6 Video-Based Dramatization System

This chapter describes the technical details about the implementation of the proposed video-based dramatization system.

6.1. Methods and Libraries

The video-based dramatization system was implemented in C++ with some algorithms running on GPU (Graphics Processing Unit). The system is based on several video processing and artificial intelligence algorithms that were implemented using some open source libraries. The next sub-sections describe the methods and libraries used in the implementation of the proposed video-based dramatization system.

6.1.1. Image and Video Processing

The main task of the video-based dramatization system is to compose and generate video sequences representing story events in real-time. The process of compositing a scene using video segments from different sources requires fast and optimized video and image processing algorithms capable of assembling multiple visual elements into a single piece of motion picture in real-time. In order to implement such algorithms, we adopted the OpenCV (Open Source Computer Vision Library),[5] which is an open source computer vision and machine learning software library written in C/C++ and designed for computational efficiency with a strong focus on real-time applications (Bradski and Kaehler 2008).

The OpenCV has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms (OpenCV, 2014). It contains a mix of low-level

[5] OpenCV - http://www.opencv.org

image-processing functions and high-level algorithms such as face detection, pedestrian detection, feature matching, and tracking. One of the OpenCV's goals is to provide simple-to-use computer vision interfaces and allow researchers to easily build sophisticated applications. The OpenCV can be used for several computer vision tasks, such as detecting and recognizing faces and other objects, processing video frames, tracking camera movements, establishing markers for augmented reality, among several other simple and complex computer vision tasks.

The OpenCV library also includes a GPU module written in CUDA (Compute Unified Device Architecture) that provides GPU acceleration for the computer vision and image processing algorithms, allowing programmers to benefit from GPU acceleration without requiring background in GPU programming (Pulli et al. 2012). The GPU module covers a significant part of the library's functionality and is still in active development. The module is designed as host API extension, which provides the user an explicit control on how data are moved between CPU and GPU memory.

The core of the proposed video-based dramatization system was built using the OpenCV functions to handle video files, process and compose video frames, and present the resulting motion picture. In order to improve the performance, the system uses the GPU module of the library whenever possible.

6.1.2. Artificial Neural Networks

During the compositing process, the dramatization system must follow the principles defined by cinematography theory and make intelligent decisions to create attractive and engaging visual presentations for the stories. In order to represent the cinematography knowledge and to make the system capable of performing intelligent decisions, we use several artificial neural networks trained to automatically select the best camera shots, visual effects and sound tracks for the scenes in real-time.

The artificial neural networks were implemented in the proposed system using the library FANN (Fast Artificial Neural Network Library),[6] which is an

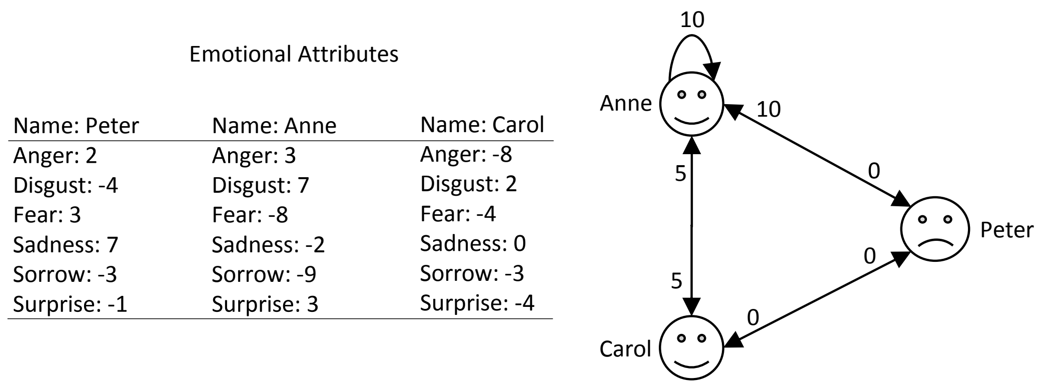
[6] FANN - http://leenissen.dk/fann/

open source library that implements multilayer artificial neural networks in C with support for both fully connected and sparsely connected networks (Nissen 2003).

A complete description of the theory about artificial neural networks is presented by Hassoun (1995), Mitchell (1997) and Russell and Norvig (2010).

6.1.3. Emotions and Relations Network

Expressing and evoking emotions is a key factor to engage the audience in a narrative. The cinematography theory describes several ways to emphasize the emotions of characters by using specific camera shots, camera movements, light and music. In this way, the emotional states of characters participating in the action are essential to apply most of the cinematography concepts in the dramatization of stories. However, the current version of Logtell does not provide enough emotional information during the generation of stories. In order to overcome this limitation and simulate the emotions of characters during dramatization, we included in our system a dynamic multi-character network (Figure 6.1), where nodes represent the emotional state of characters and bidirectional arcs define affection relationships in the social environment of the story.



Emotional Attributes

Name: Peter	Name: Anne	Name: Carol
Anger: 2	Anger: 3	Anger: -8
Disgust: -4	Disgust: 7	Disgust: 2
Fear: 3	Fear: -8	Fear: -4
Sadness: 7	Sadness: -2	Sadness: 0
Sorrow: -3	Sorrow: -9	Sorrow: -3
Surprise: -1	Surprise: 3	Surprise: -4

Figure 6.1: Emotions and Relations Network.

The emotional model adopted in the emotions and relations network is based on the six basic emotions proposed by Ekman and Friesen (1971), but we consider them lying on six emotion axes with negative and positive sides that represent opposite emotions:

- [calmness, anger];
- [liking, disgust];
- [confidence, fear];
- [joy, sadness];
- [cheeriness, sorrow];
- [anticipation, surprise].

The values in each axis are numbers within the interval [-10, +10]. The emotional state of a character i is given by intensity levels of its basic emotions $e_i^k(t) \in [-10, +10]$ and its affective relations $affection_{i,j}(t) \in [-10, +10]$ with the other characters. The sign (- or +) does not mean destructive or constructive emotions, but a connotation of drama impact and opposite states. The relations are directed and are not necessarily symmetric:

$$\exists\ i, j\ relation_{i,j}(t) \neq relation_{j,i}(t)$$

In this model, emotions can be combined to form a new emotion, for instance: love = joy + liking + confidence. Also, we can refer to extreme values on an axis as being special emotions, e.g.: grief = very high levels of sadness and ecstasy = very high levels of joy (that is, very low levels of sadness).

The network has fixed topology that is defined in an XML file by the author of the story. This definition includes the initial emotional state of all characters and their initial social relations. During dramatization, the network is updated when some event occurs. The update function can be described as:

$$\varphi(action, executor, victim)$$

where *action* indicates the action that occurred, *executor* represents the characters that performed the action, and *victim* indicates that characters that suffered the action.

Each action affects the emotions and relations in different ways. For example, considering the occurrence of a *kill* event, where a character CH_1 kills another character CH_2. In this case, the update function classifies the event into one of three types of kill according to the current state of the characters and then updates the emotional states according to the selected type of action:

- **Intentional Kill:** occurs when the killer hates the victim $\left(affection_{CH_1,CH_2}(t) < -3\right)$. In this case, the emotional state of the killer will be updated by increasing his level of joy by +3 and confidence by +2.
- **Indifferent Kill:** occurs when the killer has a relation of indifference with the victim $\left(affection_{CH_1,CH_2}(t) > -3 \text{ and } affection_{CH_1,CH_2}(t) < +3\right)$. If the killer is a villain, his emotional state will be updated by increasing his level of joy by +1 and confidence by +1; otherwise, if the killer is a good person, his emotional state will be updated by decreasing his level of joy by -3, confidence by -2 and cheeriness by - 2.
- **Non-intentional Kill:** occurs when the killer likes $\left(affection_{CH_1,CH_2}(t) > +3\right)$ or loves $\left(affection_{CH_1,CH_2}(t) > +6\right)$ the victim. In this case, the emotional state of the killer will be updated by decreasing his level of joy by -4, confidence by -3, calmness by -3, liking by -2 and cheeriness by -2.

The constant values used to update the emotions and relations for each type action must be defined by the author of the story according to his/her authorial intent.

The emotions and relations network is capable of providing the basic emotional information required by the proposed video-based dramatization system, and its ability of simulating emotions and relations was already tested and validated in some of our previous works (Lima et al. 2010; Lima et al. 2011A).

6.2. Cinematography Agents

The proposed video-based dramatization system is composed of a set of cinematography-based autonomous agents that perform the same roles played by the corresponding filmmaking professionals. The process to create video-based interactive narratives is performed by the agents and is divided into two phases:

(1) scene definition, where the logical description of the scene is defined; and (2) scene compositing, where the video frames representing the scene are generated by the system. Figure 6.2 shows an overview of the video compositing process.

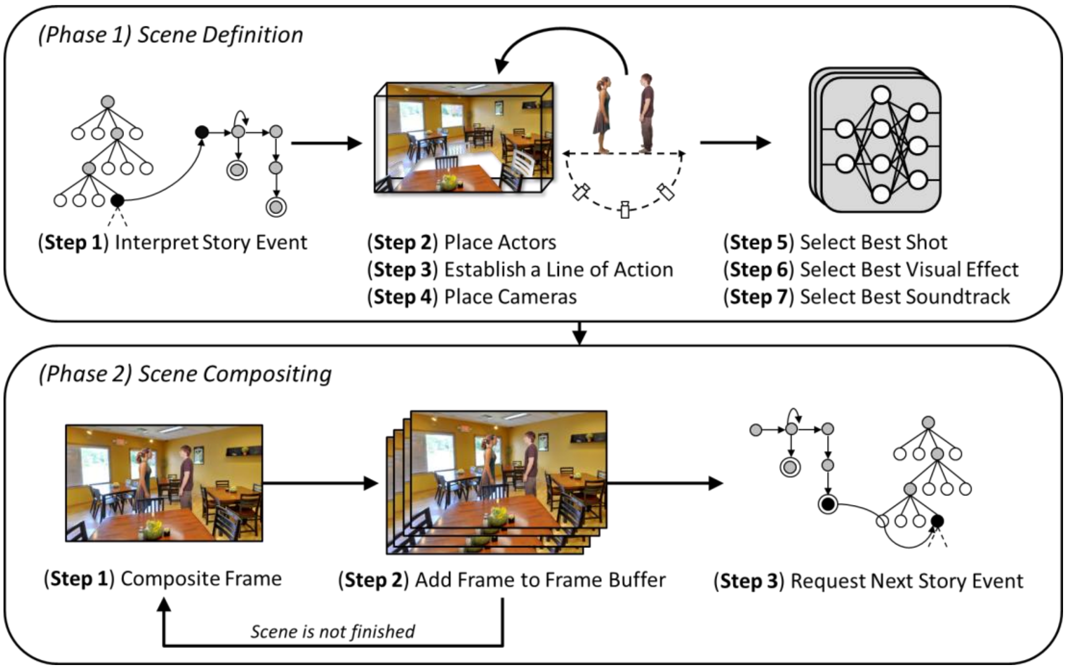

(Phase 1) Scene Definition
(Step 1) Interpret Story Event
(Step 2) Place Actors
(Step 3) Establish a Line of Action
(Step 4) Place Cameras
(Step 5) Select Best Shot
(Step 6) Select Best Visual Effect
(Step 7) Select Best Soundtrack
(Phase 2) Scene Compositing
(Step 1) Composite Frame
(Step 2) Add Frame to Frame Buffer
(Step 3) Request Next Story Event
Scene is not finished

Figure 6.2: An overview of the video compositing process.

The tasks performed by the agents are divided in three processes: film directing, film compositing and film scoring. The next sub-sections describe these processes and the implementation of each cinematography-based agent.

6.2.1. Film Directing

In filmmaking, the director is responsible for creatively translating the written script into a visual form. He/she visualizes the script by giving to abstract concepts a concrete form, which helps him/her to determine the general structure of each scene of the narrative, including the position of actors and cameras. The director is responsible for the dramatic structure and the directional flow of the film (Mascelli 1965).

In the proposed video-based dramatization system, the agents Scriptwriter and Director share the responsibility of directing the dramatization of the interactive narratives. The directing process is divided into different steps, as illustrated in the flowchart of Figure 6.3. The process starts when the Scriptwriter agent receives a new nondeterministic automaton containing the logical descriptions of a story event for dramatization. The first step of the directing process consists of parsing and interpreting the received automaton. Once the automaton has been parsed, the Director agent starts the dramatization of the first basic action described on the automaton. If execution of the basic action led to a final state in the automaton, a new automaton is requested to the story server; otherwise, if it led to a branching state, the system check's the users' suggestions to decide which path to follow in the automaton. Once the path has been chosen or when there was no branching point, the Director agent starts the dramatization of the next basic action.

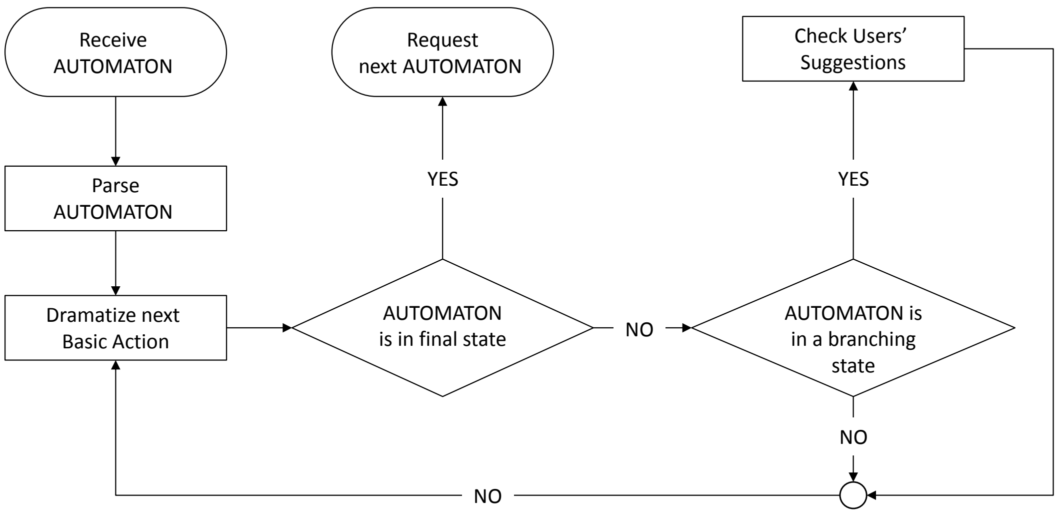


Figure 6.3: Flowchart of the directing process.

The next sub-sections describe in more details the implementation of all the steps of the directing process.

6.2.1.1.
Scriptwriter

In filmmaking, the scriptwriter (or screenwriter) is responsible for creating a compelling and coherent story for the script of the film. Similarly, the Scriptwriter agent of the proposed video-based dramatization module is responsible for interpreting and managing the interactive story plots generated by the planning algorithms of the story generator module. The agent receives and interprets the nondeterministic automata of the story events, requesting the next automaton after the successful execution of all actions present in the current automaton.

The automata interpreted by the agent are represented by adjacency list structures. They follow the same formalism introduced in Logtell by Doria et al. (2008), where states are described by invariants and transitions between states correspond to the basic actions that can be performed by the actors. States that have more than one adjacency are used as local decision points, where users can decide which action the actors should take. An example of automaton is described and illustrated in Section 5.1.1.4.

The communication between the Scriptwriter agent and the story generator module is done through a TCP/IP connection. The story generator module acts as a server and is always waiting for incoming connections on port 2563. The story dramatization module acts as a client that must connect to the IP address of the story generator module to request the automata of the story events. The communication protocol is based on the following rules:

- To request the first or the next automaton of an ongoing story, the client sends a message to the server in the format: `next#STATE_ID`, where `STATE_ID` indicates the ID of the final state reached during the execution of the current automaton. In the case of a new story, the state ID is -1.
- The server responds the client requests by sending a network message containing the next automaton of the ongoing story. The automaton is encoded in the following format:

```
plan#[AUTOMATON_ID, [STATE1_ID,
                       [STATE1_FACT1, ..., STATE1_FACTn],
                       [[BASIC_ACTION1, NEXT_STATE_ID], ...,
                       [BASIC_ACTIONn, NEXT_STATE_ID]]
                     ], ... ,
                     [STATEn_ID,
                       [STATEn_FACT1, ..., STATEn_FACTn],
                       [[BASIC_ACTION1, NEXT_STATE_ID], ...,
                       [BASIC_ACTIONn, NEXT_STATE_ID]]
                     ]]
```

which contains a list of states identified by an ID (`STATE`$_1$`_ID, ..., STATE`$_n$`_ID`). Each state is composed of a set of facts in the form of first order logic sentences describing the state (`STATE`$_n$`_FACT`$_1$`, ..., STATE`$_n$`_FACT`$_n$) and a set of basic actions (`[[BASIC_ACTION`$_1$`, NEXT_STATE_ID], ..., [BASIC_ACTION`$_n$`, NEXT_STATE_ID]]`) that also indicate the next state of the automaton (`NEXT_STATE_ID`) after the execution of the action.

The basic actions that compose the automata and can be performed by the actors are described in the form of ground first order logic sentences. For example, the action where a character *X* looks at another character or object *Y* in a location *W* is expressed by the sentence *LookAt*([*X*], [*Y*], [*W*]). The predicate indicates the action and the variable symbols indicate the characters, objects and locations related to the action. The symbols are expressed as lists of variables, which allows more than one character or object to be involved in the same action. For example, it is possible to express an event where two characters (X_1 and X_2) look at the same time to another character *Y* in a location *W* (*LookAt*([X_1, X_2], [*Y*], [*W*])). This is also very useful to represent dialog events, where a character can be speaking to several other characters. For example, the sentence *Tell*([*X*], [*Z*], [Y_1, Y_2, Y_3], [*W*]) indicates that the character *X* is speaking the utterance *Z* to the characters Y_1, Y_2 and Y_3 in the location *W*.

Once the Scriptwriter agent has received the network message containing the description of the automaton, the agent parses the message and creates an adjacency list structure representing the nondeterministic automaton. The structure is then sent to be executed by the Director agent.

6.2.1.2. Director

In filmmaking, the director is responsible for translating the script into a visual presentation. He/she controls the overall aspects of the film, including the content and flow of the narrative events, the performance of the actors, and the organization and selection of the locations in which the film will be shot. Similarly, the Director agent controls the overall flow of the dramatization by interpreting and controlling the execution of the nondeterministic automaton of the story events, deciding which path to follow in the automata according to the users' choices. The agent is also responsible for assigning roles to the actors involved in the action and selecting the locations where the scenes will be shot.

The first task of the Director agent is to load the resources (actors and locations) used for dramatization. During the initialization of the system, the Director agent reads all information about actors, locations and static scenes from the XML files `Actors.xml`, `Locations.xml`, and `Static.xml`, which are included in the narrative resource package (whose the format were presented in Sections 5.3.3, 5.3.4, 5.3.5 and 5.3.6). The actors are instantiated and their respective behaviors and associated videos are properly loaded and stored in a hash table using the actor name as key. Similarly, the locations are also instantiated and their data (video/image layers and waypoints) are loaded and stored in another hash table using the name of the location as key. Static scenes are stored in another hash table using the static predicate as key. Storing the actors, locations and static scenes in hash tables provides an optimized and efficient way of accessing the resources during the compositing process, allowing a direct mapping between the variable symbols present in the first order logic sentences of the basic actions and the key/value pairs of the hash table.

The second task of the Director agent starts when a new automaton is received for dramatization. The execution of the nondeterministic automaton begins in the initial state and ends in a final state. In each transition between states, the corresponding basic action is dramatized. When a local decision point is reached (a state with more than one adjacency), the agent consults the users' choices to decide which action the actors should take (more details about local user interactions will be presented in Chapter 7).

In order to initiate the dramatization of a basic action, the Director agent creates a structure of a scene. This structure comprises a list of scene elements that compose the scene representing the basic action. There are three types of scene elements:

1. **Location**: represents the place where the action is happening and includes the video or image layers of 8 angles of the location together with their respective waypoints and encoded information;

2. **Main characters**: represent the characters participating in the action and include the videos and alpha masks of the actors performing their current actions;
3. **Supporting characters**: represent the characters that are not directly participating in the action, but are in the same place where the action is happening.

The process of creating the structure of a scene includes a simple verification to check whether the basic action is a static scene or it needs to be dynamically composed by the system. This is accomplished by consulting the hash table of static scenes using the predicate of the basic action as key. If the basic action is a static scene, the reference to the video of the scene is added to structure as a location and, during the composition process, its frames will be directly added to the frame buffer without additional processing costs; otherwise, if the scene needs to be dynamically composed, the scene elements are included in the structure according to the description of the basic action. For example, considering the action *LookAt*([X_1, X_2], [Y], [W]), the location W and the characters X_1, X_2 and Y will be included in the list of scene elements, and the current behavior of X_1 and X_2 will be set to the action *LookAt*. In addition, other characters that are not directly participating in the action, but are located in W are also included in the scene structure as supporting characters.

Once the scene structure has been created, it is sent to the Scene Composer agent, who will be in charge of compositing a video sequence to represent the basic action.

6.2.1.3. Actors

Actors are entities that represent the characters of the stories. They are composed of a set of behaviors representing the actions they can perform during dramatization. Although their actions are selected by the Director agent based on the scene that has to be dramatized, they have the freedom to represent the specified action autonomously.

The behaviors are the core of the Actors. Each behavior implements a specific action and is composed of a set of videos representing the actor performing the action from different angles. The behaviors can be simple, like an *Idle*, *Talk* or *ReadBook*, where the video of the action is simply played on the same position of the scene; or more complex, like a *Walk*, where the actor has to physically move across the scene while playing the pre-recorded video of the action.

The proposed system includes a comprehensive set of behaviors that can be used to create video-based interactive narratives. However, different story contexts may require the implementation of new behaviors for the actors. In order to simplify such implementations, the behaviors are coded in the system using a hierarchy of classes, where the base class provides the basic functionalities needed for the implementation of new behaviors without accessing the other components of the system.

Each behavior is implemented in a separate class that is inherited from the base class `BehaviorBase`, which contains the basic information and functions of a behavior and provides access to the video data representing the action. Each behavior implements a virtual method called `ProcessBehavior`, which is automatically executed in loop and can be used to implement the logic of the behavior. Figure 6.4 shows a simple template of a behavior class inherited from the class `BehaviorBase`.

```
class SimpleBehavior : public BehaviorBase
{
     public:
       SimpleBehavior(void);
       ~SimpleBehavior(void);

       void ProcessBehavior()
       {
             //Behavior logic loop
       }
};
```

Figure 6.4: Template of a behavior class inherited from the class BehaviorBase.

Simple actions, such as an *Idle* behavior (where the actor stands in the same position), do not require the implementation of a logic loop; however, more dynamic behaviors, like a *Walk* action (where the actor has to move between waypoints), require the implementation of a logic loop to create the movement of the actor between waypoints.

The process of moving the actors between waypoints would be simple considering a scene from only one angle; however, it become more complex when we consider a dynamic angle that can change while the actors are performing the movement. If the actor is halfway between two waypoints when observed from one angle, he must be halfway when viewed from any other angle. However, the positions and even the distances between the waypoints are not the same when they observed from different angles in a 2D plane (Figure 6.5). In order to solve this problem, the position of the actors when moving between waypoints is calculated using a linear interpolation between the two waypoints. The position x and y an actor A when moving between two waypoints (WP_1 and WP_2) is given by:

$$A_x(WP_1^x, WP_2^x, p) = WP_1^x * (1 - p) + WP_2^x * p$$
$$A_y\left(WP_1^y, WP_2^y, p\right) = WP_1^y * (1 - p) + WP_2^y * p$$

where $p \in [0, 1]$ represents the relative position of the actor between the waypoints (when $p = 0$, the actor is at WP_1; when $p = 1$, the actor is at WP_2). The value of p is gradually increased according to a constant π, which defines the speed of the actor and can be adjusted to match the speed of the video of the actor walking with the speed of the physical movement.

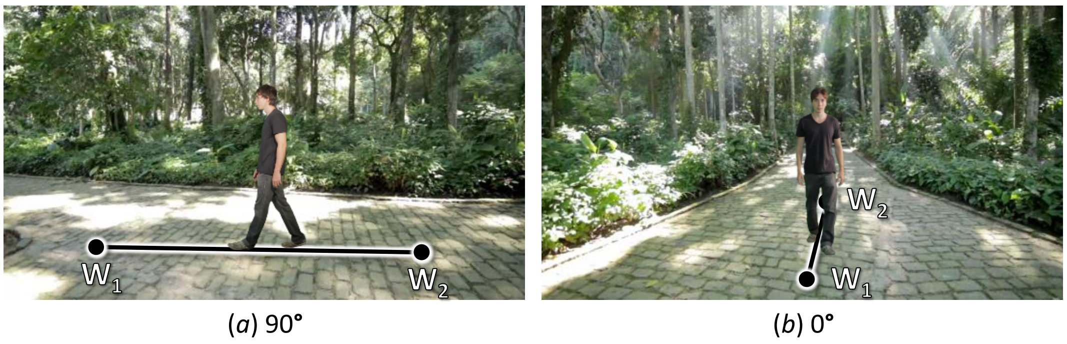


Figure 6.5: Scene of a character walking from W_2 to W_1. Image (a) shows the scene viewed from 90° and image (b) shows the same instant (p = 0.5) viewed from a virtual camera placed at 0°.

The structure of waypoints that establishes the locations where characters can be placed in the scene is represented as a graph. Sometimes the actors may have to walk through several waypoints to reach their destination. Consequently, the logical loop of the walk behavior also implements an A* search algorithm (Russell and Norvig 2010) in order to find the best paths the actors have to follow to reach their destination.

6.2.2. Film Compositing

The compositing process consists in assembling the visual elements that compose the scenes into a single piece of video. The goal is to create the illusion that all elements always existed in the same location. In real filmmaking, this process usually is manually done by several special effects professionals that work for days in short video segments to create realistic scenes. In video-based interactive storytelling, however, this process must be done in real-time and without human intervention.

As a traditional film, a video-based interactive narrative must have a cinematic look and be composed of a variety of different shots, camera movements and lighting effects. In order to create such cinematic interactive narratives, actors and settings are both shot from 8 different angles with intervals of 45° during the production process. In this way, the system has the freedom to compose scenes from different angles, simulate camera movements and create more dynamic video sequences that cover all the important aspects of the cinematography theory. However, handling such tasks without human intervention requires the development of fast and intelligent algorithms to apply the cinematography techniques to create attractive and engaging visual representations for the story events in real-time.

In the proposed video-based dramatization system, the agents Editor, Cameraman, Scene Composer and Director of Photography share the responsibility of compositing the scenes according to the information provided by the Director agent. The compositing process is divided into several steps, as illustrated in the flowchart of Figure 6.6.

The compositing process starts when the Scene Composer agent receives a new scene structure for dramatization. The first step of the compositing process consists in defining the basic setup for the scene by placing the actors that are participating in the action on the available waypoints of the location. Then, the Cameraman agent establishes the line of action and places the possible cameras to shot the scene according to the spatial information of the scene setup. Based on the available cameras, the Editor agent selects the best angle and type of shot to film the scene. Then, the agent verifies the occurrence of jump cuts. If a jump cut is detected, a new camera angle is selected; otherwise, the agent continues the process and selects the most adequate shot transition (cut, dissolve, wipe or fade). Before starting compositing the frames, the Director of Photography agent enters in the process and selects the best lighting and color effects to emphasize the emotional content of the scene. Finally, after defining all the visual aspects of the scene, starts the actual process of generating video frames representing the scene, which is the most time-consuming task. Once a frame has been generated, it is added to the frame buffer to be shown to viewers. After all frames have been successfully composed, the Scene Composer agent requests the next scene structure to be dramatized.

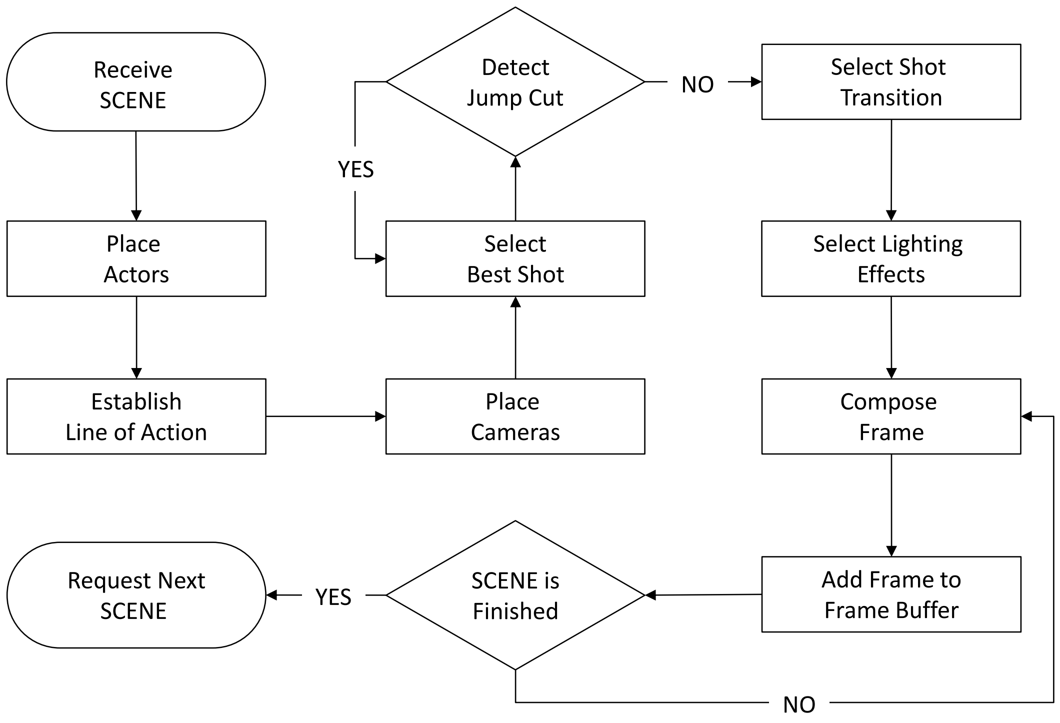


Figure 6.6: Flowchart of the compositing process.

The next sub-sections present more details about the implementation of all the steps of the compositing process.

6.2.2.1. Placing Actors and Establishing the Line of Action

In the first step of the compositing process, the basic configuration of the scene is logically defined by the Scene Composer agent. First, actors who are participating in the scene are placed on the available waypoints of the location where the scene is happening.

As previously mentioned in Section 5.3.5, there are three types of waypoints:

- **Entrance/Exit Waypoints**: used as starting or ending points to place characters when they are performing *GoIn* or *GoOut* actions, that is, when they are entering or leaving the scene location;
- **Acting Waypoints**: used to place characters when they are performing other actions in the scene;
- **Connection Waypoints**: used to connect and create paths between the other waypoints of the location. Characters only walk through these waypoints when they need to go to other waypoints.

The actors are placed on the scene according to the actions they are performing. If an actor is executing a *GoIn* action, he will be placed on the first entrance waypoint that is not occupied by another actor. If the actor is performing a *Talk* action, he will be placed in the first acting waypoint available. The position and angle of the actors are defined according to the information provided by the waypoints. However, the size of the actors must be automatically calculated by system.

As in the real world, the closer the actor is to the camera, the larger it must appear to be in relation to the rest of the scene. Accordingly, in our method the width A_w and height A_h of an actor A in a location L are given by:

$$A_w(A,L) = A_v^w\left(\frac{\alpha(A,L)}{100}\right)$$

$$A_h(A,L) = A_v^h\left(\frac{\alpha(A,L)}{100}\right)$$

where A_v^w and A_v^h represent the original size (width and height) of the video of actor *A*, and the function *α*(*A*, *L*) computes the relative size of the actor through a linear interpolation between the front line L_{F2}^{pos} and far line L_{F1}^{pos} according to his current position A_y and his relative size on the front and far lines (L_{F2}^{size} and L_{F1}^{size}):

$$\alpha(A,L) = L_{F1}^{size}\big(1-\gamma(A,L)\big) + L_{F2}^{size}\big(\gamma(A,L)\big)$$

where *γ*(*A*, *L*) is a function that normalizes the current position A_y of the actor *A* in the interval of [0,1]:

$$\gamma(A,L) = \frac{A_y - L_{F1}^{pos}}{L_{F2}^{pos} - L_{F1}^{pos}}$$

Figure 6.7 shows an example of scene containing an actor placed in two different waypoints with his size calculated using the proposed method.

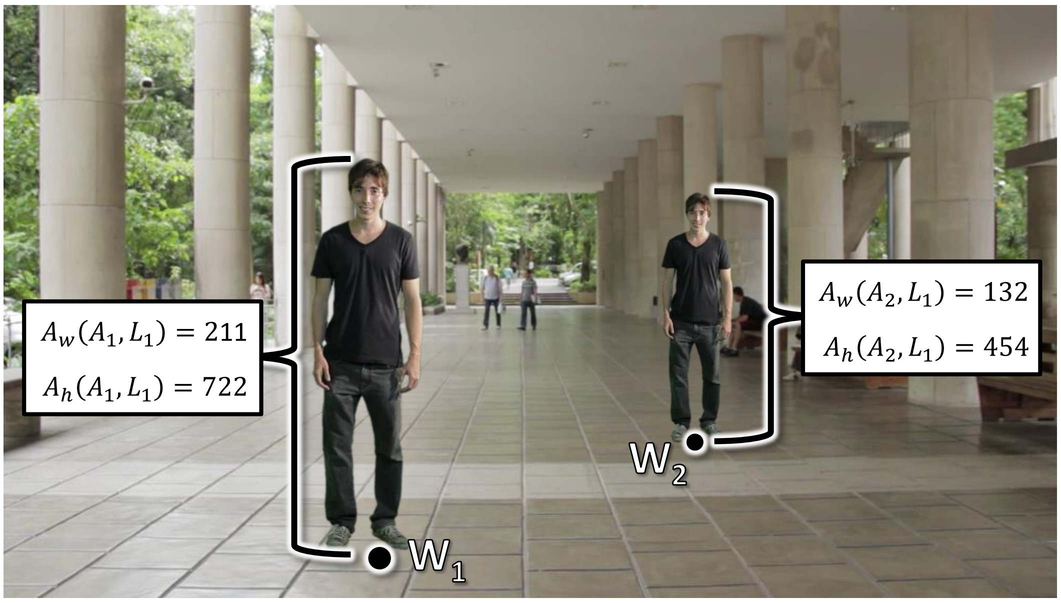


Figure 6.7: Example of scene using the proposed method to calculate the size of the actor.

Although the actors are initially placed over the waypoints, their position may change during the dramatization of the action. Thus, every time the position of an actor is modified, his relative size is recalculated and updated.

Once the position, angle and size of all actors involved in the action are properly defined, the next step of the process comprises the definition of a “line of action”, which is used to maintain the spatial continuity of the scenes. As presented in Chapter 3, the line of action (or action axis) consists of an imaginary line connecting the most important elements or directing the focus of the action in a scene. When shooting a scene, the cameras must be placed only at one side of this line (180 degree rule). The placement of the camera in different positions and angles in the same scene must occur only within the 180 degree arc. When shooting two consecutive shots of the same subject from inside of the 180 degree arc, the camera angle for the new shot must be at least 30 degrees from the angle of the previous shot (30 degree rule). In this way, the two shots can be considered different enough to avoid jump cuts (Mascelli 1965), which is an undesirable effect that causes visual jumps in either space or time of the film. These rules help to maintain the visual continuity of consecutive shots, and keep the narrative moving forward logically and smoothly, without disruptions in space or time (Brown 2011).

In order to establish the virtual line of action in the scene, the Cameraman agent adopts some common guidelines presented by Hawkins (2005) and Thompson and Bowen (2009), which state that in a scene with a single character, the line of action usually is given by the initial direction of the character. In scenes involving more characters, it is established by a line connecting the two most important characters in the scene. In this way, the virtual line of action is defined based on the position and orientation of the characters participating in the action as illustrated in Figure 6.8.

6.2.2.2. Camera Placement and Definition

After defining the position and orientation of all actors and establishing the line of action, the next step of the compositing process comprises the definition of the virtual cameras that can be used to film the scene. According to Brown (2011), camera placement is a key decision in storytelling. It determines what the audience sees and from what perspective they see it. Each shot requires placing the camera in the best position for viewing characters, setting and action at that particular moment in the narrative (Mascelli 1965). The approach employed to accomplish this task in the proposed system is based on the use of some standard arrangements for camera placement.

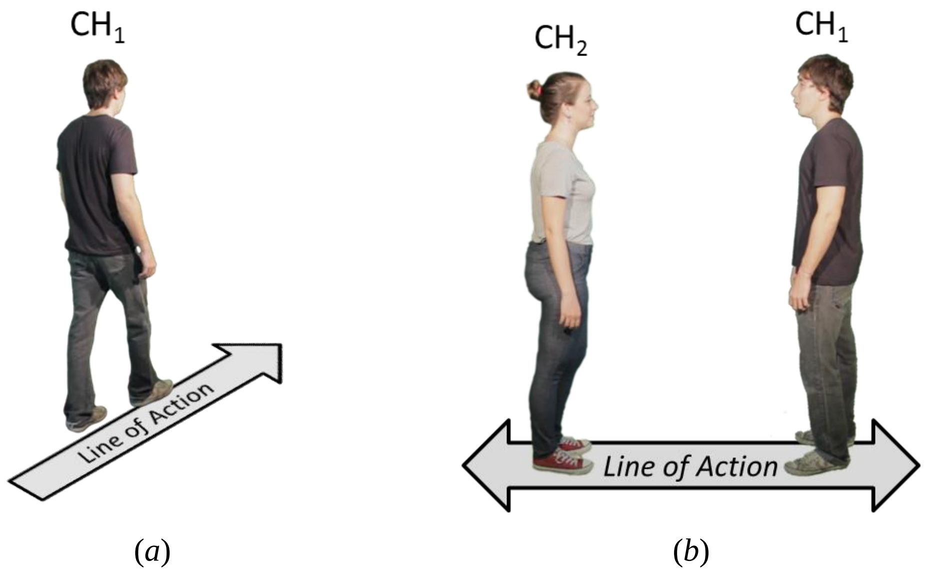


Figure 6.8: Examples of line of action. Image (*a*) shows a line of action established for a scene of a character (CH$_1$) walking in a direction *d*. Image (*b*) shows the line of action for scene of a dialog between two characters CH$_1$ and CH$_2$.

The cinematography theory defines standard patterns for camera placement depending on the type of scene (Arijon 1976; Katz 1991; Kenworthy 2009). These patterns act as a guide for possible choices of initial camera placement, with the final configuration depending on the constraints of the scene (Hawkins, 2005). In a scene of a dialog between two characters, for example, it is common to use the pattern known as the triangle system, whereby all possible shots for any subject are taken from three points forming a triangle within the currently chosen side of the line of action (Figure 6.9).

The four configurations of the triangle system illustrated in Figure 6.9 can be combined to multiply the camera options. The complete triangle system offers 7 camera viewpoints contained within a triangular formation (Figure 6.10).

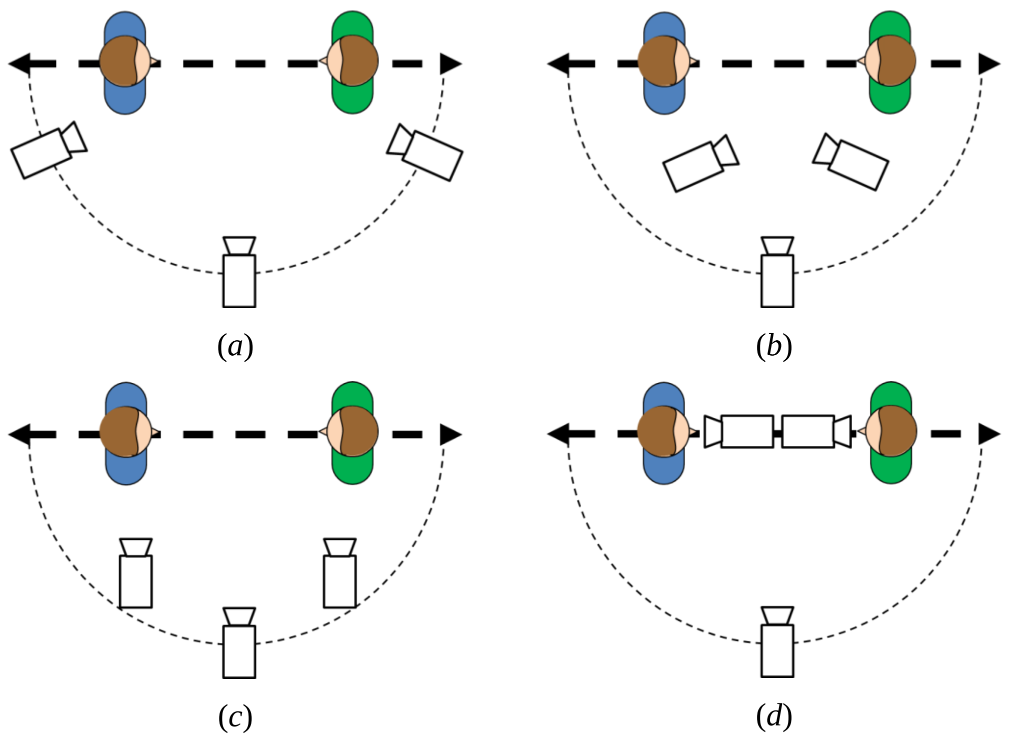


Figure 6.9: Triangle Systems. Image (*a*) shows the over-the-shoulder triangle system; image (*b*) shows the 45° triangle system; image (*c*) shows the profile triangle system; and image (*d*) shows the close-up triangle system.

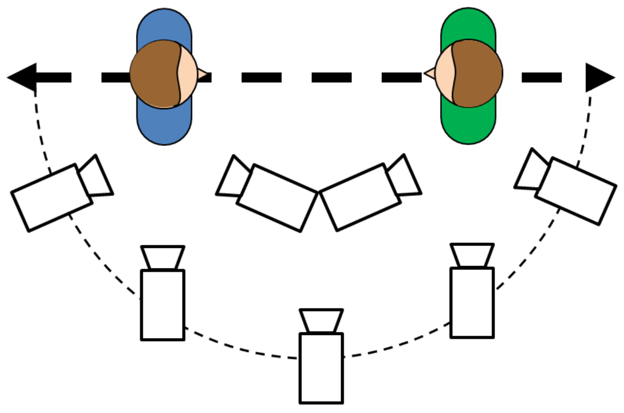


Figure 6.10: Full Triangle System.

Although the camera placement patterns provide the basic guidelines on how to place the cameras, not all angles can be produced using the pre-recorded videos of actors and locations filmed during the production process. However, the wide variety of angles and the high-definition video resolution of the material provide to the system the ability of simulating most of the possible cameras by manipulating the angle, type of shot and subjects. Figure 6.11 illustrates some of the shots that can be simulated using the video material available in a scene of a dialog between two characters.

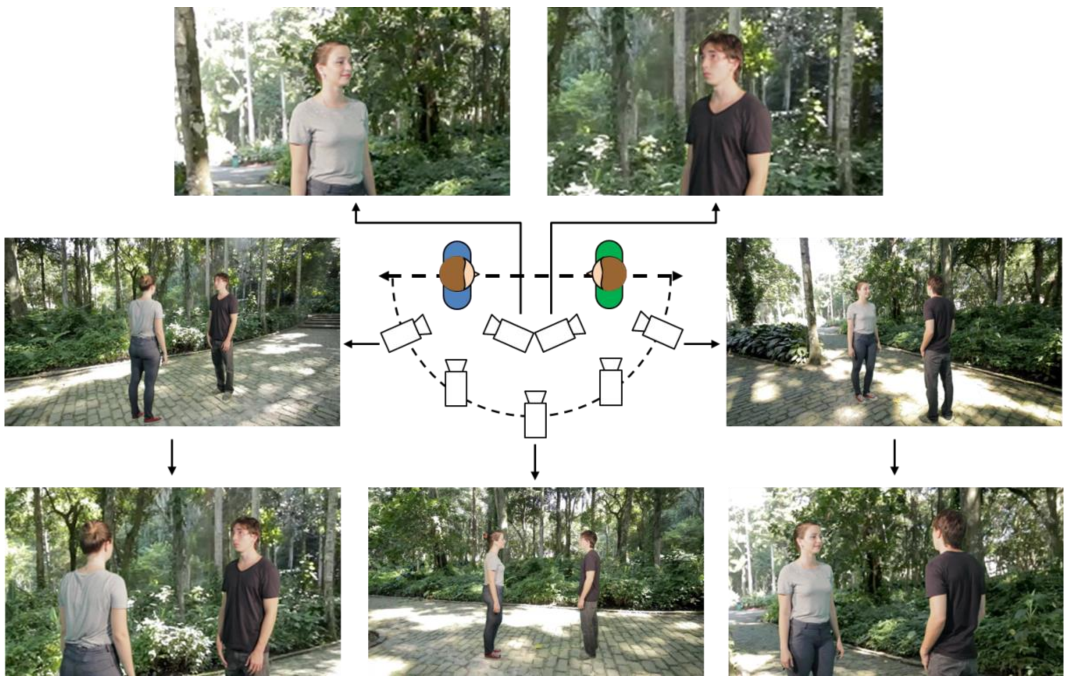


Figure 6.11: Examples of shots that can be simulated using the video material available in a scene of a dialog between two characters.

Each camera is defined by 4 parameters:

1. **Angle**: defines the angle and the position of the camera. There are 5 possible angles at one site of the line of action;
2. **Target Subjects**: defines the target subjects of the camera. The targets will be centralized in the shot;
3. **Shot Type**: determines the type of shot used by the camera. There are 5 possible types of shot: long shot, medium long shot, medium shot, medium close-up and close-up;
4. **Movement Type**: defines the type of movement executed by the camera. There are 4 types of camera movements:
    - **a. Static**: no camera movements are executed. The camera remains static in the initial configuration;
    - **b. Follow**: the camera follows the subjects and keeps them on the frame;

c. **ZoomIn**: the camera performs a zoom in movement, starting from the initial configuration and ending with the type of shot defined by a parameter;
d. **ZoomOut**: the camera performs a zoom out movement, starting from the initial configuration and ending with the type of shot defined by a parameter.

In order to simulate different types of shots using the pre-recorded videos, the system relies on the high-definition resolution of the video material. As explained in Chapter 5, during the production process the videos of actors and locations are both recorded using high-definition video resolution. Actors are filmed in a long shot (which shows their whole body), and locations are filmed in a very long shot (which includes the whole environment). The high quality of the videos permits the system to zoom in and zoom out during the compositing process, which allows the system to generate different types of shots and camera movements. As a consequence, the output video produced by the system is in a standard-definition resolution, which avoids the degradation of the video quality. If a higher output resolution is required, the video material has to be recorded in an even higher resolution. In the experiments conducted during the development of this thesis, the video material was recorded in full HD resolution (1080p), and output produced by the system is in SD resolution (480p).

The process of simulating the virtual camera during the compositing process is based on the concepts of "*World*", "*World Window*" and "*Viewport*", commonly used in computer graphics to specify coordinates systems (Hughes et al. 2013). As illustrated in Figure 6.12, the *Scene World* of the video-based dramatization is composed by the entire scene, including the video or image layers of the location and all the other elements that composed the scene. The *Camera Window* specifies the rectangular region of the video to be filmed, and is defined by the position ($shot_x$, $shot_y$) and size ($shot_w$, $shot_h$) of the shot, which are calculated by the system based on the camera parameters. The *Viewport* represents the rectangular region used to project and display the video filmed through the *Camera Window*.

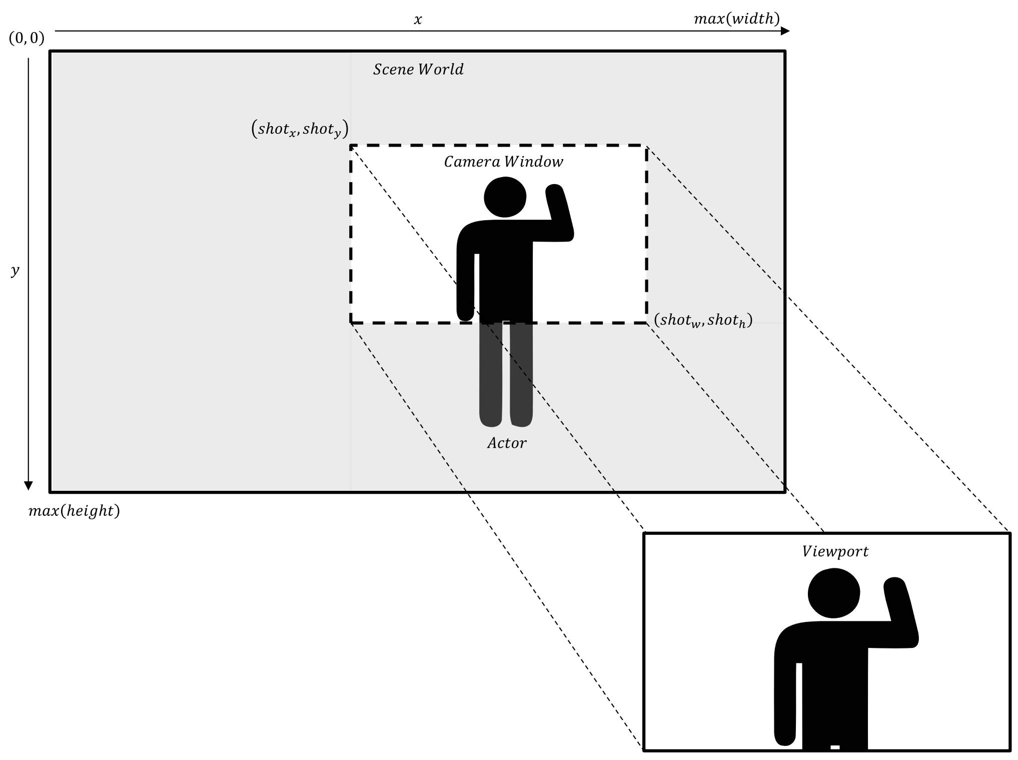


Figure 6.12: Window system.

The position (*shot*$_x$, *shot*$_y$) and size (*shot*$_w$, *shot*$_h$) of the *Camera Window* are calculated according to the type of shot and the target subjects (E) of the selected camera. The very long shot, which is the widest shot and include the whole scene, is given by:

$$shot_x(E) = 0$$
$$shot_y(E) = 0$$
$$shot_w(E) = max(width)$$
$$shot_h(E) = max(height)$$

The other types of shot are given by:

$$shot_x(E) = \varphi(E) - \frac{shot_w(E)}{2}$$
$$shot_y(E) = min(E_y) + \omega(E, \beta)$$
$$shot_w(E) = shot_h(E) * \frac{max(width)}{max(height)}$$
$$shot_h(E) = \Delta\left(max(E_y),\ min(E_y)\right) - \omega(E, \alpha)$$

where $\varphi(E)$ determines the central position of the subject elements E in a scene, and $\omega(E,c)$ is a functions that uses the constants α and β to determine the level of zoom and the height of the camera:

$$\varphi(E) = \left(\frac{\Delta(max(E_x),\ min(E_x))}{2}\right) + min(E_x)$$

$$\omega(E,c) = \Delta\big(max(E_y),\ min(E_y)\big) * c$$

The following values are used in the constants α and β to compute the respective types of shot:

- **Long shot:** $\alpha = -0.21,\ \ \beta = -0.14$
- **Medium long shot:** $\alpha = 0.15,\ \ \beta = -0.1$
- **Medium shot:** $\alpha = 0.42,\ \ \beta = -0.08$
- **Medium close-up shot:** $\alpha = 0.58,\ \ \beta = -0.06$
- **Close-up shot:** $\alpha = 0.74,\ \ \beta = -0.03$

The equations used to calculate the position and size of the *Camera Window* make the process of performing camera movements during the compositing process easier. In a static camera, the position and size of the *Camera Window* are only calculated once at the beginning of the scene. In order to implement a camera that follows the subjects, the position of the *Camera Window* just has to be recalculated in each frame of the scene. Zoom in and zoom out movements also can be easily achieved by performing a linear interpolation between the initial values of α and β and the final values in the target type of shot.

6.2.2.3. Video Editing

In filmmaking, the video editing process occurs during the post-production phase, where the editor selects the best shots from the raw footage, and combines them into sequences to create a finished motion picture. However, in video-based interactive storytelling, there is no post-production phase. All the editing tasks must be done in real-time during compositing process. It is similar to a live TV show or a live sport transmission, where the editor has to switch between different cameras without knowing beforehand the actions taken by all the actors.

When it comes to narrative storytelling, camera angle selection is a crucial editing decision. According to Brown (2011), a carefully-chosen camera angle can heighten the dramatic content of the story, while a carelessly picked camera angle may distract or confuse the audience by depicting the scene so that its meaning becomes difficult to be comprehended. Therefore, the selection of camera angles is one of the most important factors in constructing a picture of continued interest (Mascelli 1965). However, cinematography does not define strict rules on how to select the best shots. Usually, each director and editor has his own style and he/she defines the shots according to his/her own knowledge and preferences.

In the proposed system, the real-time video editing task is performed by the Editor agent, which uses cinematography knowledge of video editing to select the best cameras to film the scenes and to keep the temporal and spatial continuity of the film by avoiding jump cuts and selecting the most adequate shot transition for the scenes.

The first step of the video editing phase comprises the process of selecting the best camera configuration to film the action. The proposed approach to solve this problem consists of encoding the knowledge of a real film editor into our system. This knowledge is represented by means of several artificial neural networks trained to solve cinematography problems involving camera shot selection.

The proposed model to represent the knowledge of a real film director is illustrated in Figure 6.13. For each type of scene (e.g. dialog scene, chasing scene, fighting scene), there are two neural networks: the first one is trained to classify the best camera angle for the shot based on geometric information extracted from the scene; and the second is trained to select the best type of shot based on the camera angle selected by the first neural network and emotional information extracted from the characters participating in the scene. Distinct neural networks are used for each type of scene because the number of input and output variables available depends on the type of scene and the number of involved actors. Moreover, the choices of camera angle and shot type may change substantially in different types of scenes.

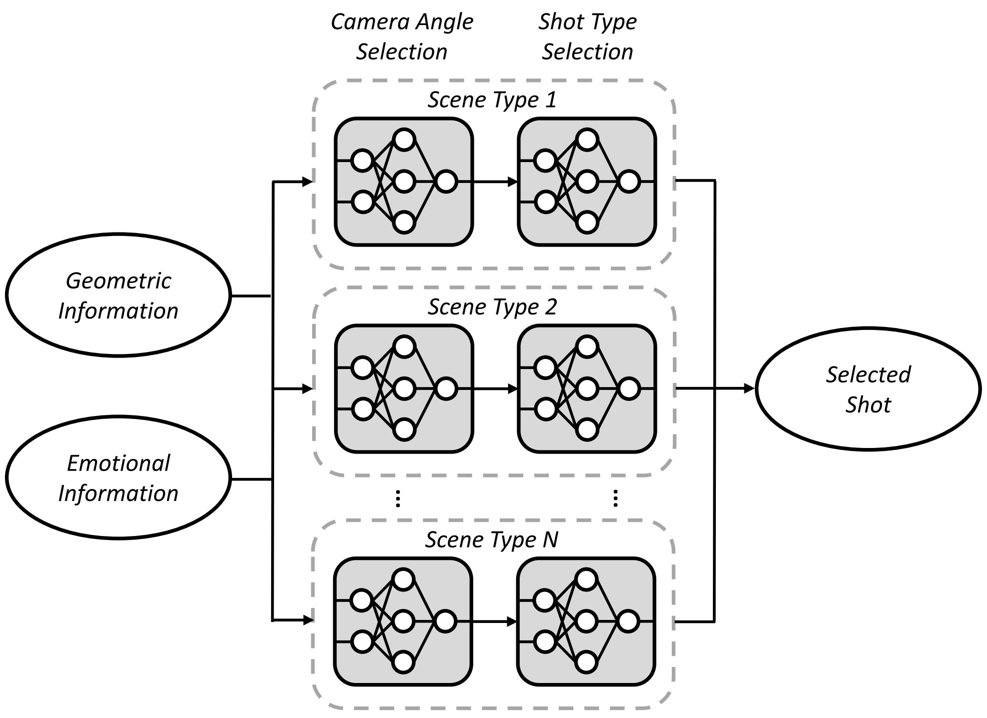


Figure 6.13: Neural network system.

The proposed method uses single hidden layer neural networks trained by a standard back-propagation learning algorithm using a sigmoidal activation function (Rumelhart et al. 1986). The structure of this type of neural network is defined in terms of input, output, and hidden layers. The input of the neural network used for selecting the best camera angle comprises a set of geometric features extracted from the scene setup. It includes the angle and position of the characters participating in the action (X, Y and Z-index, relative to the center of the scene and arranged considering the order of importance of the characters in the scene), and the id of the action performed by the main character. The number of input values depends on the type of scene and the number of characters involved in the action. For example, a dialog scene between two characters includes 9 input values, and consequently, 9 nodes in the input layer of the neural network for this type of scene. The output of this neural network comprises the possible camera angles proposed by the Cameraman agent during the camera placement phase. For example, in a scene of a dialog between two characters, there are 3 possible camera angles, and consequently, 3 nodes in the output layer of this neural network (Figure 6.14). When the output is calculated, the activated neuron in the output layer indicates the selected camera angle to be used in the shot.

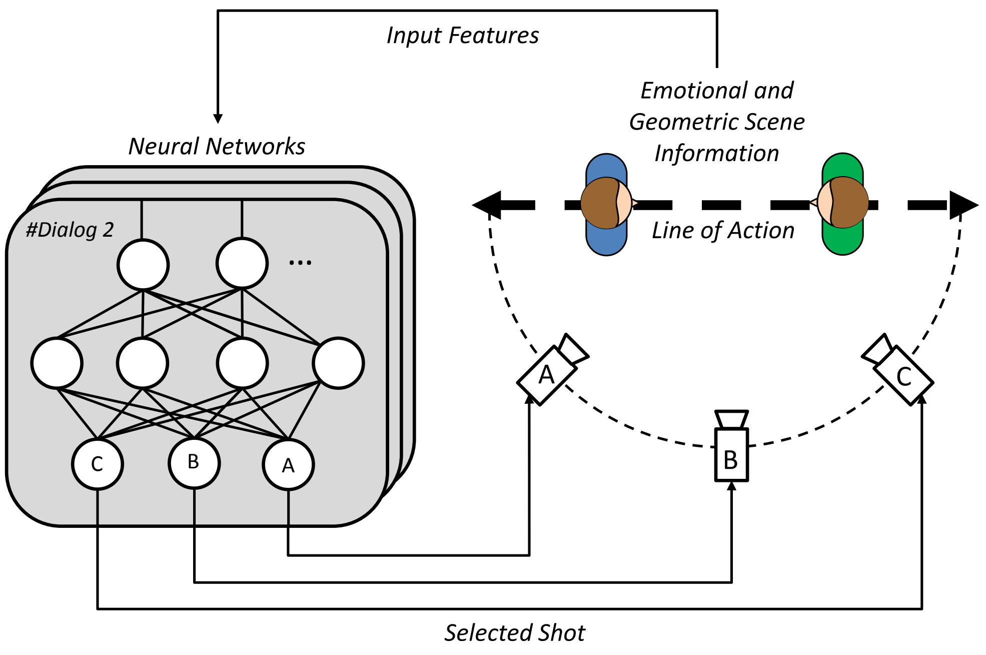


Figure 6.14: Structure of the camera selection neural network for a scene of a dialog between two characters.

With the camera angle selected by the first neural network, the next step is the selection of the type of shot. Usually, the decision of the best type of shot depends on the emotional content of the scenes (Mascelli 1965). More intimate shots, like close-ups, are often employed when there is a substantial change in the emotions of characters, highlighting the facial expressions of the subjects (Bowen and Thompson 2009). The input of the neural network used for selecting the best type of shot comprises a set of emotional features extracted from the Emotions and Relations Network (presented in Section 6.1.3). It includes the variation (relative to the previous shot) of the six emotions and the relations of characters participating in the action, together with the id of the camera angle selected by the first neural network. The number of input values depends on the number of characters involved in the action. For example, a dialog scene between two characters includes 15 input values (15 nodes in the input layer). The output of this neural network is composed of 5 nodes, which represent the five most common types of shot (close-up, medium close-up, medium shot, medium long shot and long shot). When the output is calculated, the activated neuron in the output layer indicates the selected type of shot.

Artificial neural networks are not intelligent by themselves – they need to be trained with a collection of training samples to create a classification function capable of recognizing similar situations in the future. In order to collect samples to be used as training data for the neural networks, we simulated 50 scenes and, for each one, the best shot (angle and shot type) was selected according to the vision of a real film editor. Each decision generates one training sample, which includes all the features used as input for the neural networks, together with the selected camera angle and shot type for the simulated scene. Once the neural networks were trained, they can be used in real-time to select the best cameras to film the scenes.

After selecting the best camera configuration to film the action, the next step of the video editing phase comprises the verification of the occurrence of jump cuts and the selection of the most adequate shot transition.

Every time a new camera angle is selected to film the scene, a transition between shots occurs. An important principle of cinematography is that such transitions should be unobtrusive and sustain the audience's attention on the narrative (Mascelli 1965). With an effective editor, the audience will not notice how shots of various frame sizes and angles are spliced together to tell the story. One way of complying with this principle is to avoid jump cuts. As detailed in Chapter 3, a jump cut is often regarded as a mistake in classical editing (Butler 2002). It usually occurs when two very similar shots of the same subject are joined together by a cut, producing the impression that the subject "jumps" into a new pose, causing a disorientation effect in the audience. There should be a definite change in image size and viewing angle from shot to shot. Another important cinematography principle used by conventional editors to join and maintain the continuity between shots is the use of adequate scene transitions. As introduced in Chapter 3, there are four basic ways to transit from one shot to another: cut, dissolve, wipe and fade. Each type of transition has its own applications and meanings.

Both detection of jump cuts and selection of shot transitions are based on visual similarities between shots. However, at this phase of the compositing process, the image frames of the scene have not yet been generated yet. In order to perform a visual comparison of the shots, the Editor agent requests to the Scene Composer agent the generation of a simulated first frame of the current scene based on the information that have been defined during the previous steps of the compositing process. Moreover, the agent also keeps a copy of the last frame of the last scene that has been composed by the system. In this way, it is possible to analyze the visual similarities between the shots and make the most adequate editing decision.

The proposed approach to create a computer program that is able to automatically edit video segments consists in translating cinematography principles and practices directly into logical rules. We propose the use of a similarity scale to classify the transition between two consecutive shots and to detect possible jump cuts. Firstly, given two consecutive shots C_x and C_y, we calculate the histogram C_x^h of the last frame of C_x and the histogram C_y^h of the first frame of C_y. Then we use a metric such as a correlation coefficient to determine the degree of similarity between shots:

$$F_{coor}(C_x, C_y) = \frac{\sum_i (C_x^h(i) - \overline{C_x^h})(C_y^h(i) - \overline{C_y^h})}{\sqrt{\sum_i (C_x^h(i) - \overline{C_x^h})^2 \sum_i (C_y^h(i) - \overline{C_y^h})^2}}$$

where $C_x^h(i)$ and $C_y^h(i)$ are the histogram values in the discrete interval i and $\overline{C_x^h}$ and $\overline{C_y^h}$ are the histogram bin averages. High values of correlation represent a good match between the frames, that is: $F_{coor}(C_x, C_y) = 1$ represent a perfect match and $F_{coor}(C_x, C_y) = -1$ a maximal mismatch.

Using the histogram correlation coefficient, we define three classes of similarity $\{S_1, S_2, S_3\}$. These classes can be expressed by the following rules:

- If $F_{coor}(C_x, C_y) \in (\beta,\ 1]$ then similarity class is S_1.
- If $F_{coor}(C_x, C_y) \in [\alpha,\ \beta]$ then similarity class is S_2.
- If $F_{coor}(C_x, C_y) \in [-1,\ \alpha)$ then similarity class is S_3.

The similarity scale and the classes of similarity are illustrated in Figure 6.15. The similarity class S_1 represents a class of high similarity between frames and no editing procedure is required. Transitions of videos with similarity S_1 are not noticed by the audience. The similarity class S_2 represents a condition that causes jump cuts. In this case, no shot transition can be applied and the Editor agent should select a new shot for the next scene. The similarity class S_3 represents a class of low similarity between frames. In this case, the transition between videos can be done using a cut, dissolve, wipe or fade, without causing jump cuts.

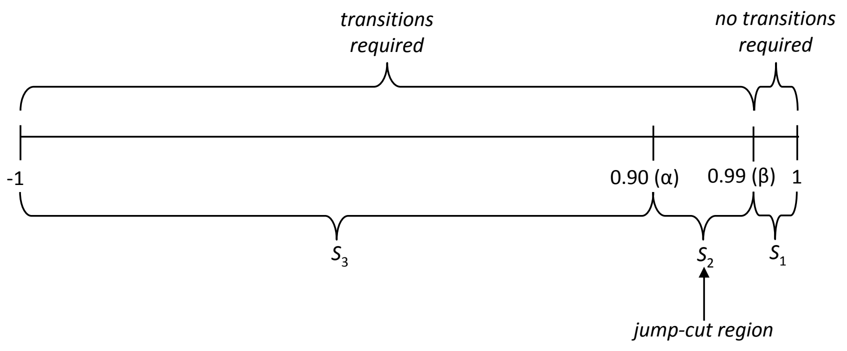


Figure 6.15: Similarity Scale (the values of α and β are experimental).

When a transition between two shots is classified as similarity class S_2, the video sequence must be changed to avoid the jump cut. If the next scene is a static one, the Editor agent uses the extra video source provided by the master scene video of the static scene to show the same action from a different camera angle, which guarantees an S_3 situation without breaking the film continuity and avoiding the jump cut; otherwise, if the scene is being dynamically composed by the system, the Editor agent returns to the previous step and selects a new camera angle and shot type based on the second most activated output neuron of the neural networks, which guarantees a different camera angle and S_3 situation in the similarity scale.

The transitions between shots classified in the similarity class S_3 lead us to the next step of the editing process: the selection of adequate transitions. To identify the most adequate transition to join two consecutive shots, we formulate a set of rules based on the cinematography literature to classify the shots into the four basic classes of transitions. Considering the functions $L_T(x)$ and $L_S(x)$ that return the temporal and spatial locations of a video segment x based on its plot action chain, we define the following transition rules:

$$\forall C_x \forall C_y \left(L_T(C_x) = L_T(C_y)\right) \wedge \left(L_S(C_x) = L_S(C_y)\right) \Rightarrow T_{cut}$$

$$\forall C_x \forall C_y \left(\left(L_T(C_x) \neq L_T(C_y)\right) \vee \left(L_S(C_x) \neq L_S(C_y)\right)\right) \wedge \left(F_{coor}(C_x, C_y) \geq 0.75\right) \Rightarrow T_{dissolve}$$

$$\forall C_x \forall C_y \left(\left(L_T(C_x) = L_T(C_y)\right) \wedge \left(L_S(C_x) \neq L_S(C_y)\right)\right) \wedge \left(F_{coor}(C_x, C_y) \leq 0.75\right) \Rightarrow T_{wipe}$$

$$\forall C_x \forall C_y (\neg C_x \vee \neg C_y) \Rightarrow T_{fade}$$

Video segments classified as class T_{cut} are ready to be processed and the direct cut can be executed. The video segments classified as class $T_{dissolve}$, T_{wipe} or T_{fade} must pass through another analyzer to determine the duration of the transition. In a conventional editing process, the editor usually uses the duration of a transition to represent the temporal variation that occurs during the transition. Based on this idea and considering t the exhibition time (usually in minutes or seconds) and T the story time (usually hours or years), the duration of the transition t_d is given by:

$$t_d(C_x, C_y) = t_{min} + \left(\frac{(\Delta L_t(C_x, C_y) - T_{max})(t_{max} - t_{min})}{T_{max} - T_{min}}\right)$$

where T_{max} and T_{min} represent respectively the maximal and minimal temporal variation in the story time, and the variables $t_{\max}$ and $t_{\min}$ represent the maximal and minimal duration of the transition. Usually $t_{min} = 0.5$ and $t_{max} = 2.0$ seconds (the minimal and maximal time for a transition, using dissolve transition as a reference).

Figure 6.16 illustrates the process of computing the transition between two consecutive shots using the proposed editing method.

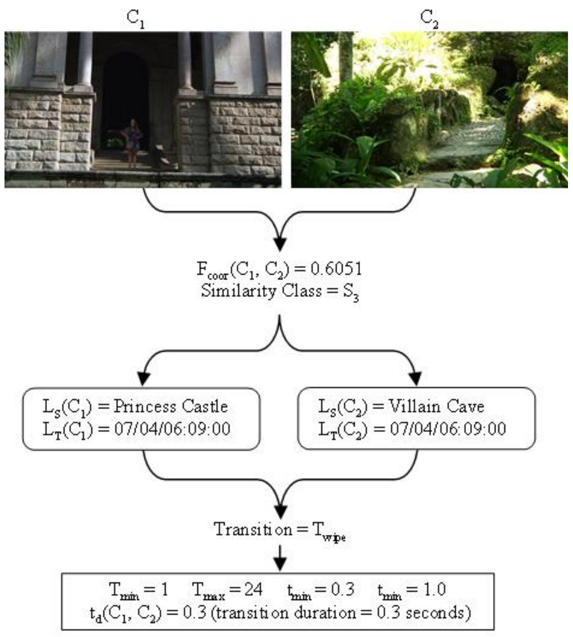

C1
C2
$F_{coor}(C_1, C_2) = 0.6051$
Similarity Class = S_3
$L_S(C_1)$ = Princess Castle
$L_T(C_1)$ = 07/04/06:09:00
$L_S(C_2)$ = Villain Cave
$L_T(C_2)$ = 07/04/06:09:00
Transition = T_{wipe}
$T_{min} = 1$ $T_{max} = 24$ $t_{min} = 0.3$ $t_{min} = 1.0$
$t_d(C_1, C_2) = 0.3$ (transition duration = 0.3 seconds)

Figure 6.16: Example of a transition computation.

6.2.2.4. Color and Lighting Effects

In films, emotions can be expressed not only through the dramatization of actors, but also through colors, lighting and other visual effects. In filmmaking, the director of photography is responsible for the quality of the photography and the cinematic look of the film. Using his/her knowledge of lighting, lenses, cameras, and film emulsions, the director of photography creates the appropriate mood, atmosphere, and visual style of each shot to evoke the emotions required for each scene (LoBrutto 2002).

In the proposed video-based dramatization system, the Director of Photography is the agent responsible for defining the visual aspects of the narrative, manipulating the illumination and applying lens filters to improve and create the emotional atmosphere of scenes. The proposed approach to create an autonomous agent capable of performing this task in real-time consists of encoding the knowledge of a real director of photography into our system. This knowledge is represented by means of an artificial neural network trained to solve cinematography problems involving the selection of the emotions of scenes.

The neural network used to represent the knowledge of the Director of Photography follows the same formalism of the one used by the Editor agent. Its input comprises a set of 6 emotional features, the relations of the characters participating in the action, and another feature describing the mood of the location where the scene is happening. All features are extracted from the Emotions and Relations Network and their values are calculated in a way to reflect the overall mood of the whole scene described by a single automaton according to the intensity of the characters' emotions and their importance to the narrative.

The emotion ε of an actor *A* in a scene *S* is given by:

$$A_\varepsilon(S) = max([A_\varepsilon^1, A_\varepsilon^2 \dots A_\varepsilon^n] \in S) * A_i$$

where *max* is a function that returns the maximum intensity value of the emotion ε of the actor *A* along the scene, and $A_i \in [0 \dots 1]$ is the importance factor of the actor in the narrative.

Considering a scene composed of a set of actors S_α, the overall emotion of the scene to be used as input to the neural network can be calculated by:

$$S_\varepsilon(S_\alpha) = max\big(A_\varepsilon(S_\alpha)\big)$$

The scene affectivity can be calculated based on the affectivity average of the actors participating in the scene. Considering $A_{\beta(x,y)}$ the set values of affectivity of an actor x to an actor y along a scene *S*, the actors' affectivity can be described as:

$$A_\beta(S) = \frac{1}{n}\sum_{i=0}^{n-1} A_{\beta(x,y)}^i \in S$$

Similarly, the affectivity of a scene composed of a set of actors S_α can be calculated by:

$$S_\beta(S_\alpha) = \frac{1}{n_\alpha}\sum_{i=0}^{n_\alpha - 1} A_\beta\big(S_\alpha^i\big)$$

The output of the neural network comprises a set of emotional profiles, which describe specifics moods and the visual effects that must be simulated in order to produce their respective emotions in the scene. Table 6.1 shows examples of emotional profiles used in the implementation of the proposed video-based dramatization system.

Profile	**Visual Effect**
Sad Scene	Soft sepia lens filter.
Fear Scene	Low brightness and soft film grain effect.
Anger Scene	Warm lens filter.
Happy Scene	High brightness.
Tension Scene	Soft warm lens filter.

Table 6.1: List Emotional profiles used by the Director of Photography agent.

In order to collect samples to be used as training data for the neural network, we simulated 50 scenes and, for each of them, the best emotional profile was selected according to the vision of a real director of photography. Each decision generates one training sample, which includes all the features used as input for the neural network, together with the selected emotional profile for the simulated scene. Once the neural network is trained, it can be used in real-time to select the best visual effects to represent the scenes' emotions.

6.2.2.5. Frame Compositing

After defining the whole structure of a scene, starts the actual process of compositing video frames representing the scene based on the information defined in the previous steps of the compositing process. In the proposed architecture, this task is performed by the Scene Composer agent, who employs parallel and optimized image processing algorithms to assemble the multiple visual elements that compose the scene into a single piece of motion picture in real-time.

The process of compositing video frames is the most time-consuming task and must be performed in real-time. The system must be able to generate at least 30 frames per second. In order to accomplish this task, it is proposed a parallel architecture capable of managing and compositing multiple video frames at the same time (Figure 6.17). In this architecture, the Scene Compositing Control manages the compositing process and the execution of several threads that are responsible for compositing the frames. To each thread, a specific frame of the scene is assigned and, when a thread finishes compositing a frame, it is added to the Frame Buffer, which is an ordered list of frames that are ready for exibition.

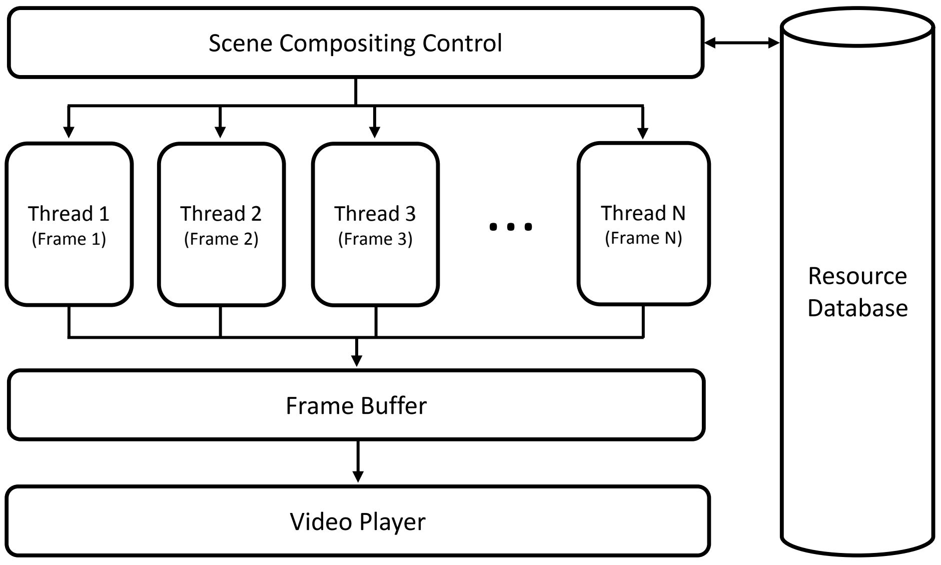


Figure 6.17: Parallel video compositing architecture.

Each compositing thread generates its assigned frame according to the information provided by the scene structure. The pseudocode of compositing algorithm is illustrated in Figure 6.18. The algorithm receives the id of the frame that has to be generated (`frame_id`) and a reference to the current scene structure (`scene_structure`). The compositing process starts by retrieving the background frame (`bg_frame`) of the current scene location defined in the scene structure according to the angle chosen by the Editor agent to film the scene. It is important to notice that the frame is retrieved based on the id of the frame that has been assigned to the thread (`frame_id`). Then, for each scene element present in the scene structure (actors and location layers), the algorithm retrieves its frame (`element_frame`) and alpha mask frame (`mask_frame`) based on the `frame_id` and orientation of the element in the scene. Next, the `element_frame` and `mask_frame` are combined to create an RGBA image (`alpha_frame`) that uses the `mask_frame` as the alpha channel of the image. Then, the `alpha_frame` is resized according to the width and height of the scene element that were defined in the previous steps of the compositing process. The next step consists of a clipping operation, which is performed over the `alpha_frame` in order to eliminate parts of the element that are not inside of the frame region defined by the angle and type of shot selected by the Editor agent. Before blending the `alpha_frame` with `bg_frame`, a color correction operation is performed to adjust the color of `alpha_frame` according to the color of its area in `bg_frame`. Then, an alpha GPU compositing operation is performed to blend the resulting `alpha_frame` with the `bg_frame`, which completes the compositing process of the scene element. Once all scene elements have been composed, the emotional profile selected by the Director of Photography is simulated by applying some color filters and lighting effects in `bg_frame`. After applying the visual effects, the algorithm returns the composed frame (`bg_frame`).

```
function compose_frame(frame_id, scene_structure)

  get bg_frame of frame_id from the location defined in
                                                    scene_structure

  foreach scene_element in scene_structure do
    get element_frame and mask_frame of frame_id from
                                                      scene_element
    combine element_frame and mask_frame to create a alpha_frame
    resize alpha_frame according to the size of scene_element
    perform clipping operation in alpha_frame
    correct the color of alpha_frame based on bg_frame
    perform an alpha GPU compositing operation blending
                                            alpha_frame with bg_frame
  end

  apply color and lighting effects in bg_frame to simulate
                                      the selected  emotional profile

  return bg_frame
end
```

Figure 6.18: Pseudocode of the compositing algorithm.

The process of creating the alpha frame consists in adding the alpha mask to the RGB video frame of the scene element as an alpha channel, which creates an RGBA image. This image contains, besides the RGB color information, an extra alpha channel that retains the matte information. Combining the frames into a single image simplifies the resize operation that is performed in the next step of the compositing algorithm. The scaling algorithm used to resize the frames is based on the nearest-neighbor interpolation method, which is the fastest image scaling algorithm implemented by the OpenCV library (Bradski and Kaehler 2008).

The clipping operation is used to eliminate parts of the scene elements that are not inside of the rectangular region of the frame, which is defined by the configuration of the virtual camera (position, angle and type of shot). The OpenCV library offers an optimized way of performing clipping operations by allowing the definition of a region of interest (ROI) in images. Image processing algorithms only operate inside of the region defined by the ROI. In this way, the clipping operation consists of calculating the region of the scene element that is inside of the rectangular region of the scene frame based on its position and size, and then adjusting the ROI of the scene element to match the scene frame.

The color correction algorithm used in the frame compositing process is based on the exposure compensation method proposed by Brown and Lowe (2007) to correct color differences in panorama image stitching, which is the process of combining multiple images with overlapping fields of view to produce a segmented panorama or high-resolution image. The approach proposed by Brown and Lowe (2007) adjusts the intensity gain level of the images to compensate for appearance differences caused by different exposure levels.

The exposure compensation method is used in the frame compositing process to adjust the exposure levels of the scene elements based on the background frame. The intensity gain of the frame I_i of a scene element is given by the error function defined by the sum of normalized intensity gain errors for all overlapping pixels in the background frame I_j:

$$e = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij}((g_i\bar{I}_{ij} - g_j\bar{I}_{ji})^2/\sigma_N^2 + (1-g_i)^2/\sigma_g^2$$

where N_{ij} is the number of pixels in frame i that overlap frame j, and $\bar{I}_{ij}$ is the mean value of frame I_i in the overlapping region of frame I_i and I_j. The parameters σ_N and σ_g represent the standard deviation of the intensity errors and the gain standard deviation respectively, which have been empirically set to

$\sigma_N = 10.0, I \in \{0..255\}$ and $\sigma_g = 0.1$ (Brown and Lowe 2007). The criterion employed to determine the intensity gain level aims at the minimization of the error function with respect to the gain g.

The alpha compositing operation is applied to blend the alpha frame of the scene element with the background frame. The algorithm uses an "over" operator to blend the color and alpha values of the images together on a pixel-by-pixel basis. Considering r_A, g_A and b_A the RGB components that define the color of a pixel in element A with alpha value α_A, and r_B, g_B and b_B the RGB components that define the color of a pixel in the background element B with alpha value α_B, the resultant RGB color components of the pixel r_C, g_C and b_C are given by:

$$\begin{aligned} r_C &= \alpha_A r_A + \alpha_B r_B (1 - \alpha_A) \\ g_C &= \alpha_A g_A + \alpha_B g_B (1 - \alpha_A) \\ b_C &= \alpha_A b_A + \alpha_B b_B (1 - \alpha_A) \end{aligned}$$

The alpha compositing algorithm runs on GPU and takes advantage of its parallel architecture to compute the color of several pixels simultaneously, which improves the performance of the process and allows the system to compose the interactive scenes in real-time.

The process of applying the emotional profile selected by the Director of Photography in the generated frames consists of simulating lens filters and modifying the brightness and contrast of final frame according to the selected profile. Lens filters are simulated by overlaying a colored image layer over the final frame. The opacity of the image layer controls the intensity of the simulated filter. The brightness and contrast are automatically adjusted to match the description of the emotional profile. Both operations are performed on GPU to improve the performance of the compositing process.

6.2.3. Film Scoring

In filmmaking, scoring consists in the process of writing and compositing the soundtracks of the film. Music is an integral part of the film, as it helps to connect the emotional content with the events on the screen. During the post-production phase, the music director is the person in charge of creating the mood and the emotional atmosphere of the film.

As a traditional film, a video-based interactive narrative must emphasize the dramatic content of the story and use music as a tool to express the emotions of the narrative. In the proposed video-based dramatization system, the agent Music Director is responsible for selecting the most adequate soundtracks and manipulating the audio of the narrative to create the emotional atmosphere of each scene according to the information provided by the Director agent.

6.2.3.1. The Music Director Agent

The proposed approach to create an autonomous agent capable of selecting the most adequate soundtracks to create the emotional atmosphere of the narratives in real-time consists of encoding the knowledge of a real music director into our system. This knowledge is represented by means of an artificial neural network trained to solve cinematography problems involving the selection of the best soundtracks for the narrative scenes.

The neural network used to represent the knowledge of the Music Director agent is very similar to the one used in the Director of Photography agent. Its input comprises the same set of emotional features extracted from the Emotions and Relations Network and its output consist of a set of emotional profiles describing specifics moods and the soundtracks that can be used to produce their respective emotions in the narrative. Table 6.2 shows examples of emotional profiles used in the implementation of the proposed video-based dramatization system.

In order to collect samples to be used as training data for the neural network of the Music Director agent, we simulated 50 scenes and, for each of them, the best emotional profile was selected according to the vision of a real music director. Each decision generates one training sample, which includes all the features used as input for the neural network, together with the selected emotional profile for the simulated scene. Once the neural network is trained, it can be used in real-time to select the best soundtracks for the narrative.

Scene	Audio
Happy Scene	Soundtrack with major keys; rapid tempos; high pitched with large variations.
Sad Scene	Soundtrack with minor keys; slow tempos; narrower range melodies.
Fear Scene	Soundtrack with rapid tempos; dissonance; small pitch variations.
Anger Scene	Soundtrack with minor keys; fast tempos; high pitched.
Tension Scene	Soundtrack with minor keys; ascending melodies; Dissonant harmonies.

Table 6.2: List Emotional profiles used by the Music Director agent.

6.3. Conclusion

This chapter described the technical details of the implementation of all cinematography-based agents that compose the proposed dramatization system, including the algorithms for real-time video compositing and editing.

The system was inspired by cinematography theory and designed to apply and respect the main cinematography principles and rules in the video compositing process. The proposed technique to represent the knowledge of some cinematography-based agents using a collection of artificial neural networks trained by their corresponding filmmaking professionals allows the system to learn the personal style of the human professionals and replicate it during the video compositing process, giving to the system the ability to apply the cinematography rules and principles while keeping the signature of the human artist in the computer generated content.

The next chapter will present the technical details about the implementation of the user interaction module and the multi-user interaction mechanisms of the proposed system.