13. Audio visual analysis of player experience

Feedback-based gameplay metrics

RAPHAËL MARCZAK AND GARETH R. SCHOTT

This chapter introduces readers to an innovative method designed specifically to meet game researchers' need to be able to analyse and explore players' experiences with any off-the-shelf PC game. We introduce how insights into players' experiences are achieved from automatically processing significant elements of the audio-visual feedback (i.e., moving image and sound) produced by the game.

Videogames contain a wealth of significant and (behaviourally) influential information that is conveyed to the player to facilitate advancement through a game. Such information can relate to virtual progress (e.g., maps, mission logs), player reserves (e.g., acquisitions) or vitality (e.g., health or stamina levels), to name but a few. By exploiting and analysing what is transmitted to the player during play, feedback-based gameplay metrics utilise the same content that the player comes into contact with during gameplay, therefore tying the method directly to individual *player experiences*. A distinguishing feature of this approach, when compared to existing methods available for gathering gameplay metrics relates to the way a feedback-based gameplay metric approach operates as a post-processing method that works with gameplay once it has been captured and saved as an audio-visual file. We aim to highlight the advantages of this approach in terms of the way it allows the researcher to revisit and deconstruct a gameplay session from a number of different angles, combining various different audio-visual processing techniques to answer either pre-defined or supplementary research questions. Examples of the different means of extracting information from the audio and visual content of videogames are presented to provide a strong sense of the capabilities of this particular method. The aim is to outline its contribution and value as a stand-alone method, although the method is also designed to work effectively within a mixed methods approach (see Schott, et al., 2013).

When confronted with data in the form of an audio-visual file that contains footage documenting hours of gameplay completed with a commercially available off-the-shelf game title, that is also the outcome of a player's distinct approach or playing style and determined by individual differences in learning style, comprehension and perception to name but a few variables, the task of understanding player experience constitutes a highly complex task. In order to be able to address this complex task, we approach gameplay as a predominantly *configurative practice* (Vught, et al., 2012), in that we acknowledge the way gameplay requires the player to "work with the materiality of a text, [that is,] the need to participate in the construction of its material structure" (Klevjer, 2002). In other words, we understand game-

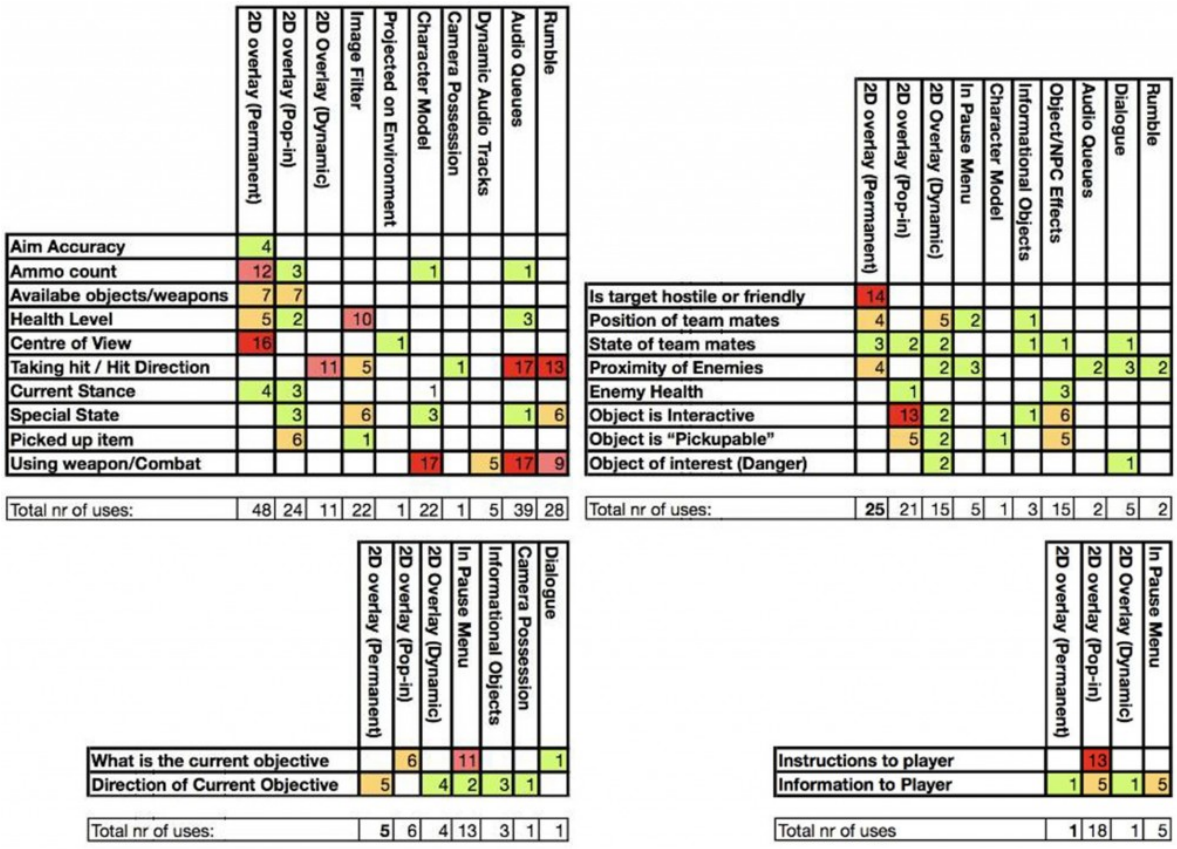
play as an encounter between a player and a game system. In adopting this perspective, our approach to understanding player experience is guided, firstly, by the need to be able to break down a game's core structural features (Zagal, et al., 2008; chapters 3–4) before, secondly, mapping a player's relationship with those structures. The object of our analysis is the *result* of specific behavioural choices and action selections during the course of play that carry *implications* spatially (e.g. pathways taken, navigational choices and progression) and temporally (e.g., pace and the difference between voluntary versus an imposed suspension of forward momentum). In order to operationalize this process during research and achieve a method capable of identifying the relationship between player input and a game's structure, we have found it necessary to 1) characterise the structure of a game as a multimedia document (segmentation) before 2) generating a description of its content (indexing). The first task is a conceptual exercise that demands that the structure of specific games is identified, while the second requires a means of extracting desired information from game content. This process demands conceptual knowledge derived from game studies combined with the capacity to extract the required information from the game system, derived from the computer sciences. The process outlined briefly here in combination with the different scholarly traditions that it draws upon, reflects how this variety of game metrics is conceived as a game research tool rather than a tool for evaluating usability with the aim of contributing to game development processes.

Expanding game metrics

The potential connected with the appropriation of conventional methods of gathering game metrics for player experience research (as distinct from user experience research) has been both compelling (Nacke, et al., 2009) and well received by the wider game studies research community. It is safe to say that the current value of game metrics to player research is not that dissimilar however to its initial value to user research, in that its appeal can be traced to its standing as a quantitative method. Within user research, game metrics represented advancement in accuracy and process on the qualitative methods that were more typically employed to articulate user-experiences with game systems under development (Hilbert and Redmiles, 2000). A similar appeal has seen game metrics extend beyond computer sciences and enter the humanities inaugurated discipline of game studies. Game metrics have come to represent a potentially powerful technique that holds the ability to transform what has been conceptualized or theorized to-date in relation to player experience. Here we refer specifically to the value of game metrics for explaining the exact nature and function of player-system relations during play (Choi, 2000; Drachen, 2008; Waern, 2012). Typically, gameplay metrics are capable of providing researchers with information on player activity in relation to specific areas such as: "navigation, item- and ability use, jumping, trading, running and whatever else players actually do inside the virtual environment" (Drachen, et al., 2013, p.10). In addition, they are capable of covering the wider conditions of managing play that absorbs "all interaction the player performs with the game interface and menu" (p.11) termed interface metrics.

With the exception of recording player input, such as pressing keys on a keyboard (or keystrokes) that can be recorded using any type of key logger software system, the gathering of gameplay metrics typically prescribes a certain level of access to gaming software that is authorized or sanctioned by game developers. This can take the form of access to the game source code, through agreements, modding provision (Sasse, 2008) or if a release takes the form of an open source game (Pedersen, et al., 2010). The approach outlined in this paper seeks to circumvent these conditions by instead utilizing and translating video and sound feedback experienced by the player during play. To this effect, Fagerholt's (2009)

research provided a strong indicator of the potential that exists for focusing on the audio-visual feedback provided to the player during play. From a review of First Person Shooter (FPS) games he was able to identify a number of game elements and outline the way information is broadcast to the player (see figure 1). Focusing on the *heads up display* (HUD) Fagerholt was able to draw attention to the manner in which videogames compensate for the player's physical disconnection from the game world requiring the game system to construct a sense of presence.



	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	Image Filter	Projected on Environment	Character Model	Camera Possession	Dynamic Audio Tracks	Audio Queues	Rumble
Aim Accuracy	4									
Ammo count	12	3				1			1	
Availabe objects/weapons	7	7								
Health Level	5	2		10					3	
Centre of View	16				1					
Taking hit / Hit Direction			11	5			1		17	13
Current Stance	4	3				1				
Special State		3		6		3			1	6
Picked up item		6		1						
Using weapon/Combat						17		5	17	9
Total nr of uses:	48	24	11	22	1	22	1	5	39	28

	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	In Pause Menu	Character Model	Informational Objects	Object/NPC Effects	Audio Queues	Dialogue	Rumble
Is target hostile or friendly	14									
Position of team mates	4		5	2		1				
State of team mates	3	2	2			1	1		1	
Proximity of Enemies	4		2	3				2	3	2
Enemy Health		1					3			
Object is Interactive		13	2			1	6			
Object is "Pickupable"		5	2		1		5			
Object of interest (Danger)			2						1	
Total nr of uses:	25	21	15	5	1	3	15	2	5	2

	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	In Pause Menu	Informational Objects	Camera Possession	Dialogue
What is the current objective		6		11			1
Direction of Current Objective	5		4	2	3	1	
Total nr of uses:	5	6	4	13	3	1	1

	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	In Pause Menu
Instructions to player		13		
Information to Player	1	5	1	5
Total nr of uses	1	18	1	5

Figure 1. Erik Fagerholt and Lorentzon's (2009) summary of information broadcast using HUD elements within 19 FPS games.

All of the elements identified in Fagerholt's research refer to graphical and sound streams, which led us to consider the possibilities of capturing and measuring this information automatically. Here we refer to the potential application of algorithms that have been created to detect the presence of specific objects within moving-image (Bay, et al., 2008), automatically segment moving-image into sub-scenes (Huang and Liao, 2001; Saraceno and Leonardi, 1997) or extract symbolic information such as annotations (Chen, et al., 2001) and numbers (Ye, et al., 2005).

Additionally, signal processing of audio streams constitutes an established and active research field within the computer sciences. Algorithms have been created to detect speech and music in radiophonic streams (Richard, et al., 2007), music recognition (Orio, 2006) or to indicate moments of intensity in movies via audio tempo analysis (Yeh, et al., 2009). In processing digital games we have found it possible to adapt such algorithms as they display large amounts of information in abstract and schematic ways (e.g., via life-bar representation, icons of obtained objects, on-screen blur, screen desaturation,

heartbeat sfx). Our method is based on the idea that different audio and visual design elements can be extracted accurately from output streams and analyzed to create a picture of the nature and developmental changes in players' interaction and experience with a particular game.

Segmentation and indexing gameplay performance

Gameplay *segmentation* has already formed the focus of research by Zagal, et al. (2008) as part of the *Game Ontology Project*. In this context we share the same understanding of segmentation as a process in which "a game is broken down into smaller elements of gameplay" (Zagal, et al., 2008, p.176). In their work, Zagal, et al., were clear to define the role of segmentation as an exploration of the structure of gameplay that supports an analysis of the role of design elements. We, however, seek to extend this process to encompass gameplay as both representative of *a* game system and *a* performance (Laurel, 1993). That is, we seek to segment games based on how players engage with the game structure and the possibilities offered by it rather than analyse a game for its structure and how it is likely to create specific gameplay experiences. In this context, performance becomes critical as it emphasizes the unfolding nature of games and relevance of player input to that process. The role of the player then becomes something more than just a necessary component to activate the game system (Aarseth, 2007).

Zagal, et al. (2008) have segmented *gameplay* on the basis of their temporal, spatial and challenge characteristics (see also, chapter 4). Yet, in illustrating their approach they apply their framework to vintage arcade games, games that foreground the rule system by virtue of their simplicity. This inevitably leads them to concede that contemporary games are likely to include "multiple forms of segmentation, that are interrelated, or even co-occur," (p.178) with novel game design also likely to require further ways of segmenting gameplay that may in turn call for a re-examination of existing segmentation principles. In this way, Zagal et al. acknowledge how such processes are required to evolve, or demand a more open-ended approach. By using a player performance determined segmentation process we aim to achieve this, in doing so, utilizing structure to achieve a segmentation that isolates relevant player experience.

When tailored to the experience of playing a game, segmentation needs to be based on portions of play that can be clearly identified by the homogeneity of the way breaks in the play experience present themselves to the player. Games are sub-divided in many different ways, the most basic being a new level or mission (e.g., *Battlefield 3* [Electronic Arts, 2011]) that can be accompanied by a pause whilst the next section of the game is loading. In doing so, a splash screen is often presented to players that can then be detected and time-stamped and used to illustrate the pace at which a player has progressed through the game. We use the term indexing to distinguish the identification and location of information that denotes a) where in the structure of the system the player is active, for example, in-game verses menus, b) the nature of the player's involvement, for example, Calleja's (2011) distinction between narrative and ludic involvement, or c) the degree of interactivity, for example, fully, semi or non-interactive. To summarise, segmentation is therefore the determination of the boundaries of a coherent section of play that is also comprised of a set of indexical properties. For example, the presence of a cut-scene can often represent the end of a large section of play and beginning of a new one, with the content of a cut-scene often containing a significant plot points that drive the change (segment). This event also denotes a distinction between the ludic and narrative involvement of the player and degree of interactivity of the player (index). Given our focus on player interaction, as indicative of player experience, this required that the algorithms employed to automatically segment game footage were capable of detecting indexi-

cal differences between interactivity and non-interactivity of different sequences. This was necessary in order to avoid incorrectly detecting actions on-screen like fighting, looting, walking as play when they occur in a sequence such as a non-interactive cut-scene.

In order to be able to reach more fine-grained aspects of a play experience, we conceptualize and divide the segmentation process into a multi-layered process. The layers, listed below, are employed in two different ways, the first, relating to the *process* of segmentation in which audio-visual footage of a gameplay session is processed or deconstructed as part of a method. Secondly, once this is completed, aspects of player experience can then be reconstructed using the layers to discern the meaning of a section of gameplay as part of the process of analysis. The five layers are:

- game system
- game world
- spatial-temporal
- degree of freedom
- interaction

The first step in our process is to acknowledge and treat the game system as a whole. That is, the initiation of gameplay, as the diegetic experience of playing in a fiction world, only occurs once players move from splash screens (e.g., copyright, production credits) to eventually reach a higher order main menu where players are able to activate play and enter the game world. Only when play is initiated does the player move from the *game system* layer to the *game world* layer, the 3D space in which the game is situated and play is realized. From that point onward, play is either broken or paused by the player, typically exiting play through higher order menus. The game world layer contains what we term *instances* of gameplay. During post-processing audio-visual analysis of such instances the player is present only as the entity behind, and responsible for generating and triggering the game footage under examination. The first key task in this process is to distinguish between in–out game and active–inactive as well as what this entails in terms of coherence between two consecutive frames. After this we begin to distinguish the *spatial-temporal* information contained within the game world layer as we identify deliberate pausing or detachments from the game world by the player, or information that is indicative of player progress relating to terrain traversed or activities completed that might trigger cut-scenes or new missions via a loading screen. These elements constitute identifiable nodes that map the progress and journey of the player and also the timing of when players experience core events in the game (useful for cross-player comparisons). Related to player progression through a game are the *degrees of freedom* and *interactivity* layers that constitute the manner in which the logic and rule system of the game is conveyed to the player and the degree to which the player is required to engage with the information provided by the game, or is permitted to ignore cues provided by the system.

Example 1: Looting in *Bioshock 2*

To begin defining the contribution and function of each layer we present the example of the act of *looting* in *Bioshock 2* (2K Games, 2010) and examine its relevance as a source of information in terms of its meaning within each of our five layers. In this process the layers do not function in a hierarchical top-down manner, meaning that we arrive at the act of looting as an end result of a process of refinement, but instead, the act of looting can be understood in terms of its relationship to each of the layers. In this

way the layers function is transversal, allowing for movement, meaning and implication to be derived from each one.

Looting a corpse in FPS body-horror game *Bioshock 2*, allows the player to gather resources, such as money, from deceased non-player characters. The player's attention is drawn to the possibility of looting by the appearance of a looting panel that overlays the game world and signals that there are items available to be looted. The appearance of a looting panel on-screen can be identified using the audio and video feedback streams. Additionally, a money logo accompanied by a sound file validates that money has been acquired if such an action is selected by the player. The presence of the looting panel, the money logo and the acquisition sound indicate an action has been performed in line with the cue provided. Whereas a looting panel and no money logo or acquisition sound indicates a player has not opted to loot in that instance (*interaction*).

The option to loot is built into the game system, providing a mechanism for the player to collect and store items and money that can be applied (at the player's discretion) in playing the game. The player is afforded the freedom of choice to loot or not loot. However, this option is also conditional on the game system allowing the player to explore the game space. That is, in interacting with the game space, player movement combined with first person perspective will lead the player to pass corpses on the ground. Proximity and gaze cues the game system to invite the player to loot by presenting the contents held by the corpse. Additionally, opting to not loot (in the here and now) may impact on players' *degree of freedom* further into the game world, giving meaning to the extent to which a player loots or not throughout the game.

Corpses are only found when the player moves through and acts inside the fictional space of the game (the city of Rapture in the case of *Bioshock 2*). It is not available to players in other spaces, like the help menu, loading screen or pause menu. As an act completed during gameplay, it is possible to map the instances of looting as a value of *spatial and temporal* movement. That is, how quickly or slowly a player progresses through the game and how they navigate the game.

Engagement with objects and structures within the city of Rapture is not offered to players until they decide to actually start the game and remain available only in the diegetic space of the game world (as distinct from higher order menus). It is possible to therefore identify when a player is in or out of the game world with reference to such acts such as looting.

Looting interactions are clearly part of the game system and the outcome of which are represented, collated, viewed and managed in higher order menus, representing actions that have been made in the context of 'game world' 'interactions' that can be related to 'degrees of freedom' afforded by the system but also are conditional on the degrees of freedom that a player possesses in future actions.

Example 2: Quick Time Event in Battlefield 3

In this second example, using *Battlefield 3* (Electronic Arts, 2011) we work back through the layers in the opposite direction. The game contains Quick Time Events (QTEs) that force the player to complete a series of rote-based actions (e.g., press E, left click mouse, then right click mouse). These prompts from the system are not presented to the player in a diegetic or narrative form, but remain procedural only really acknowledging the need for player input. In the context of QTEs the player temporarily loses all

other agency possibilities (i.e. they are unable to move freely or use strategy or weapons of choice). The degree of freedom becomes highly prescriptive, as the system (which is always in control of such conditions) is much more explicit in its treatment of the player requiring the necessary input to activate content and progress gameplay. Each interaction is preceded by an on-screen prompt (or video feedback stream from the perspective of our metric method), that indicates the action required (e.g., a blue icon matching the expected player input, E, mouse icon with left or right highlighted). Should the player follow this prompt with the correct input, the icon will then blink in blue in response as means of validating the player's action. Failure to follow the prompt will lead to a red icon, indicating that a response was either incorrect or absent.

The *interactions* defined by their *degrees of freedom*, are built into the game system as a form of mini-game (a task outside what one might expect in an FPS game environment) that is defined by success or failure, upon which progression is conditional and non-negotiable. As a marker of player progression, when a QTE occurs for the player it is also indicative of space and time. That is, specific QTEs (like missions or levels) are conditional on players' ability to reach specific locations on a game map, but also indicative of how long it takes a player to reach these nodes within the game. A QTE will therefore be triggered only once a player has reached a pre-defined point in the game, and should the player succeed, the same QTE will not reappear in that version of the game again. To this degree, the time taken to activate different QTEs provide a marker of pace and rate of progression attributable to the levels of mastery possessed by the player, or nature and style of game-playing (e.g. exploratory and thorough verses action and goal oriented). Lastly, whilst an obvious statement, QTEs are part of the game world and therefore cannot appear should a player activate a pause or opt to manage the conditions of play through engaging in higher order menus. This provides a clear indicator for automatic processing of a game's audio-visual feedback as to when QTEs materialize for the player and the nature and degree of player activity that the player experiencing when QTEs occur.

Automatic processing

The chapter now moves on to demonstrate examples of audio and video processing algorithms that are being appropriated and used to gather information on player experience. The aim of automatic processing is obviously to reduce the complexity and time-demanding task of segmenting a *gameplay performance* manually. The algorithms introduced in this chapter cover the detection of both static and moving information sources, assessing bar progressions (e.g. health bars) and identifying specific sounds. It is worth noting that when algorithms are applied, they are done so with knowledge of the value of symbolic information in terms of what it signifies in relation to player activity and involvement. For example, when a logo-detection algorithm is applied to footage of gameplay, in order to confirm the presence of a logo the intent is not to learn about the morphological properties of the logo (for instance colour, size, shape, potential text content) but the meaning of its presence or absence. For instance, the presence of a HUD on-screen signifies gameplay is in session, whereas its absence can signify other events are in progress (e.g., a cut-scene). Similarly with sound detection, the intent is not to analyse the sound itself (in terms of frequency, amplitude or tonality) but the function of the sound (e.g., it might signify successful object acquisition). The researcher is therefore required to engage with the game under consideration in advance of processing in order to elicit the key gameplay concepts and match them to their various audio-visual representations.

The implementation of audio-visual processing algorithms, as described in this chapter, is an accessible process that one can repeat using the open source library *OpenCV* (Intel Corporation, Willow Garage and Itseez, 2014) for computer vision processing, or *libsndfile* (de Castro Lopo, 2011) for audio processing.

TREATING AUDIO-VISUAL PRESENTATION AS DATA

An *image* can be represented as a *matrix* or two-dimensional array of values matrix (Gonzales and Woods, 2007, chapter 2). In this case the value is represented by a *pixel* that represents both a colour value and assumes a specific location inside the *matrix*. The process of partitioning and translating an image into a set of values for processing is termed *discretization* (representing a transformation from continuous representation to discrete representation), or *sampling*. This process is also applied to moving image and sound (Roads, 1996, chapter 1) in order to create a database of elements that are accessible, readable and measureable.

In the case of an image, there is no immediate link between the physical dimensions of an image (width and height in cm) and the digital dimensions that are assigned to it (number of columns and rows). Indeed, the number of pixels can vary for the same image depending on the quality and detail of processing and level of access required. The image as a *matrix* of pixels allows for the position of each pixel to be plotted. In addition to the locality of a particular pixel, information on the colour represented by each pixel becomes significant in image processing.

Quantization represents the process of assigning a value to a pixel. For colour images, a pixel will be assigned three values (generally ranging between 0–255) that provide information on the ratio of red, green and blue (or RGB) within any colour. For instance, <255, 0, 0> represents red, <0, 255, 0> represents green, <0, 0, 255> represents blue, while <255, 255, 0> represents yellow. Alternatively, values can be assigned based on hue, saturation and value (or HSV). The key advantage of HSV in comparison to RGB, is that the former comes closest to human perception of colour, meaning that two close pixel values are more likely to look and be categorised as the same colour. By contrast, RBG contains much more variation, as it distinguishes minor changes in the ratio, whereas shades of the same colour will invariably contain the same hue value, allowing the presence or absence of a colour on-screen to be identified more easily.

Moving-images can be converted and conceptualised as a sequence of images that are temporally organized. In a similar manner to the way that the number of pixels used to define a physical area depends on the desired detailed required for the *discretization* process, there is no standard for how many images are needed to represent one second of video. While 25 to 30 images per second is usually used, this will vary depending on the function and intended usage of the video-image. For example, surveillance cameras can function effectively operating at 15 images per second, or cinematic presentation can rise to 48 images per second, while videogames that sometimes needs to run at 60 images per second. Each image is termed a *frame* that can be assigned a frame number. So, in a 30 frames per second (FPS) video, frame 90 represent the frame initiating the third second of the video. Once translated in this manner, a frame can be processed as a regular image, using *matrix* and *colour* representation as described above.

Finally, the notion of a *mask image* probably requires some introduction, as all the video-based algorithms presented bellow require a *mask image* as one of their key inputs. A *mask image* describes a means of specifying a sub-area of interest in an image or a video for processing. A *mask image* represents a

binary image comprised of black and white (without grey nuances). White matches the video pixel positions that the researcher wishes to take into account, while black functions to identify areas that should be disregarded.

Detection of Static Information

The first algorithm introduced in this chapter represents the most straightforward processing technique which is also highly accurate and generic enough to be applicable at any layer of *segmentation* process. Beginning with the objective of differentiating moments of play from other experiences that games also offer as a hybrid medium, it is useful to determine experiences categorized by reduced levels of interactivity. The algorithm that has proved extremely useful for discerning player activity is derived from research that pursues the automatic detection of logos enclosed in TV-streams (Mikhail and Vatolin, 2011; dos Santos and Kim, 2006). Typically, this strand of research seeks to automatically evaluate the presence, or absence of specific TV-channel logos that are indicative of normal stream (standard logo), live transmission (modification of logo) or commercial breaks (disappearance of logo). The application of logo detection to gameplay footage extends its application to TV-streams significantly due to the high number of symbolic elements broadcast via the video stream to the player (Fagerholt, 2009; Ruch, 2010). Indeed, detecting logos in the context of game analysis can mean:

- The detection of the presence of the Head-Up Display (HUD) that is indicative of the sequence being fully interactive (walking, running, crouching, crawling, collecting, fighting etc.), while its absence typically indicates cut-scenes, menu activation.
- The detection of the appearance or disappearance of a pop-up panel, informing or warning the player about a change in the game world, for example, when a new goal is created for the player to fulfill, when an NPC is trying to communicate with the player (e.g. radio icon) or when the nature of the challenge faced by the player intensifies (enemy wave icons, Zagal, et al., 2008) or increases in levels of difficulty (e.g., timed tasks).
- The detection of cues that relate to *player choices* (e.g., skip a cut-scene) or affordances available to the player, for more effective or strategic play (e.g. loot a corpse or container, spend money). When a player opts to follow a cue, this too is often visually represented (illustrating compliance or adherence to the cues presented), for example, the presence of an audio-diary icon in *Bioshock 2* is triggered when a player opts to listen to an audio-tape.
- The detection of specific graphical or design features illustrative of a distinct space, for example, in *Bioshock 2* where a neon logo is indicative of the vending machines which are accessible to the player.

Any kind of static information on screen can indeed be abstracted as a logo including any static text or static portion of screen (typically non-diegetic information that is superimposed over the game world). This processing method can be executed using either edge or colour recognition, depending on the design of the game under evaluation. In the case of *Max Payne 3* (Rockstar Games, 2012) the game employs a semi-transparent logo that updates the player on the status of avatar health. The transparency of the logo combined with player movement permits colours from the bottom layer (game world) to alter the colour of that logo. In this instance edge detection, focusing on logo shape, is preferable to colour detection as a means of recognizing the logo.

To conduct logo detection, the algorithm employed requires the following inputs:

1. A *reference image*, sourced from a screen capture containing the desired logo of interest that matches the size of the video frame.
2. A *mask image* indicating the area in which the logo is present and should be found.
3. An *error tolerance value*, specifying how much error is tolerated during the logo detection process between the reference image and the screen image.
4. A *time tolerance value*, specifying the minimum period of time that the logo should remain on-screen.
5. A *frame step* that reduces the processing process by not requiring each individual frame to be processed, but instead every 2nd or 10th frame.

Because the algorithm is based on edge detection and comparison, the first step is to transform the input reference image from a full colour image to a binary edge representation. A mask image is then employed to erase areas outside the mask. Figure 2 illustrates these steps with *Max Payne 3* from the reference image containing the HUD (lower-right corner) though to its conversion into grey scale then edge representation. Finally, the edged version is masked to determine that only the HUD area is assessed (see figure 3).

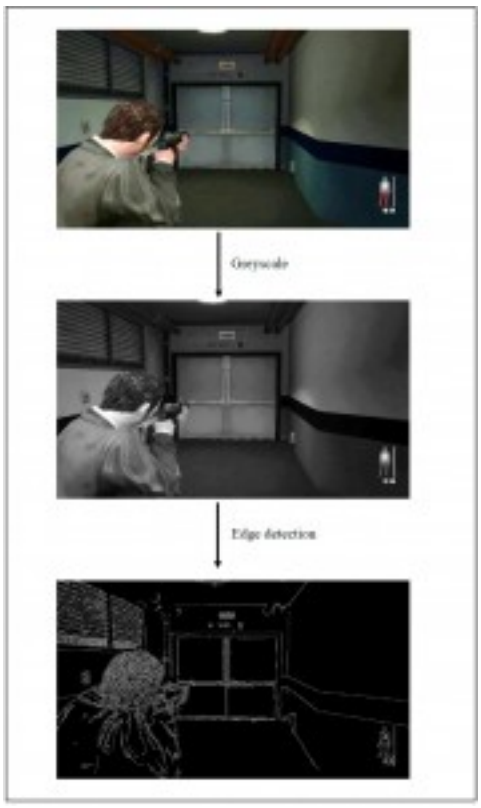


Figure 2. Image conversion to greyscale to edge detection.

Once the inputs are ready to be processed the algorithm is executed. This works in a similar fashion by automatically converting each video frame into grey scale and generating an edge image following the same process that produced the reference image. The frame and the reference image are then compared

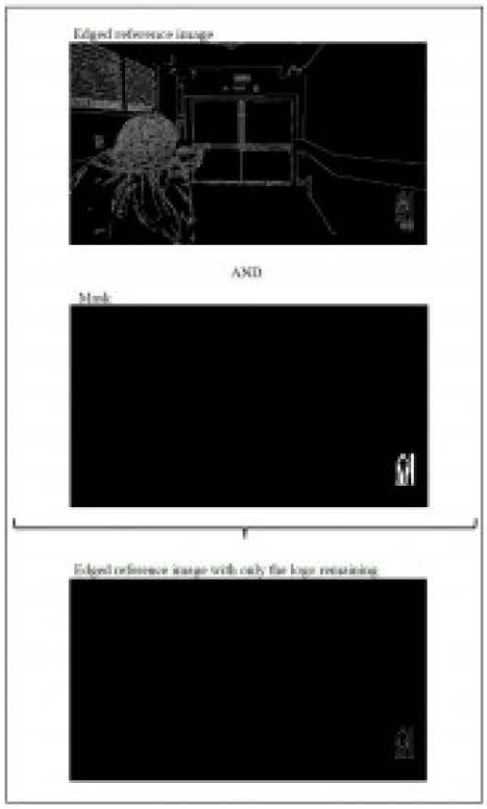


Figure 3. Achieving a masked edge image.

using a method for computing relative distance. For each white pixel (1) in the reference image, the distance between the pixel value and the matching pixel (i.e., pixel at the same position) value in the frame image is computed, resulting in a 0 if the two pixels are identical (i.e., both white), or a 1 if the two pixels fail to match. The distances are summed, and then divided by the total numbers of white pixels (1) in the reference image. The closer the result is to 0, the stronger the two images match, and the logo has been detected successfully. The relative distance is actually representative of an error value with 0 indicating no errors and 1 no match. Values between 0 and 1 represent a difference ratio. This is where the error tolerance value is employed. When the relative distance result is below the error tolerance value, the detection is accepted. If the value is above, the detection is rejected (see figure 4).

The following illustrations outline different results using a *static information detection* algorithm. Each result shown is linked to a different segmentation layer. Figure 5 illustrates the detection of the main menu of *Bioshock 2*, through the detection of the title logo. This demonstrates how the *static information detection* approach can indicate the instigation of the *game world instance* layer. This can be seen occurring in the beginning of the play session as shown in the timeline:

Figure 6 illustrates the detection of mission screens in *Battlefield 3*, using text as image and the word "mission" as a reference image. A mission-loading screen appears between each mission, signifying progression and also immediately after screen-death, reloading and sending the player back for a re-try. This result is linked to the *spatial temporal* segmentation layer. Here the timeline shows loading screens that occur as a result of level completion or as a result of screen-death (resetting the game). To distinguish between these two instances of the loading screen we would examine the results of this automatic

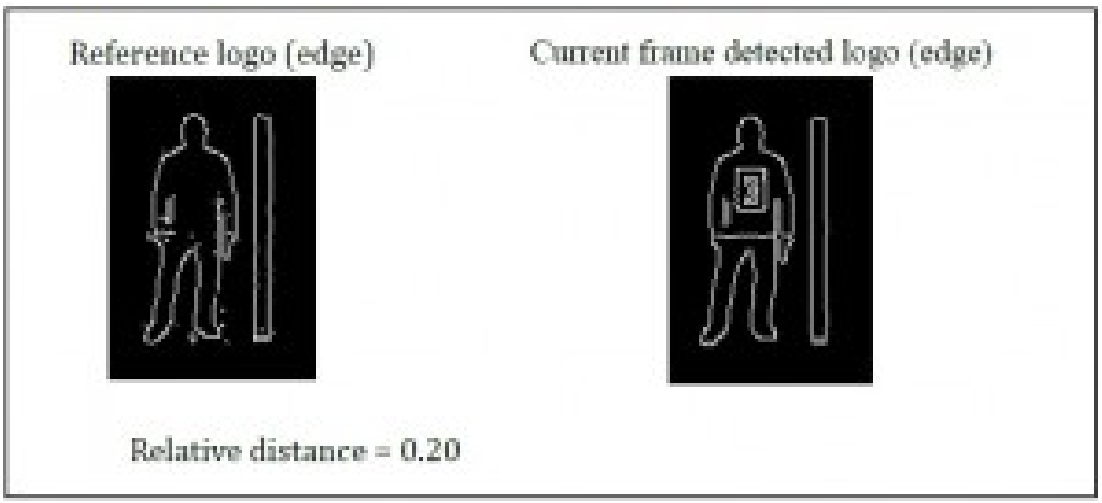


Figure 4. Relative distance result.

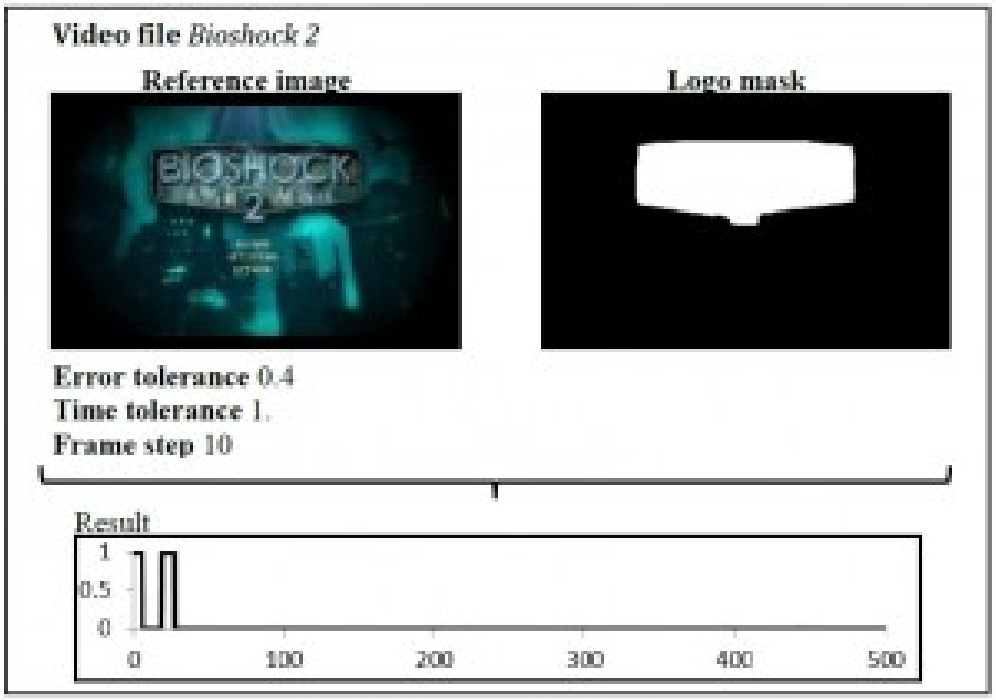


Figure 5. Main menu detection in Bioshock 2.

detection with the results of sound detection (discussed further on) to examine whether the sound indicating screen-death occurs prior to the loading screen. Its occurrence plus loading screen would indicate screen death and rule out mission completion.

Finally, figure 7 shows a moment in the game *Dead Island* (Deep Silver, 2011) when the player restores avatar health via the application of a first aid kit. Activating this process results in a first aid icon appearing in the upper-right corner of the screen. This represents feedback of player interaction. It is important to note that within the degree of freedom and the interaction layers, some static information can appear fleetingly (e.g., feedback icons, possible action cues), effecting the results of detection. For exam-

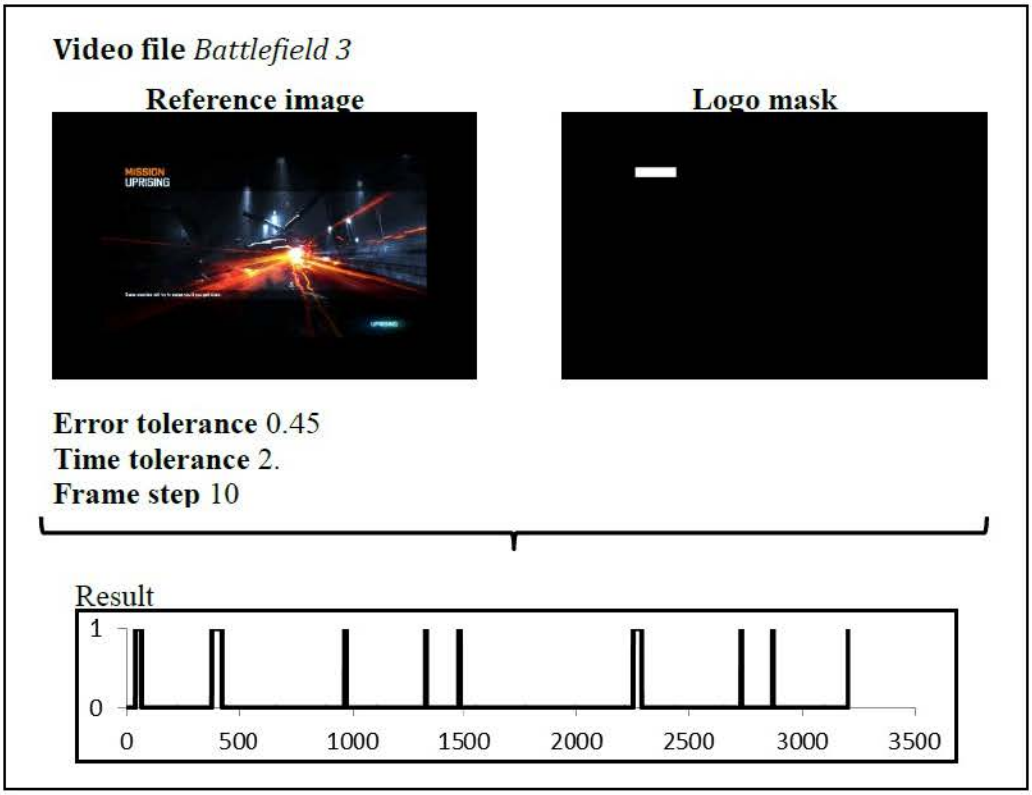


Figure 6. Mission screen detection in Battlefield 3.

ple, the detection of a logo prompting looting possibilities, discussed earlier in *Bioshock 2,* revealed how the automatic processing method missed a small number of logos when compared with a manual-coded process. These instances were missed due to a combination of the brief screen-time and because the logo colour blended with the background colour. However, such detection failures are negligible and do not compromise the overall validity of the results. The timeline demonstrates the frequency with which the player activated a health kit. Again these results will typically be contextualised with other results, such as measurement of health bar progression (discussed further on), to illustrate whether health kits are used in response to sudden health drops as the most likely outcome of intense conflict.

MOVING INFORMATION DETECTION

When an element of interest is not static or fails to appear in the same screen position, it is necessary to execute an algorithm capable of dealing with the localization and detection of objects that change position on screen. Here, a strand of research termed *image registration* (Pratt, 1978, chapter 19; Fitzpatrick, Hill and Maurer, Jr., 2000, chapter 8) can be employed. This research executes algorithms that are efficient in identifying commonalities between two images thus highlighting how they relate to each other. Thus, a smaller object can be located with a larger image when its position is not guaranteed. One method employed within this strand of research is *cross-correlation* (Pratt, 1978, p.553; Fitzpatrick, et al., 2000, p.489; Dos Santos, et al., 2006). This works by taking a *discretized* image, and scanning the reference image over each section of the larger image to seek a match. For each scanned position a correlation value is calculated (using a cross-correlation formula) with a value of 1 representing a perfect match.

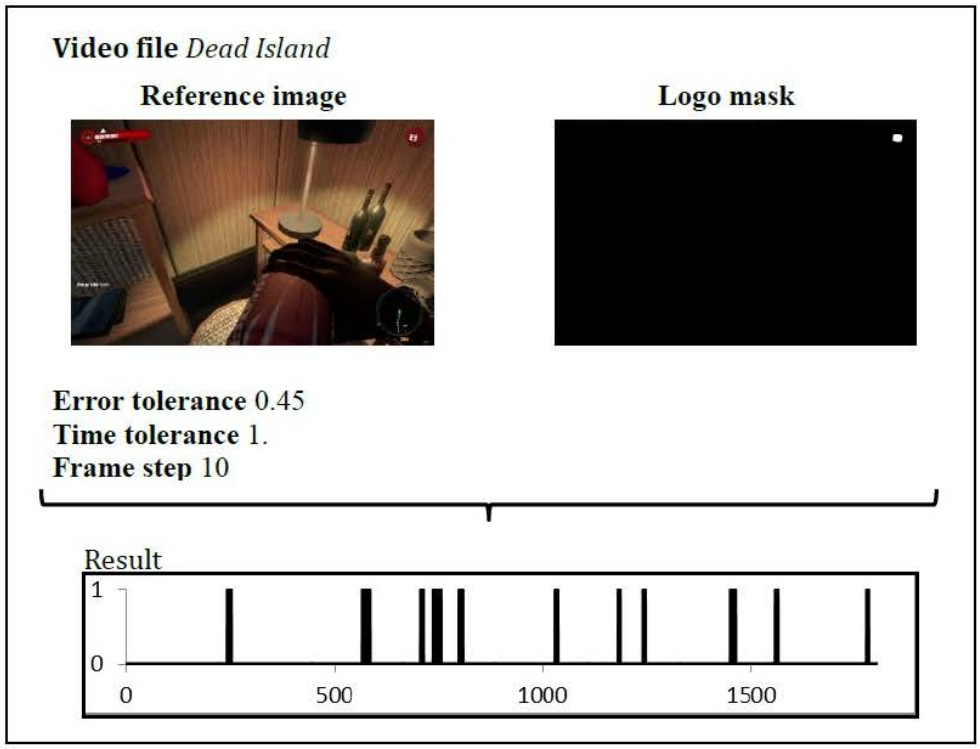


Figure 7. First aid kit used in Dead Island.

To execute detection of a non-static object, the algorithm employed requires the following inputs:

1. An *object reference*, representing the object that will be cross-correlated.
2. A *mask image* can be used to restrict the search area for an object in cases where the object only occupies a smaller area on screen. If no *mask image* is specified then the whole screen is scanned.
3. Once the scanned image locates likely matches, then an *error tolerance value*, will be employed to determine whether the object identified is a correct match.
4. A *time tolerance value*, specifying the minimum period of time that the object should appear on-screen.
5. A *frame step* that reduces the processing process by not requiring each individual frame to be processed, but instead every 2nd or 10th frame.

The algorithm functions similarly to the static object algorithm outlined above with the addition of several processing functions that account for the challenges of having to locate an object (or several objects) that may be repositioning. Once the object image has been converted into a binary edge representation it is shifted sequentially through every possible position over the video frame (beginning in the upper left corner of the frame and ending in the bottom right corner).

The correlation process (as shown in figure 8) illustrates the process of locating a "cross" logo that appears in the game *Battlefield 3* and functions to indicate the positions of enemies at a distance (enemies that are also hidden behind objects). The cross logo has been sequentially shifted across the image, in doing so, correlation scores have been generated and stored in the image matrix (the whiter the pixel,

the better the correlation score). Figure 8 shows an instance which display two perfect correlations in the upper right hand corner. These logos inform the research about the nature of the space (the player is in an environment where enemies are present), it informs the researcher as to the number of enemies (number of logos) and also provides information on the player's reaction (i.e., if the logo is in the centre of the image, then the player is typically attempting to aim at those enemies).

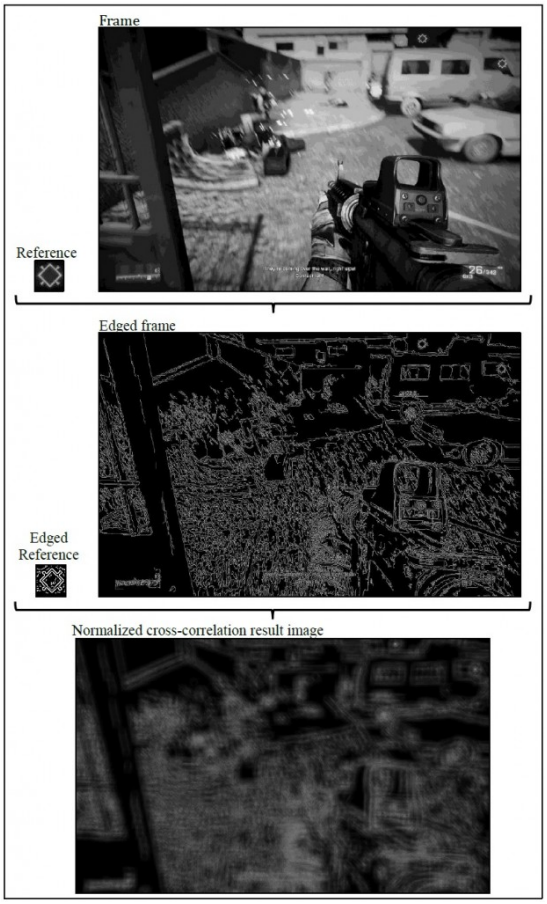


Figure 8. Cross correlation image.

As the above example demonstrates, several identical logos can be present on the same frame. In order to be able to achieve multiple detections, once an object has been detected for the first time, the correlation value of that position (where the object was detected) is reset to 0 in order to be able to repeat the process and initiate a new detection cycle (discounting objects that have already been found). The process is repeated until the logo detection algorithm is no longer able to locate the desired object. At the end of the process the results will convey the number of objects found and their position (in this case 1585, 121 and 1267, 21). Figure 9 illustrates this process.

Figure 10 illustrates the result of the example discussed above (from *Battlefield 3*) in which the logo representing enemy position can appear anywhere on-screen. The graphs included in figure 10 demonstrate the value of the logo position for the player. While the top graph demonstrates the appearance of logos on-screen, the lower graph indicates how the player has then responded to this information by adjusting their position, thus centring the logo position on-screen presumably to take aim and fire at the enemy (an action that can be further confirmed by keystroke measurement).

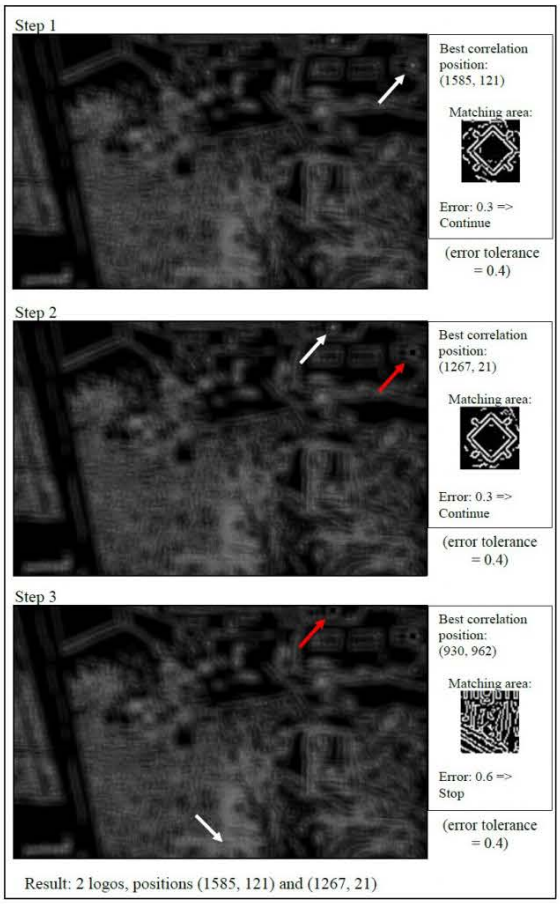


Figure 9. Multi logo detection on real example.

BAR PROGRESSION ASSESSMENT

The detection algorithms employed on graphical information presented thus far essentially produce discrete values, that is, being able to say whether a logo is detected or not and how many logos are detected and where. The information presented to the researcher utilizing these methods is therefore limited to information that confirms an action or presents the player with information that can cue behaviours (e.g., looting). However, there are other forms of information that are constantly present on-screen (whilst in play) that provide continuous updates on player's standing in the game, for example, health, power or stamina bars. *Bar progression assessment* addresses these informational sources that present continuous data. That is, a value can increase or decrease between minimum and maximum values. Such information allows the player to adapt their strategy appropriately in response to the current standing of their avatar. In the game *Dead Island*, for instance, a low health reading during a fight may trigger a desire in the player to run away, however, if stamina the levels required for running are low this may not be possible.

In order to execute an assessment of continuous values a *colour ratio algorithm* is employed. In this instance, a *colour ratio algorithm* is used to summarize and visualize the constant movement of a bar progression throughout play. Such a result can then be analysed to pinpoint moments in which key attributes related to avatar performance such as health or stamina drop. Conversely, when a player's standing increases suddenly, for instance when health is fully replenished after discovering a health kit.

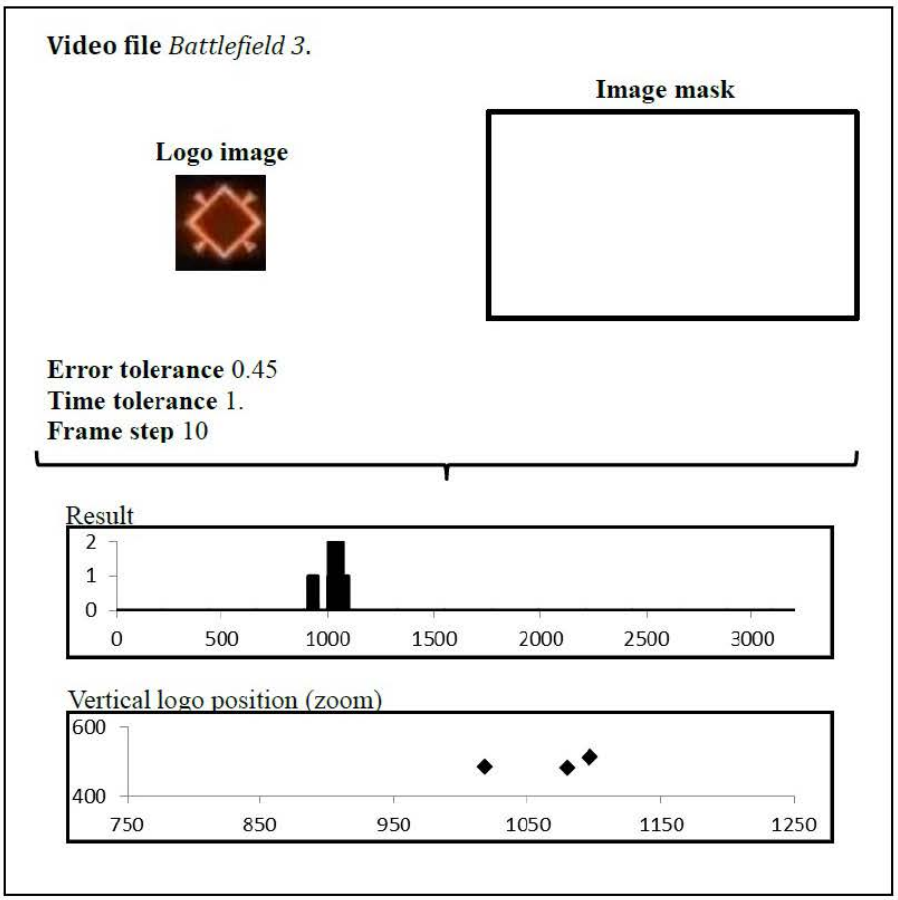


Figure 10. Enemy locations in Battlefield 3.

Bar progression activity serves as a key signifier of the intensity of action, as increases or decreases tend to occur in active moments that stand in contrast to more passive moments during gameplay where bar progression does not fluctuate so drastically. Such information is nearly impossible to extract accurately when done manually, due to its continuous nature.

A *colour ratio algorithm* functions to study each pixel colour in a given area (utilizing a mask image to pinpoint bar location and shape) comparing the background or vessel pixel colour with pixel colour that is employed to signify progression. The ratio between matching pixels and the maximum number of pixels in the area is calculated, with 1 representative of all the pixels matching the colour being searched, 0 in the case of no match. For this algorithm, the method of representation used for colour detection is HSV (rather than RBG). As discussed earlier in the chapter, HSV domain value is able to compensate for levels of transparency when a HUD overlays the game world. The required inputs to execute a colour ratio algorithm for a bar progression analysis are:

1. A *mask image* indicating the bar position, or at least a meaningful portion of it.
2. A *reference colour image* containing the colour (or colours) utilised in the bar to be processed.
3. The *tolerance values* for Hue, Saturation and Value of the colour domain that controls how much a pixel value can divert from the colour of interest and still be considered a match.
4. A frame step.

The algorithm functions by processing the reference colour image by extracting the HSV values from each pixel. Then, the number of white pixels in the mask image is calculated to represent the number of pixels indicative of the size and range of the bar (maximum number of pixels). For each video frame, the area defined by the mask is considered. Each pixel included in the bar is first converted into its HSV values, before comparing those to the reference colour. If the differences are lower than the tolerance values stipulated then the pixel is considered a match. A result of 1 is returned for a full bar, 0 for an empty bar with intermediate results between 0 and 1 indicating how full or empty the bar is at any given point. Figure 11 illustrates the application of the *colour ratio algorithm* to a gameplay frame taken from the game *Dead Island*. In this example, the health bar is processed. The figure shows how the bar is extracted from the frame using the mask image. When the ratio between the matched pixels and the total number of bar pixels was computed the frame revealed that 65% of the bar is complete. Figure 12 extends this example, to include the variance and changes in a health bar over a period of 30 minutes. As discussed, there are a number of moments in which health drops orrises dramatically amongst more stable moments. It is interesting to note that we can combine the output from the colour ratio algorithm with the findings shown for static logo detection (illustrated in figure 7), that displays the presence or absence of the health kit logo (indicating its use). We can therefore confirm whether health restoration is the result of a player acquiring a health kit.

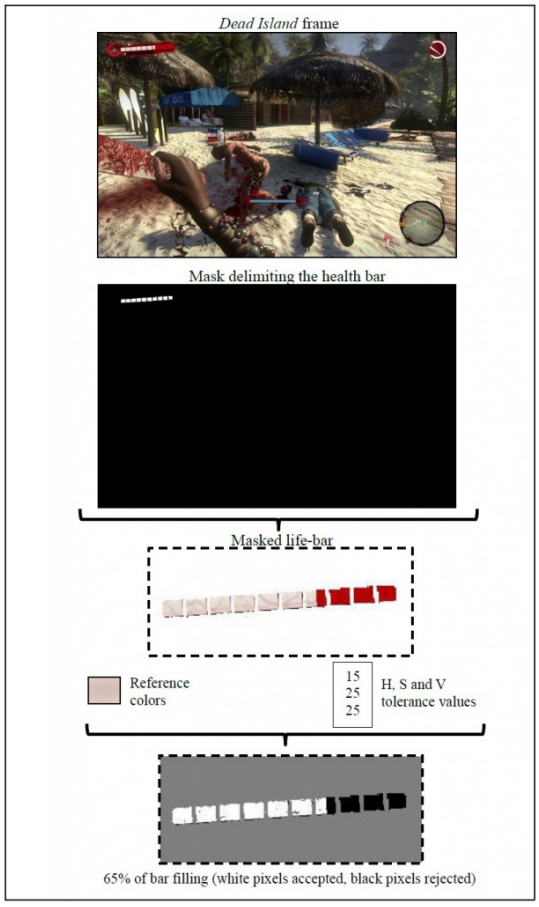


Figure 11. Life bar progression assessment in Dead Island.

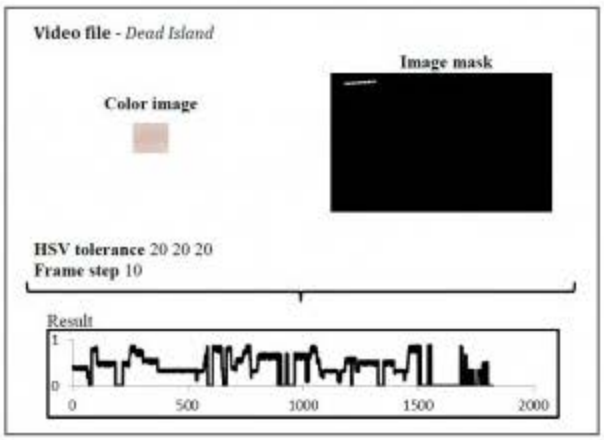


Figure 12. Health processed in Dead Island through life bar analysis.

SOUND

We would like to conclude this outline of feedback-based gameplay metrics with an example of the value of sound and how it is processed. The algorithm introduced in this final section of the chapter utilises *sound correlation* (Roads, 1996, p.509; Yarlagadda, 2010, chapter 2) techniques to identify key and significant sounds enclosed within the multi-layered audio-stream of the *gameplay performance* footage. The sonic atmosphere of most videogames carries a considerable amount of information. Some sounds extend beyond the graphic information directly presented to the player as objects in the diegetic world of the game can generate sounds even when out of the player sight. Furthermore, most of the graphical elements experienced within games are also paired with audio feedback that confirms actions. For example, a loss of health is accompanied by avatar screams in *Bioshock 2*, the use of slow motion in *Max Payne 3* possesses its own sound motif and even the most basic functions such as menu activation is also accompanied by specific sounds (e.g., *Dead Island*).

Similar to *cross-correlation* for image processing, a *sound cross-correlation* approach is employed to identify specific sounds within a multi-layered soundtrack. A similarity score is calculated to indicate when a comparison is located for a particular sound within a larger soundtrack. Again, the higher the score, the more likely the two segments are a match. The required inputs to execute a cross correlation for sound processing are:

1. A *reference sound* file representing the sound to find. This can be extracted either from footage taken from the game, selecting a clear articulation of the sound desired, or alternatively, the game can be played with soundtrack off in order to be able to go back and extract specific sounds using a sound editing application.
2. A *threshold value* is used in order to determine that only highest values are identified, for example, above 0.5.

The algorithm works by scanning the reference sound over the soundtrack, and comparing their amplitude (i.e., visualisation as a sound wave). For each scanned position a cross-correlation value is then generated. The closer the results are to 1, the better the match. Figure 13 demonstrates an example of the use of a sound cross-correlation using a 35-second extract from the game *Battlefield 3* in which the sound

that accompanies screen-death is searched for. The figure shows how the correlation curve reaches its maximum at around 15 seconds. A time stamp is generated at the point of the highest value (above a provided threshold).

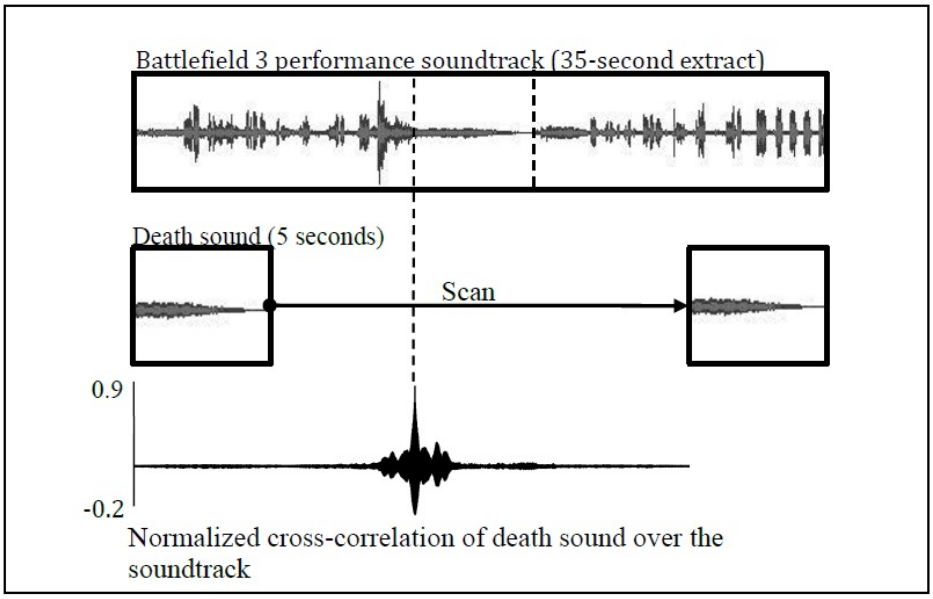


Figure 13. Sound cross-correlation.

Figure 14, illustrates the results of the process discussed above across a full session of gameplay. In this result we can see that the player died five times during the session. The validity of this result that can be confirmed by the detection of the loading screen that appears once a player dies as the game reloads (see figure 6).

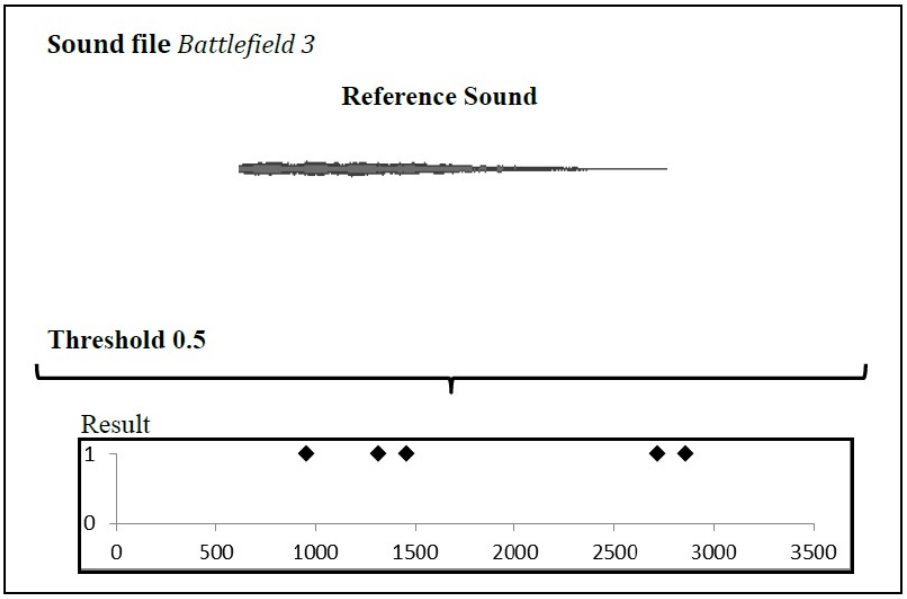


Figure 14. Death detection using sound in Battlefield 3.

Summary

This chapter has introduced a novel approach for the generation of game metrics that exploits the audio and visual feedback experienced by the player during play. Whilst it may appear a cumbersome method at this stage of its development, the application of algorithms are combined and tailored to the specific visual and sound design of the game under examination providing flexibility in its application. The method also has the advantage of allowing game researchers to accurately process footage from any PC-based game, reducing the demands and error associated with a manual analysis and permitting an examination of gameplay across a sample of players. Underpinning this method is also a framework for the segmentation and analysis of a gameplay performance to which the specific tools of the method can be contextualized and the value of specific informational sources can be assessed in relation to player experience. For an example of the application of this method readers are pointed to Schott, et al.'s (2013) use of the method in order to process play with *Max Payne 3* and assess the game's use of slow motion in terms of its impact on player experience. This research assessed whether the use of slow motion functions to glorify or aestheticize violence, or works strategically as part of the degrees of freedom afforded players.

This chapter has served to outline just a small number of examples of the algorithms that can be appropriated and adapted from computer sciences in order to automatically detect key graphical and sound streams in games. All the examples provided function with a high degree of accuracy, and contribute to the construction of an automatically generated summary of a play performance. Such summaries can then be employed in an interpretation of a player's experience of play from a behavioral perspective. The algorithms represent a straightforward method in terms of implementation, providing meaningful results that can contribute to the wider endeavors of player experience research.

References

2K Marin, 2010. *Bioshock 2* [game]. Microsoft Windows. 2K Games.

Aarseth, E., 2007. I Fought the Law: Transgressive play and the implied player. In: *Proceedings of DiGRA 2007: Situated Play*. Tokyo. September. pp.130–133.

Bay, H., Ess, A., Tuytelaars, T., and van Gool, L., 2008. SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3), pp.346–359.

Calleja, G., 2011. *In-game: From immersion to incorporation*. Cambridge: The MIT Press.

Chen, D. and Bourlard, H., 2001. Video OCR for sport video annotation and retrieval. In: *Proceedings of the 8th International Conference on Mechatronics and Machine Vision in Practice*. pp.57–62.

Choi, F.Y.Y., 2000. Advances in domain independent linear text segmentation. In: *NAACL 2000 Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. pp.26–33.

Drachen, A. and Canossa, A., 2008. Defining personas in games using metrics. In: *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, New York: ACM, pp.73–80. DOI=10.1145/1496984.1496997.

Drachen, A., Seif El-Nasr, M. and Canossa, A., 2013. Game analytics—The basics. In: M.Seif El-Nasr et al., eds., *Game analytics*. London: Springer, pp.13–40.

Electronic Arts, 2011. *Battlefield 3* [game]. Microsoft Windows. PUBLISHER.

Erik de Castro Lopo, 2011. *libsndfile*. V1.0.25 [computer program]. Availabe at: <https://github.com/erikd/libsndfile/>.

Fagerholt, E. and Lorentzon, M., 2009. *Beyond the HUD: User interfaces for increased player immersion in FPS Games*. MS. Chalmers University of Technology. Available at <http://publications.lib.chalmers.se/records/fulltext/111921.pdf>.

Fitzpatrick, J.M., Hill, D.L.G. and Maurer, Jr., C.R., 2000. *Handbook of medical imaging*. Bellingham: SPIE PRESS.

Gonzales, R.C. and Woods, R.E., 2007. *Digital image processing*. 3rd ed. New Jersey: Prentice Hall.

Hilbert, D. M. and Redmiles, D.F., 2000. Extracting usability information from user interface events. *ACM Computing Surveys (CSUR)*, *32*(4), pp.384–421. DOI=10.1145/371578.371593.

Huang, C.-L. and Liao, B.-Y., 2001. A robust scene-change detection method for video segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(12), pp.1281–1288.

Intel Corporation, Willow Garage and Itseez, 2014. *OpenCV*. V2.4.9 [computer program]. Available at: <http://opencv.org/>.

Klevjer, R., 2002. In Defense of cutscenes. In: F. Mäyrä, ed., *Proceedings of computer games and digital cultures conference*. Tampere: Tampere University Press, pp.191–202.

Laurel, B., 1993. *Computers as theatre*. Reading: Addison-Wesley.

Mikhail, E. and Vatolin, D., 2011. Automatic logo removal for semitransparent and animated logos. In: *Proceedings of GraphiCon 2011*. Moscow. September 26–30. GraphicCon.

Nacke, L. E., Drachen, A., Kuikkaniemi, K., Niesenhaus, J., Korhonen, H.J., Hoogen, V.D.W., Poels, K., IJsselsteijn, W. and Kort, Y. 2009. *Playability and player experience research*. London, September. Available at: < http://www.digra.org/wp-content/uploads/digital-library/09287.44170.pdf >.

Orio, N. 2006. Music retrieval: A tutorial and review. *Foundations and Trends in Information Retrieval*, 1(1), pp.1–90.

Pedersen, C., Togelius, J. and Yannakakis, G.N., 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1), pp.54–67. DOI=10.1109/TCIAIG.2010.2043950.

Pratt, W.K., 1978. *Digital image processing*. New York: Wiley-Interscience Publication.

Richard, G., Ramona, M. and Essid, S., 2007. Combined supervised and unsupervised approaches

for automatic segmentation of radiophonic audio streams. In: *Proceedings – ICASSP.* PLACE. DATE, pp.461–464.

Roads, C., 1996. *The computer music tutorial*. Cambridge: MIT Press.

Ruch, A.W., 2010. Videogame interface: Artefacts and tropes. In: *Videogame Cultures and the Future of Interactive Entertainment Global Conference*. Oxford. DATE.

Santos, A.R. and Kim, H.Y., 2006. Real-time opaque and semi-transparent TV logos detection. In: *Proceedings of the 5th International Information and Telecommunication Technologies Symposium (I2TS)*.

Rockstar Studios, 2012. *Max Payne 3* [game]. Microsoft Windows. Rockstar Games.

Saraceno, C. and Leonardi, R., 1997. Audio as a support to scene change detection and characterization of video sequences. In: *Proceedings of 1997 IEEE international conference on acoustics, speech, and signal processing.* IEEE, pp.2597–2600.

Sasse, D., 2008. *A framework for psychophysiological data acquisition in digital games.* MS. Otto-von-Guericke-University Magdeburg. Available at: <http://www.gamecareerguide.com/thesis/080520_sasse.pdf>.

Schott, G., Vught, J.V. and Marczak, R., 2012. The "dominant effect" of games: content vs. medium. *CoLab: Journal of Creative Technologies*, Available at: <https://colab.aut.ac.nz/journal/the-dominant-effect-of-games-content-vs-medium/>.

Schott, G., Marczak, R., Mäyrä, F. and Vught, J., 2013. DeFragging regulation: From putative effects to 'researched' accounts of player experience. In: *Proceedings of DiGRA 2013: Defragging game studies.* Atlanta, Georgia. August. Available at: <http://www.digra.org/wp-content/uploads/digital-library/paper_29.pdf>.

Techland, 2011. *Dead Island* [game]. Microsoft Windows. Deep Silver.

Vught, J.V., Schott, G. and Marczak, R. 2012. Age-restriction: Re-examining the interactive experience of "harmful" game content. In: Nordic DiGRA. Tampere: June. Available at: <http://www.digra.org/wp-content/uploads/digital-library/12168.32309.pdf>.

Waern, A., 2012. Framing games. In: *Proceedings of Nordic DiGRA 2012*. Tampere. June. Available at: <http://www.digra.org/wp-content/uploads/digital-library/12168.20295.pdf>.

Yarlagadda, R.K.R., 2010. *Analog and digital signals and systems*. London: Springer.

Ye, Q., Huang, Q., Jiang, S., Liu, Y. and Gao, W., 2006. Jersey number detection in sports video for athlete identification. In *Proc. SPIE 5960, Visual Communications and Image Processing*. International Society for Optics and Photonics, pp.59604P-59604P.

Yeh, C.-H., Kuo, C.-H. and Liou, R.-W., 2009. Movie story intensity representation through audio-visual tempo analysis. *Multimedia Tools and Applications*, 44(2), pp.205–228. DOI=10.1007/s11042-009-0278-8

Zagal, J. P., Fernandez-Vara, C. and Mateas, M., 2008. Rounds, levels, and waves: The early evolution of gameplay segmentation. *Games and Culture*, 3(2), pp.175–198.

Zagal, J.P., Mateas, M., Fernández-Vara, C., Hochhalter, B., and Lichti, N., 2005. *Towards an ontological language for game analysis*. In: Proceeding of DiGRA 2005 Conference. Vacouver. June. Available at: <http://www.digra.org/dl/db/06276.09313.pdf>.