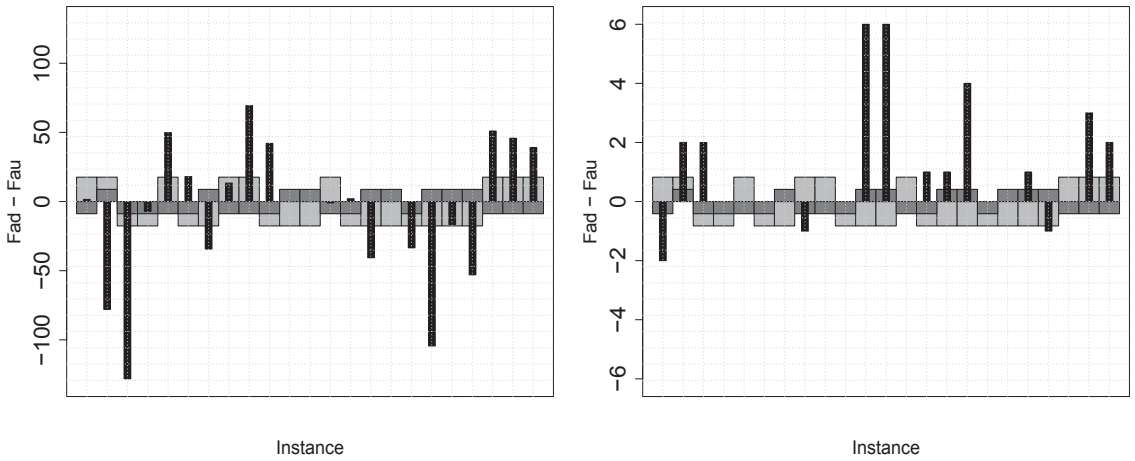
Table 8.7: Average performance values of the players with respect to preference. The third column contains the p-values of a paired two sample t-test between the two measurements across camera control type. The features for which the null hypothesis can be rejected with a 5% of significance (i.e. there is a significant effect of the camera control scheme on the feature) appear in bold.

The first two columns of Table 8.6 contain the mean values of each feature in the levels featuring the adaptive camera behaviour and the levels without. The last column displays the p-values of a two-sample paired t-test between the two groups of features.

These statistics reveal that, overall, the number of collected canisters and the number of falls is affected by the camera control paradigm as these are the only features for which it is possible to reject the null hypothesis.

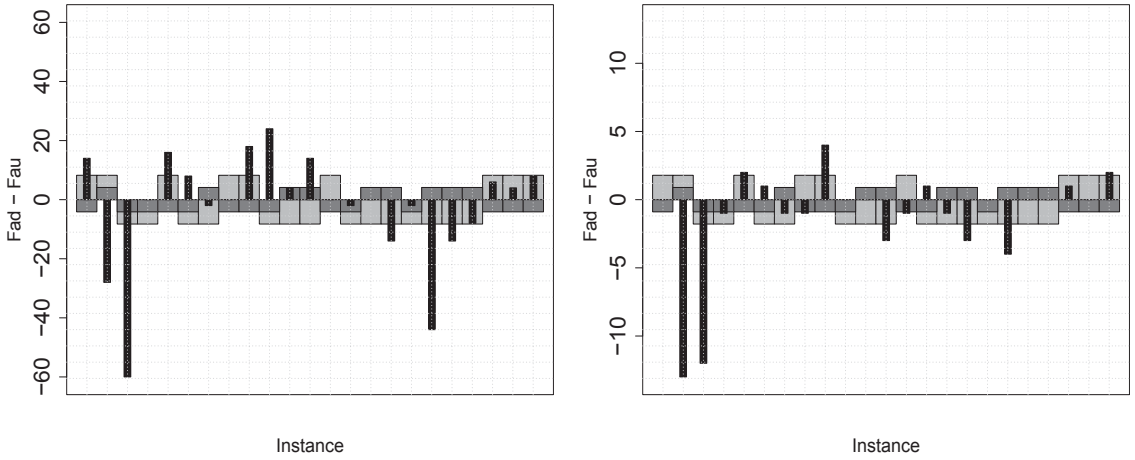
It is worth noting how both these features are not affecting the player preference, as displayed in Table 8.7; this appears to be another reason for the lack of a clear preference towards the adaptive version of the game. Completion time is, overall, the only feature affecting the players' preference (the players tended to prefer the level that took them less time to complete) and adaptivity did not affect this aspect of the players performances.

Within the group of novice players two different performance features yield a significant difference between the level types: the number of collected canisters and the number of falls. The number of canisters is significantly higher (154%) in the levels featuring adaptation and the number of falls decreases by 72% between the two types of levels. This finding reveals



(a) Δ Completion time

(b) Δ Number of fuel canisters collected



(c) Δ Number of jumps performed

(d) Δ Number of falls

Figure 8.7: Differences $\Delta F = F_{ad} - F_{au}$ of completion time (a), number of canisters collected (b), number of jumps (c) and number of falls (d) between the levels played with the adaptive camera behaviours and the ones without. The background color pattern shows which level was preferred and which level was played first for each game played. If the dark grey bar is in the upper half of the plot the level featuring the adaptive camera controller was preferred and vice versa. If the light grey bar is in the upper half of the plot the level featuring the adaptive camera controller was played first and vice versa.

that, for a part of the subjects, the adaptation mechanism allowed the automatic camera controller to overcome the limitations emerged in Chapter 7. Moreover, if we compare the average values for these two features, it is worth noticing how the adaptation mechanism improves the performance of the novice players getting them closer to the performances of

the other players and, above all, it allows the novice players to complete the level as the average number of canisters increases above 6 (the minimum number of canisters required to complete a level) and the completion time is approximately 40 second lower than the maximum level duration. Such an improvement is not noticeable for the expert players.

Moreover, when comparing the significant features that are presented in Table 8.6d and Table 8.6d, it appears that the adaptation of the camera behaviour supported the collection of fuel canisters and lead to a stronger preference of the novice players for the levels featuring adaptivity.

8.4 Summary

The chapter presented a case study that evaluated the applicability of the adaptive camera control methodology described in Chapter 4.

The camera behaviour is modelled using a combination of information about players' gaze and virtual camera position collected during a game experiment. The data collected is sorted into three sets of areas according to the game-play provided to the player. For each group the data is clustered using k-means to identify relevant behaviours and the relationship between the clusters and the game-play experience is modelled using three ANNs.

The evaluation of the ANN accuracy in predicting camera-behaviour is analysed with respect to the number and type of features used as input to the model. The analysis reveals that the best prediction accuracies (i.e. 76.43% for jump, 82.29% for collection and 70.04% for fight) are achieved using a representation of the past events which includes the previous area and the average features of the other previously visited areas. Moreover, sequential feature selection reduces vector size which results in higher accuracies for all ANNs.

The chapter concluded with the presentation of a user evaluation of the adaptive camera control architecture methodology. Through this experiment, the performance of the adaptivity and the impact of adaptivity on automatic camera control in third-person computer games are evaluated.

While the results show no clear preference for the levels featuring adaptivity, it appears that the adaptive camera controller is able to overcome some of the limitations emerged in the evaluation of the automatic controller. Moreover, the adaptation mechanism showed to be able to provide a satisfactory experience for most of the non-expert participants.

Chapter 9

Discussion and Conclusions

This thesis aimed to answer three main research questions: how to effectively solve the camera composition and animation problems in real-time in dynamic and interactive environments; how does automatic camera control affect the player experience in three-dimensional computer games; and, within the framework of automatic camera control, how can the player affect the cinematographic experience. In order to answer these research questions, this thesis pursued two objectives: develop and evaluate a real-time automatic camera controller, and design and evaluate a methodology for adaptive camera control. The rest of the chapter summarises the results presented in the thesis and discusses the extents of the its contributions and the limitations.

9.1 Contributions

This section summarizes this thesis' main achievements and contributions that advance the state-of-the-art in automatic camera control for interactive and dynamic virtual environments. To a lesser degree, results of this thesis can be of use to the fields of dynamic optimisation, animation and human-computer interaction. More specifically, the contributions of this thesis can be described as follows.

9.1.1 Dynamic Virtual Camera Composition

This thesis introduces a hybrid search algorithm, combining local and population-based search, for virtual camera composition in real-time dynamic virtual environments. The algorithm is a composite of a Genetic Algorithm employing an APF-based mutation operator. The proposed algorithm is evaluated and compared against state-of-the-art algorithms for optimisation in automatic virtual camera control.

To test this algorithm, we identify the building blocks of the camera control problem and we study the complexity of each. In particular, we identify three principal objective

functions, we relate these to the state-of-the-art definition of frame constraints and we systematically analyse the complexity of each objective function and their combination. As a result of this analysis we identified 11 test problems which represent a wide range of problem complexity values in terms of both static and dynamic optimisation.

These problems (see Section 6.2) compose a benchmark on which we evaluate the performance of the proposed hybrid meta-heuristic algorithm and compare it with other five search algorithms widely used in automatic camera control. The algorithms are compared on two aspects: accuracy and reliability. Accuracy is evaluated using the average best function value (ABFV) measure suggested by Schönemann (2007), while reliability is measured as the average amount of time in which the algorithm is unable to find a solution with an objective function value higher than 0.25.

The proposed search algorithm is able to find and track solutions with the highest accuracy and reliability on most test problems (see Section 6.4). Moreover, the results obtained by the hybrid Genetic Algorithm in the optimisation of the test problems including all the frame constraints are very positive: the algorithm shows the best accuracy and reliability independently of the number of subjects and the type of virtual environment. Such test problems are the most realistic and the most common encountered in interactive virtual environments, where most of the times the camera needs to follow an avatar and a number of other objects.

Moreover, the performance measures and the test problems employed in the evaluation of the hybrid Genetic Algorithm stand, to the best of the author's knowledge, as the first systematic benchmark for virtual camera composition in dynamic environments (see Section 6.2).

9.1.2 Real-Time Automatic Camera Control

The thesis introduces a novel automatic camera control architecture (see Chapter 3), that combines the aforementioned hybrid Genetic Algorithm and a lazy probabilistic roadmap to effectively solve the problems of composition and animation of the camera in dynamic and interactive virtual environments. The proposed approach employs the hybrid Genetic Algorithm to iteratively search the camera configuration that optimises the composition requirements at each frame. The lazy probabilistic roadmap algorithm is used to animate the camera to the new solution found at each iteration of the system. We evaluated the system's quality and impact on player experience by conducting a user study in which participants play a game in which they manually control the camera and play the same game featuring the automatic camera control system presented in this thesis.

The thesis presented a user evaluation of the proposed automatic camera control system in a commercial-standard three-dimensional computer game (see Chapter 7). Through this experiment, the performance of the camera control system and the impact of automatic camera control in third-person computer games were evaluated.

The pairwise preferences reported by the participants indicate that automatic camera control is perceived as an enhancement to the game by simplifying the interaction. Furthermore, the subjects reported no complaints on the camera behaviour, which, along with the dominating preference for automatic camera control, suggests that the camera control system was able to elicit a satisfactory experience without noticeable issues on the quality of the generated images and camera motion. The analysis of the features representing aspects of player performance in the game revealed that when players did not control the camera, they completed the level in less time, confirming our hypothesis that the automatic camera controller would positively influence the players' performance.

9.1.3 Adaptive Camera Control

This thesis introduced a methodology to build adaptive virtual camera control in computer games by modelling camera behaviour and its relationship to game-play (see Chapter 4). Camera behaviour is modelled using a combination of information about players' gaze and virtual camera position collected during a game experiment. The data collected is clustered using k-means to identify relevant behaviours and the relationship between the clusters and the game-play experience is modelled using three ANNs. These models of camera behaviour are then used to select — at specific times in the game — which camera profile to select in order to instruct the automatic camera controller and provide a personalised cinematographic experience.

The methodology is evaluated on a three-dimensional platform game and the model accuracy in predicting camera-behaviour is analysed with respect to the number and type of features used as input to the model (see Chapter 8). The analysis reveals that the best prediction accuracies are around 76% for areas in which the main task is to jump, 82% for the ones in which items collection is the main task and around 70% for fight areas. Such results are achieved using a representation of the past in-game events which includes game and player features of the previous area as well as the average features of the other previously visited areas. Moreover, sequential feature selection reduces the input space of the models which results in higher accuracies for all ANNs.

The models built from this data are employed for camera behaviour adaptation in the same game and the adaptation mechanism is evaluated in a user study. The pairwise preferences reported by the participants do not indicate a strong preference towards adaptive

camera control which is, in general, not perceived as an enhancement to the game. However, if the subjects are sorted by their playing skill level, it appears that the two groups of players with lower in-game performance significantly prefer adaptive camera control over simple automatic camera control, while the opposite effect is present for the expert players. Moreover, for the novice players the number of falls significantly decreases (by 106%) when the adaptive camera control is used. Such findings suggest that, for a part of the subjects, the adaptation mechanism allowed the automatic camera controller to overcome its limitations and to support a wider set of player actions.

9.2 Limitations

The limitations of the proposed approach to virtual cinematography, as appeared throughout the experiments of this thesis, are summarized in this section. Moreover the generality of the obtained experimental results is also discussed.

9.2.1 Methods

The first limitation of the camera control approach proposed emerges from the hybrid optimisation algorithm per se. The accuracy and reliability demonstrated by the algorithm in the optimisation of the projection size frame property for more than one subject are low as the algorithm is unable to provide an acceptable solution, on average, in 91% of the time. As shown in Section 6.4, similar performance is manifested also by the standalone APF algorithm, suggesting that the approximated derivative of the projection size objective function used to drive the APF operator misleads the search process. This aspect requires a further investigation, and a more precise force function needs to be designed. The hybrid GA appears also unable to accurately and reliably optimise the most dynamic of the problems considered: a vantage angle constraint on one subject in the forest environment. We believe that such a behaviour is the consequence of an imprecise detection of early convergence, which induces the scheduler to constantly switch between the application of the standard operators and the APF-based operator. A better early convergence detection mechanism would probably allow the algorithm to achieve better accuracy and reliability. This is a direction that should be considered in order to shed further light on the potential of hybrid optimisation for real-time camera composition.

A second limitation emerges from the user evaluation of the automatic camera controller presented in Chapter 7: a large amount of players had more difficulty to perform jumps. The reason for such a negative impact is probably the camera profile chosen, which did not support all the types of actions the player had to perform. This issue has been addressed by adapting the camera profile to the player's preferences and the type of interaction; as it

appears from the results of the evaluation of the adaptive camera controller, the number of falls decreases significantly when the game is played with a personalised camera.

While the incorporation of adaptation elements to automatic camera control appears to improve, to some extent, the performances of the players, the preferences expressed in the user evaluation reveal that the proposed adaptive camera controller is not successful in improving the experience for a large number of players. The data collected in the experiment presented in Chapter 8 suggest that the expert players systematically preferred the game without adaptation and their performance is mostly unaffected by the camera control scheme.

The reasons for such results are manifold and can probably be identified in all three stages of the adaptation methodology: the data collection experiment, behaviour modelling and camera profile adaptation.

The analysis of the data collected to build the camera behaviour models (see Section 8.1) suggests that the game used in the collection experiment is visually very complex and a multitude of objects have high visual saliency and compete for the player's attention. This is especially evident in the camera behaviour clusters in which the players do not focus primarily on the task-related objects but spend a large amount of time observing the virtual environment. This effect could be minimized using a different version of the game during the data collection phase, which incorporates textures with low saliency values.

While the accuracy of the models evaluated in Section 8.2 is relatively high, none of the models used in the evaluation has a prediction accuracy higher than 85%. This means that there is a substantial number of cases for which the models are unable to predict the right behaviour leading to a selection of incorrect camera profiles during the adaptation phase. Different directions could be pursued to increase these results, such as other representations, a larger data set, a global search based feature selection. A different clustering algorithm could also be employed, as the error might be a consequence of a wrong cluster separation. However, considering also that building generalised human behaviour models is a hard problem, it might be impossible to obtain better prediction results; therefore, the model itself could be adjusted during the gameplay, to better match the behaviour of each specific player.

Finally, the camera profile adaptation process employs an arbitrary method to translate the behaviours emerging from the data collected to camera profiles; the same criterion, probably, cannot be used for all the players.

9.2.2 Experiments

The game used for the evaluation of the proposed automatic camera controller and the adaptive camera control methodology has been designed to represent a large number of games and, therefore, allow the generalisation of the findings to other games. However, aspects such as the number of participants and the specific characteristics of the games need to be covered to evaluate the extent of the generality of the evaluation results.

In the evaluation presented in Chapter 7, the participants were asked two state their preference between a game played using automatic camera control and a game in which they were required to manually control the camera. In order to maintain a coherent control scheme, the participants are required to change the hand they use to control the avatar (the left one when the camera is controlled manually and the right one when the camera is automatically controlled). While this lack of symmetry might have influenced the participant's judgement, this scheme appeared the most natural during the pilot study conducted previous to the experiment. Other options would have required to participants to either change the controls completely or to interact from very uncomfortable positions. Moreover, only one participant mentioned the avatar control key mapping as uncomfortable and suggested the use of *W,A,S,D* keys.

The game interaction consists of four main activities: jump, fight, collect and explore. It is therefore possible to assume that the findings can be applied to any three-dimensional game in which the gameplay is based primarily on these aspects (e.g. platform games, action games or fighting games). Although these games often include a strong narrative component, which is almost completely absent on the test game, we still assume our findings to be valid for the rest of the game interaction. Moreover, the impact of automatic camera on narrative is largely assumed to be positive in the literature (Drucker et al., 1992; Charles et al., 2002; Christie et al., 2002) and a user survey by Jhala and Young (2010) confirms such an assumption.

Another critical aspect to estimate the validity of the findings is the size of the participants' sample. The number of subjects in our user studies is relatively small; however, considering that only one independent variable has been tested, that the findings are restricted to a specific typology of games and that the results appear to be largely in favour of automatic camera control, we believe the sample size to be sufficient. One or more larger user survey studies will be required to evaluate automatic camera control across more game genres.

9.3 Extensibility

The experiments evaluating the different aspects of this thesis work have demonstrated the solidity of the proposed solutions. Here, we discuss a number of research directions that could benefit from the findings of this thesis. Moreover, on the basis of the results obtained, we suggest new research objectives and possible solutions.

9.3.1 Benchmarking

One of the challenges encountered during this thesis work consisted of designing the evaluation of the methodologies proposed. In automatic camera control, in contrast to other fields such as computer vision, there is a lack of standard data corpora and measures which would allow to fairly evaluate and compare different approaches. This thesis proposes an initial benchmark and suggests a set of performance measures for the evaluation of virtual camera composition algorithms in dynamic environments. We believe that, starting from this initial experiment, a complete camera control benchmark can be developed and that this would be beneficial for future advancements in the field of both automatic camera control and dynamic optimisation.

9.3.2 Optimisation

The results demonstrated by the hybrid Genetic Algorithm proposed in this thesis are an encouraging starting point in the development of hybrid optimisation algorithms for dynamic camera composition. This aspect deserves further empirical analysis to identify the ideal combination of algorithms. Considering the high dynamism of the camera control objective function and its very rough landscape (see Chapter 6), an algorithm that successfully optimises this function could prove very effective in other dynamic optimisation problems (Branke, 2001).

Moreover, investigating mechanisms for dynamic population handling during the optimisation, could improve the start up performance and the problem of early convergence. This mechanism, as well as the best pattern of application of the genetic operators, could be learned using machine learning. For this purpose, the benchmark problems introduced in this thesis could be employed as a training set.

Finally, since the intrinsic shape of the the camera composition objective function contains multiple alternative solutions often visually different, such solutions can be used to generate multiple different shots from the same description. It is already known that from an optimization standpoint, these problems are usually multi-modal so that algorithms with restart and/or niching components are needed as we may otherwise miss the best optima.

Supported by some initial results (Preuss et al., 2012; Burelli et al., 2012), we believe it is worth to explore the objective function landscapes and see how the difficulty of finding multiple good shots varies across different scenes and problems in both static and dynamic environments.

9.3.3 Manual Control

The automatic camera controller's evaluation presented in Chapter 7, compares the impact of two camera control schemes on player preferences and performances. While, the results clearly show a strong preference of the players towards automatic camera control, it would be interesting to compare the proposed camera controller against manual camera control schemes other than the one used in this thesis (orbit camera). Such an extended study could lead to an empirical taxonomy of camera control schemes in different game genres.

9.3.4 Aesthetics

The evaluations of the automatic camera controller and the adaptation methodology have both revealed a very low number of subjects identifying aesthetics as a motivation for their preference. In particular, the lack of a reported aesthetic improvement between the game featuring manual camera control and the one employing the automatic camera controller suggests the necessity to extend the authors' evaluation of the ability of the system to provide well composed shots in dynamic environments (Burelli and Yannakakis, 2010a).

Furthermore, it appears very important to investigate the role of image composition and virtual camera control in computer game aesthetics it differences from photography and classing cinematography. For this purpose, an evaluation of the impact of different directorial styles on player experience in different game genres should be carried out; such a study could lead to the design of a structured taxonomy of interactive virtual cinematography.

9.3.5 Animation

The animation capabilities of the camera control architecture presented in this thesis are limited to the control of a few dynamic parameters of the camera such as speed and acceleration. These parameters showed to be sufficient as control attributes for a virtual camera of a 3D platform game; however, in order to generate more cinematographic experiences, we believe it is necessary to explore new virtual camera animation methods and the integration of trajectory prediction (Halper et al., 2001) algorithms in the path planning process, to reduce undesired discontinuities in the animation.

9.3.6 Adaptation

In the proposed methodology of camera behaviour adaptivity, the user models are built a-priory and do not change during the experience. By integrating a form of feedback about the user cognitive state would allow to adapt, not only the camera profile, but the model itself. Investigating the use of multiple input modalities, such as computer vision, electroencephalogram (EEG), electromyogram (EMG) and other physiological signals appears also a promising way to detect and model the user's affective state (Yannakakis et al., 2010), allowing to better understand the relationship between virtual cinematography and the player.

9.4 Summary

This dissertation has presented an approach to interactive virtual cinematography addressing the problems of efficient and effective camera placement and implicit camera interaction. The presented approach includes a novel architecture for automatic camera control, a novel hybrid optimisation algorithm and a methodology to design adaptive virtual camera experiences. These approaches have been described in detail and each one of the architecture's components has been evaluated in terms of both numerical performance and its effect on player experience. The key findings of the thesis suggest that, by embedding domain knowledge into a global optimisation algorithm through the Artificial Potential Field operator, the algorithm is able to deal with highly dynamic and multi-modal fitness functions. Moreover, the user evaluation of the proposed automatic camera control architecture, shows that the combination of hybrid optimisation and path planning allows to accurately and smoothly control the camera. Finally the suggested methodology for camera behaviour modelling and adaptation, proved to be able to overcome some of the limitations of pure automatic camera control.

Bibliography

Ali, M. M. and Törn, A. A. (2004). Population set-based global optimization algorithms: some modifications and numerical studies. *Computers & Operations Research*, 31(10):1703–1725.

Arijon, D. (1991). *Grammar of the Film Language*. Silman-James Press LA.

Bares, W. H. and Lester, J. C. (1997a). Cinematographic User Models for Automated Realtime Camera Control in Dynamic 3D Environments. In *International Conference on User Modeling*, pages 215–226. Springer-Verlag.

Bares, W. H. and Lester, J. C. (1997b). Realtime Generation of Customized 3D Animated Explanations for Knowledge-Based Learning Environments. In *Conference on Innovative Applications of Artificial Intelligence*.

Bares, W. H. and Lester, J. C. (1999). Intelligent multi-shot visualization interfaces for dynamic 3D worlds. In *International Conference on Intelligent User Interfaces*, pages 119–126, Los Angeles, California, USA. ACM Press.

Bares, W. H., McDermott, S., Boudreaux, C., and Thainimit, S. (2000). Virtual 3D camera composition from frame constraints. In *ACM Multimedia*, pages 177–186, Marina del Rey, California, USA. ACM Press.

Bares, W. H., Zettlemoyer, L. S., Rodriguez, D. W., and Lester, J. C. (1998). Task-Sensitive Cinematography Interfaces for Interactive 3D Learning Environments. In *International Conference on Intelligent User Interfaces*, pages 81–88, San Francisco, California, USA. ACM Press.

Barraquand, J. and Latombe, J.-c. (1990). A Monte-Carlo algorithm for path planning with many degrees of freedom. In *IEEE International Conference on Robotics and Automation*, pages 1712–1717.

Baumann, M. A., Dupuis, D. C., Leonard, S., Croft, E. A., and Little, J. J. (2008). Occlusion-free path planning with a probabilistic roadmap. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2151–2156. IEEE.

Beckhaus, S., Ritter, F., and Strothotte, T. (2000). CubicalPath - Dynamic Potential Fields for Guided Exploration in Virtual Environments. In *Pacific Conference on Computer Graphics and Applications*, pages 387–459.

Bernhard, M., Stavrakis, E., and Wimmer, M. (2010). An empirical pipeline to derive gaze prediction heuristics for 3D action games. *ACM Transactions on Applied Perception*, 8(1):4:1—-4:30.

Blinn, J. (1988). Where Am I? What Am I Looking At? *IEEE Computer Graphics and Applications*, 8(4):76–81.

Bohlin, R. and Kavraki, L. (2000). Path planning using lazy PRM. In *IEEE International Conference on Robotics and Automation.*, volume 1, pages 521–528. IEEE.

Bourne, O., Sattar, A., and Goodwin, S. (2008). A Constraint-Based Autonomous 3D Camera System. *Journal of Constraints*, 13(1-2):180–205.

Branke, J. (2001). *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers.

Burelli, P., Di Gaspero, L., Ermetici, A., and Ranon, R. (2008). Virtual Camera Composition with Particle Swarm Optimization. In *International symposium on Smart Graphics*, pages 130–141. Springer.

Burelli, P. and Jhala, A. (2009a). CamOn: A Real-Time Autonomous Camera Control System. In *AAAI Conference On Artificial Intelligence In Interactive Digitale Entertainment Conference*, Palo Alto. AAAI.

Burelli, P. and Jhala, A. (2009b). Dynamic Artificial Potential Fields for Autonomous Camera Control. In *AAAI Conference On Artificial Intelligence In Interactive Digitale Entertainment Conference*, Palo Alto. AAAI.

Burelli, P., Preuss, M., and Yannakakis, G. N. (2012). Optimising for Multiple Shots: An Analysis of Solutions Diversity in Virtual Camera Composition. In *FDG Workshop On Intelligent Camera Control and Editing*.

Burelli, P. and Yannakakis, G. N. (2010a). Combining Local and Global Optimisation for Virtual Camera Control. In *IEEE Conference on Computational Intelligence and Games*, page 403.

Burelli, P. and Yannakakis, G. N. (2010b). Global Search for Occlusion Minimisation in Virtual Camera Control. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Barcelona. IEEE.

Burelli, P. and Yannakakis, G. N. (2011). Towards Adaptive Virtual Camera Control In Computer Games. In *International symposium on Smart Graphics*.

Burelli, P. and Yannakakis, G. N. (2012a). Automatic Camera Control in Computer Games: Design And Evaluation. *Under Journal Review*.

Burelli, P. and Yannakakis, G. N. (2012b). Real-Time Virtual Camera Composition In Dynamic Environments. *Under Journal Review*.

Calinski, T. and Harabasz, J. (1974). A Dendrite Method For Cluster Analysis. *Communications in Statistics*, 3(1):1–27.

Cardamone, L., Loiacono, D., and Lanzi, P. L. (2009). On-line Neuroevolution Applied to The Open Racing Car Simulator. In *IEEE Congress on Evolutionary Computation*.

Caruana, R., Lawrence, S., and Giles, L. (2001). Overfitting in Neural Nets : Backpropagation, Conjugate Gradient and Early Stopping. In *Advances in Neural Information Processing Systems*, pages 402–408. MIT Press.

Charles, D. and Black, M. (2004). Dynamic Player Modelling : A Framework for Player-centred. In *International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 19–35.

Charles, F., Lugrin, J.-l., Cavazza, M., and Mead, S. J. (2002). Real-time camera control for interactive storytelling. In *International Conference for Intelligent Games and Simulations*, pages 1–4, London.

Christianson, D., Anderson, S., He, L.-w., Salesin, D. H., Weld, D., and Cohen, M. F. (1996). Declarative Camera Control for Automatic Cinematography. In *AAAI*, pages 148–155. AAI.

Christie, M. and Hosobe, H. (2006). Through-the-lens cinematography. In Butz, A., Fisher, B., Krüger, A., and Olivier, P., editors, *International symposium on Smart Graphics*, volume 4073 of *Lecture Notes in Computer Science*, pages 147–159–159, Berlin, Heidelberg. Springer Berlin Heidelberg.

Christie, M., Languénou, E., and Granvilliers, L. (2002). Modeling Camera Control with Constrained Hypertubes. In *International Conference on Principles and Practice of Constraint Programming*, pages 618–632, London, UK. Springer-Verlag.

Christie, M. and Normand, J.-M. (2005). A Semantic Space Partitioning Approach to Virtual Camera Composition. *Computer Graphics Forum*, 24(3):247–256.

Christie, M., Olivier, P., and Normand, J.-M. (2008). Camera Control in Computer Graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218.

Corke, P. I. (1994). Visual Control Of Robot Manipulators – A Review. In *Visual Servoing*, pages 1–31. World Scientific.

Davies, D. L. and Bouldin, D. W. (1979). A Cluster Separation Measure. *Pattern Analysis and Machine Intelligence*, (2):224–227.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Drachen, A., Canossa, A., and Yannakakis, G. N. (2009). Player modeling using self-organization in Tomb Raider: Underworld. In *IEEE Symposium on Computational Intelligence and Games*, pages 1–8. IEEE.

Drucker, S. M., Galyean, T. A., and Zeltzer, D. (1992). CINEMA: a system for procedural camera movements. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics*, page 67.

Drucker, S. M. and Zeltzer, D. (1994). Intelligent camera control in a virtual environment. In *Graphics Interface*, pages 190–199.

Dunn, J. C. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Cybernetics and Systems*, 3(3):32–57.

El-Nasr, M. S. and Yan, S. (2006). Visual Attention in 3D Video Games. In *ACM SIGCHI international conference on Advances in computer entertainment technology*.

Filliat, D. and Meyer, J.-A. (2002). Global localization and topological map-learning for robot navigation. In *From Animals to Animats*.

Floudas, C. A. and Pardalos, P. M. (1990). *A collection of test problems for constrained global optimization algorithms*. Springer-Verlag New York.

Frazzoli, E., Dahleh, M., and Feron, E. (2000). Robust hybrid control for autonomous vehicle motion planning. In *IEEE Conference on Decision and Control*, volume 1, pages 821–826. IEEE.

Gleicher, M. and Witkin, A. (1992). Through-the-lens camera control. In *ACM SIGGRAPH*, volume 26, page 331.

Halper, N., Helbing, R., and Strothotte, T. (2001). A Camera Engine for Computer Games: Managing the Trade-Off Between Constraint Satisfaction and Frame Coherence. *Computer Graphics Forum*, 20(3):174–183.

Halper, N. and Olivier, P. (2000). CamPlan: A Camera Planning Agent. In *International symposium on Smart Graphics*, pages 92–100. AAAI Press.

Hamaide, J. (2008). A Versatile Constraint-Based Camera System. In *AI Game Programming Wisdom 4*, pages 467–477.

Hart, P. E., Nilsson, N. J., and Bertram, R. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*.

Hastings, E. J., Guha, R. K., and Stanley, K. O. (2009). Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):1–19.

Haykin, S. (2008). *Neural Networks and Learning Machines*. Number v. 10 in Neural networks and learning machines. Prentice Hall.

He, L.-w., Cohen, M. F., and Salesin, D. H. (1996). The virtual cinematographer: a paradigm for automatic real-time camera control and directing. In *ACM SIGGRAPH*, pages 217–224. ACM Press.

Holland, J. H. (1992). Genetic Algorithms. *Scientific American*, 267:66–72.

Houlette-Stottler, R. (2004). Player Modeling for Adaptive Games. In *AI Game Programming Wisdom 2*, pages 557–566. Charles River Media, Inc.

Hubert, L. J. and Levin, J. R. (1976). Evaluating object set partitions: Free-sort analysis and some generalizations. *Journal of Verbal Learning and Verbal Behavior*, 15(4):459–470.

Jardillier, F. and Languénou, E. (1998). Screen-Space Constraints for Camera Movements: the Virtual Cameraman. *Computer Graphics Forum*, 17(3):175–186.

Jhala, A. and Young, R. M. (2005). A discourse planning approach to cinematic camera control for narratives in virtual environments. In *AAAI*, number July, pages 307–312, Pittsburgh, Pennsylvania, USA. AAAI Press.

Jhala, A. and Young, R. M. (2009). Evaluation of intelligent camera control systems based on cognitive models of comprehension. In *International Conference On The Foundations Of Digital Games*, pages 327–328.

Jhala, A. and Young, R. M. (2010). Cinematic Visual Discourse: Representation, Generation, and Evaluation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):69–81.

Jones, T. and Forrest, S. (1995). Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann.

Kavraki, L., Svestka, P., Latombe, J.-c., and Overmars, M. (1994). Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. In *IEEE International Conference on Robotics and Automation*, page 171.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *IEEE Conference on Neural Networks*, pages 1942–1948.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98.

Kittler, J. (1978). Feature Set Search Algorithms. *Pattern Recognition and Signal Processing*, pages 41–60.

Krzanowski, W. J. and Lai, Y. T. (1988). A criterion for determining the number of groups in a data set using sum-of-squares clustering. *Journal of Biometrics*, 44(1):23–34.

Kyung, M. and Kim, M. (1995). Through-the-lens camera control with a simple jacobian matrix. In *Graphics Interface*.

Land, M. F. and Tatler, B. W. (2009). *Looking and Acting: Vision and Eye Movements in Natural Behavior*. Oxford University Press.

Lino, C., Christie, M., Lamarche, F., Schofield, G., and Olivier, P. (2010). A Real-time Cinematography System for Interactive 3D Environments. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 139–148.

Lucas, S. M. (2005). Evolving a Neural Network Location Evaluator to Play Ms . Pac-Man. In *IEEE Symposium on Computational Intelligence and Games*.

Lucas, S. M. (2008). Computational intelligence and games: Challenges and opportunities. *International Journal of Automation and Computing*, 5(1):45–57.

Mackinlay, J. D., Card, S. K., and Robertson, G. G. (1990). Rapid controlled movement through a virtual 3D workspace. In *ACM SIGGRAPH*, volume 24, pages 171–176, Dallas.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 233, pages 281–297.

Mahlmann, T., Drachen, A., Togelius, J., Canossa, A., and Yannakakis, G. N. (2010). Predicting Player Behavior in Tomb Raider: Underworld. In *IEEE Conference on Computational Intelligence and Games*.

Mahlmann, T., Togelius, J., and Yannakakis, G. N. (2011). Towards Procedural Strategy Game Generation : Evolving Complementary Unit Types. In *European Conference on Applications of Evolutionary Computation*.

Marchand, E. and Courty, N. (2002). Controlling a camera in a virtual environment. *The Visual Computer*, 18:1–19.

Martinez, H. P., Jhala, A., and Yannakakis, G. N. (2009). Analyzing the impact of camera viewpoint on player psychophysiology. In *International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE.

Moravec, H. and Elfes, A. E. (1985). High Resolution Maps from Wide Angle Sonar. In *IEEE International Conference on Robotics and Automation*, pages 116 – 121.

Munoz, J., Yannakakis, G. N., Mulvey, F., Hansen, D. W., Gutierrez, G., and Sanchis, A. (2011). Towards Gaze-Controlled Platform Games. In *IEEE Conference on Computational Intelligence and Games*, number Section V, pages 47–54.

Nacke, L., Stellmach, S., Sasse, D., and Lindley, C. A. (2009). Gameplay experience in a gaze interaction game. In *Communication by Gaze Interaction*, pages 49–54.

Nieuwenhuisen, D. and Overmars, M. H. (2004). Motion Planning for Camera Movements in Virtual Environments. In *IEEE International Conference on Robotics and Automation*, pages 3870 – 3876. IEEE.

Olivier, P., Halper, N., Pickering, J., and Luna, P. (1999). Visual Composition as Optimisation. In *Artificial Intelligence and Simulation of Behaviour*.

Oskam, T., Sumner, R. W., Thuerey, N., and Gross, M. (2009). Visibility transition planning for dynamic camera control. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–65.

Phillips, C. B., Badler, N. I., and Granieri, J. (1992). Automatic viewing control for 3D direct manipulation. In *ACM SIGGRAPH Symposium on Interactive 3D graphics*, pages 71–74, Cambridge, Massachusetts, USA. ACM Press.

Picardi, A., Burelli, P., and Yannakakis, G. N. (2011). Modelling Virtual Camera Behaviour Through Player Gaze. In *International Conference On The Foundations Of Digital Games*.

Pickering, J. (2002). *Intelligent Camera Planning for Computer Graphics*. PhD thesis, University of York.

Pinelle, D. and Wong, N. (2008). Heuristic evaluation for games. In *CHI*, CHI '08, page 1453, New York, New York, USA. ACM Press.

Pontriagin, L. S. (1962). *Mathematical Theory of Optimal Processes*. Interscience Publishers.

Preuss, M., Burelli, P., and Yannakakis, G. N. (2012). Diversified Virtual Camera Composition. In *European Conference on the Applications of Evolutionary Computation*, Malaga.

Riedmiller, M. and Braun, H. (1993). *A direct adaptive method for faster backpropagation learning: the RPROP algorithm*. IEEE.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Nature*, 323:533–536.

Salichs, M. A. and Moreno, L. (2000). Navigation of mobile robots: open questions. *Robotica*, 18(3):227–234.

Schmalstieg, D. and Tobler, R. F. (1999). Real-time Bounding Box Area Computation. Technical Report Figure 1, Vienna University.

Schönemann, L. (2007). Evolution Strategies in Dynamic Environments. In Yang, S., Ong, Y.-S., and Jin, Y., editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 51–77. Springer Berlin Heidelberg, Berlin, Heidelberg.

Shaker, N., Yannakakis, G. N., and Togelius, J. (2010). Towards automatic personalized content generation for platform games. In *AAAI Conference On Artificial Intelligence In Interactive Digitale Entertainment*.

Shannon, C. E. (1950). Programming a computer for playing chess. *Philosophical Magazine Series 7*, 41(314):256–275.

Snedecor, G. W. and Cochran, W. G. (1989). *Statistical Methods*. Number vb. 276 in Statistical Methods. Iowa State University Press.

Soanes, C. and Stevenson, A. (2005). *Oxford Dictionary of English*. Oxford University Press.

Storn, R. (1996). On the usage of differential evolution for function optimization. In *North American Fuzzy Information Processing*, pages 519–523. IEEE.

Storn, R. and Lampinen, J. A. (2004). Differential Evolution. In *New Optimization Techniques in Engineering*, chapter 6, pages 123–166. Springer-Verlag.

Storn, R. and Price, K. (1997). Differential Evolution A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359–359.

Sundstedt, V., Stavrakis, E., Wimmer, M., and Reinhard, E. (2008). A psychophysical study of fixation behavior in a computer game. In *Symposium on Applied perception in graphics and visualization*, pages 43–50. ACM.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press.

Tesauro, G. (1994). TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation*, 219:215–219.

Thawonmas, R., Kurashige, M., Keita, I., and Kantardzic, M. (2006). Clustering of Online Game Users Based on Their Trails Using Self-organizing Map. In *International Conference on Entertainment Computing*, number 16500091, pages 366–369.

Thrun, S. (1995). Learning To Play the Game of Chess. *Advances in Neural Information Processing Systems*, 7.

Thue, D., Bulitko, V., Spetch, M., and Wasylishen, E. (2007). Interactive Storytelling : A Player Modelling Approach. In *AAAI Conference On Artificial Intelligence In Interactive Digitale Entertainment*, number July, pages 43–48, Stanford, CA.

Thurau, C., Bauckhage, C., and Sagerer, G. (2003). Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behavior for a Commercial Game. In *GAME-ON*, pages 119–123.

Thurau, C., Bauckhage, C., and Sagerer, G. (2004). Learning human-like movement behavior for computer games. In *International Conference on Simulation of Adaptive Behavior*, pages 315–323.

Togelius, J., Karakovskiy, S., and Koutn, J. (2009). Super Mario Evolution. In *IEEE Symposium on Computational Intelligence and Games*, pages 156–161.

Togelius, J., Nardi, R. D., and Lucas, S. M. (2006). Making Racing Fun Through Player Modeling and Track Evolution. In *SAB Workshop on Adaptive Approaches to Optimizing Player Satisfaction*.

Togelius, J., Yannakakis, G. N., Stanley, K. O., and Browne, C. (2010). Search-based Procedural Content Generation. In *European Conference on Applications of Evolutionary Computation*, pages 1–10.

Tomlinson, B., Blumberg, B., and Nain, D. (2000). Expressive autonomous cinematography for interactive virtual environments. In *International Conference on Autonomous Agents*, page 317.

Törn, A., Ali, M., and Viitanen, S. (1999). Stochastic Global Optimization: Problem Classes and Solution Techniques. *Journal of Global Optimization*, 14(4):437–447–447.

Ware, C. and Osborne, S. (1990). Exploration and virtual camera control in virtual three dimensional environments. *ACM SIGGRAPH*, 24(2):175–183.

Wolf, M. J. P. (2001). Genre and the video game. In Wolf, M. J. P., editor, *The medium of the video game*, chapter 6, pages 113–134. University of Texas Press.

Yannakakis, G. N. and Hallam, J. (2011). Rating vs. Preference: A comparative study of self-reporting. In *Affective Computing and Intelligent Interaction Conference*. Springer-Verlag.

Yannakakis, G. N., Hallam, J., and Lund, H. H. (2008). Entertainment capture through heart rate activity in physical interactive playgrounds. *User Modeling and User-Adapted Interaction*, 18(1-2):207–243.

Yannakakis, G. N. and Maragoudakis, M. (2005). Player modeling impact on players entertainment in computer games. In Ardissono, L., Brna, P., and Mitrovic, A., editors, *International Conference on User Modeling*, volume 3538 of *Lecture Notes in Computer Science*, pages 74–78, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yannakakis, G. N., Martinez, H. P., and Jhala, A. (2010). Towards Affective Camera Control in Games. *User Modeling and User-Adapted Interaction*.

Yannakakis, G. N. and Togelius, J. (2011). Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, pages 1–16.

Yarbus, A. L. (1967). *Eye movements and vision*. Plenum press.