CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

Autoria de Documentos Multimídia Interativos Baseada na Sincronização de Efeitos Sensoriais em Relação ao Conteúdo Audiovisual

Raphael Silva de Abreu

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientadores:
Joel André Ferreira dos Santos
Eduardo Bezerra da Silva

Rio de Janeiro,
20 de julho de 2018

Autoria de Documentos Multimídia Interativos Baseada na Sincronização de Efeitos Sensoriais em Relação ao Conteúdo Audiovisual

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Raphael Silva de Abreu

Aprovada por:

Presidente, Prof. Joel André Ferreira dos Santos, D.Sc. (orientador)

Prof. Eduardo Bezerra da Silva, D.Sc. (co-orientador)

Prof. Gustavo Paiva Guedes e Silva, D.Sc.

Prof. Glauco Fiorott Amorim, D.Sc. (CEFET/RJ)

Prof. Débora Christina Muchaluat Saade, D.Sc. (UFF)

Rio de Janeiro,

Julho 2018

CEFET/RJ – Sistema de Bibliotecas / Biblioteca Central

A162 Abreu, Raphael Silva de
Autoria de documentos multimídia interativos baseada na sincronização de efeitos sensoriais em relação ao conteúdo audiovisual / Raphael Silva de Abreu.—2018.
x, 86f. : il. (algumas color.) , grafs. , tab. ; enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca , 2018.
Bibliografia : f. 78-86
Orientadores : Joel André Ferreira dos Santos
Eduardo Bezerra da Silva

1. Sistemas multimídia. 2. Redes neurais (Computação). 3. NCL (Linguagem de marcação de documento). 4. Ciência da computação. I. Santos, Joel André Ferreira dos (Orient.). II. Silva, Eduardo Bezerra da (Orient.). III. Título.

CDD 006.6

Elaborada pela bibliotecária Lívia Lima CRB-7/5904

RESUMO

Autoria de Documentos Multimídia Interativos Baseada na Sincronização de Efeitos Sensoriais em Relação ao Conteúdo Audiovisual

Raphael Silva de Abreu

Orientador:
Joel André Ferreira dos Santos

Resumo da Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do título de mestre.

Avanços da computação ubíqua têm se mostrado importantes para aumentar os níveis de imersão de usuários em ambientes virtuais. Tais avanços impulsionam diversas pesquisas visando aumentar a imersão do usuário em aplicações multimídia. Uma delas é a adição de efeitos sensoriais, possibilitando as aplicações *mulsemedia*. Tais aplicações realizam interface com outros sentidos humanos além da visão e audição. Entretanto, o desenvolvimento destas aplicações necessita de um esforço de autoria para sincronizar efeitos sensoriais com conteúdo audiovisual. Além disso, aplicações interativas carecem de abstrações para facilitar a autoria de efeitos sensoriais. Portanto, este trabalho apresenta uma abordagem para facilitar a autoria de aplicações *mulsemedia* interativas. Para resolver essas questões, este trabalho se concentrou em três frentes. A primeira é o conceito de âncoras abstratas, que realiza a sincronização de efeitos sensoriais com um objeto de mídia de forma semiautomática. Neste contexto, é apresentado um processador para a linguagem NCL que utiliza redes neurais para identificar quando conteúdos são apresentados para realizar a sincronização. A segunda frente é uma arquitetura de rede neural bimodal, visando melhorar a identificação de conteúdos presentes em objeto audiovisual ao levar em consideração as modalidades de áudio e vídeo. A terceira frente é permitir a definição de efeitos sensoriais em linguagens multimídia declarativas, tornando possível a autoria de aplicações interativas com efeitos sensoriais de acordo com o padrão MPEG-V. Ainda este trabalho apresenta uma extensão do sistema de posicionamento do MPEG-V, permitindo o uso de coordenadas esféricas. Por fim, um simulador 3D de um ambiente *mulsemedia* interativo é apresentado. Como resultado, este trabalho facilita a autoria das seguintes formas. Primeiramente ao abstrair a sincronização de efeitos sensoriais. Em seguida por aprimorar o método de identificação do conteúdo de um objeto audiovisual. Por fim, permitindo uma definição mais genérica de efeitos sensoriais junto com seu posicionamento em aplicações multimídia interativas e, por meio do simulador 3D, dar suporte ao autor visualizar a execução destes efeitos.

Palavras-chave:
Autoria Multimídia; Redes Neurais; Efeitos Sensoriais; Simulador; MPEG-V; NCL.

Rio de Janeiro,

20 de julho de 2018

ABSTRACT

Interactive Multimedia Document Authoring Based on Synchronization of Sensory Effects in Relation to Audiovisual Content

Raphael Silva de Abreu

Advisor:
Joel André Ferreira do Santos

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

Advances in ubiquitous computing has been important to increase the immersion level of users in virtual environments. This breakthrough has boosted research to increase user immersion levels in multimedia applications. One of them is the addition of sensory effects, enabling the so-called *mulsemedia* applications. Such applications can stimulate other human senses beyond sight and hearing. However, the development of these applications require an authoring effort to synchronize sensorial effects with audiovisual content. In addition, interactive applications lack abstractions to facilitate the authoring of sensory effects. Therefore, this work presents an approach to facilitate the authoring and of interactive *mulsemedia* applications. To address these issues, this work focused on three work fronts. The first is the concept of abstract anchors, which allows to perform the semi-automatic synchronization of sensory effects with a media object. In this context, we present a processor for the NCL language that uses neural networks to identify when contents are presented to perform the synchronization. The second front is a bimodal neural network architecture, aimed at improving the identification of an audiovisual object content by taking into account the audio and video modalities. The third front is to allow the definition of sensory effects in declarative multimedia languages, making possible the authoring of interactive applications with sensory effects according to the MPEG-V standard. This work presents an extension of the MPEG-V positioning system, allowing the use of spherical coordinates. Finally, a 3D simulator of an interactive *mulsemedia* environment is presented. As a result, this work facilitates the authoring in the following ways. First by abstracting the synchronization of sensory effects. Next, by improving the method of identifying the content of an audiovisual object. Finally, by allowing a more generic definition of sensory effects along with its positioning in interactive multimedia applications and, through the 3D simulator, support the author to visualize the execution of these effects.

Key-words:
Multimedia Authoring; Neural Networks; Sensory Effects; Simulator; MPEG-V; NCL.

Rio de Janeiro,

20 de julho de 2018

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AAP	Abstract Anchor Processor 32, 33, 34, 35, 38, 39, 40, 44, 53, 54, 75, 77
API	Application Programming Interface 40, 41, 75
CAVE	Computer Assisted Virtual Enviroment 16
EBNF	Extended-Backus-Naur Form 10
IANA	Internet Assigned Numbers Authority 59
LSTM	Long Short-term Memory 46, 47
MPE	Media Processing Engine 9, 22, 61, 65, 66
MPEG	Moving Picture Experts Group 1
MULSEMEDIA	Multiple Sensorial Media 1, 3, 4, 6, 7, 16, 19, 20, 21, 41, 55, 75, 76
NCL	Nested Context Language 4, 5, 6, 7, 13, 22, 23, 24, 28, 29, 30, 31, 35, 37, 38, 41, 42, 44, 55, 56, 57, 58, 59, 60, 61, 62, 69, 70, 76
PTS	Presentation Time Stamp 13, 60
QOE	Quality Of Experience 1, 8, 16, 77
SEDL	Sensory Effect Description Language 9, 12, 13, 57, 58, 59, 60, 61, 65, 66, 76
SEM	Sensory Effects Metadata 9, 10, 20, 22, 23, 25, 69, 70, 71, 74
SEV	Sensory Effect Vocabulary 9, 59
TES	Tradutor De Efeitos Sensoriais 59, 60, 61, 62, 63, 64, 65, 66
UDP	User Datagram Protocol 70, 71
WYSIWYG	What You See Is What You Get 70
XML	EXtensible Markup Language 9, 10, 12, 13, 71

Capítulo I Introdução

Os avanços nas interações homem-computador oferecem diversas oportunidades para enriquecer aplicações multimídia com novos recursos [Jaimes and Sebe, 2007; Cruz-Neira et al., 1992; Oviatt, 2003]. Desde o início desta década, houve significativo interesse em tecnologias mais imersivas (*displays* 3D, VR, etc.) [Freina and Ott, 2015]. Tal interesse resultou no aumento dos esforços da comunidade multimídia para desenvolver novos métodos para melhorar a imersão dos usuários em aplicações multimídia [Sulema, 2016].

Estudos sugerem que a inclusão de estímulos a outros sentidos humanos podem proporcionar aos usuários maior imersão em aplicações multimídia [Dinh et al., 1999]. Adicionalmente, a reprodução de efeitos sensoriais sincronizados com conteúdo audiovisual pode proporcionar uma maior imersão do usuário e têm potencial de melhorar a Qualidade da Experiência (do inglês *Quality Of Experience* - QoE) [Timmerer et al., 2012].

Neste contexto, Ghinea et al. [2014] indicam que experiências com múltiplos efeitos sensoriais serão o novo desafio da área de multimídia para os próximos dez anos, cunhando o termo aplicações *mulsemedia* (*Multiple Sensorial Media*). Diferente das aplicações multimídia tradicionais que são exclusivamente bi-sensoriais (i.e., visão e audição), as aplicações *mulsemedia* são aquelas que envolvem três ou mais de nossos sentidos (e.g., visão, audição, olfato) [Ghinea et al., 2014]. Para produzir tais aplicações, pode-se usar dispositivos sensores para identificar estados do ambiente (e.g., temperatura, reação do usuário) e atuadores para gerar efeitos sensoriais (e.g., vento, névoa, calor) [ISO/IEC 23005-1, 2016].

Nesse mesmo contexto, o Moving Picture Experts Group (MPEG) cunhou o padrão MPEG-V [ISO/IEC 23005-1, 2016]. O padrão define um conjunto de metadados capazes de representar a comunicação entre o mundo real e virtual. Utilizando o padrão, aplicações podem ser anotadas com efeitos sensoriais.

I.1 Motivação

Aplicações contendo efeitos sensoriais requerem um esforço de autoria para realizar a sincronização de efeitos sensoriais com o conteúdo audiovisual. Este esforço está relacionado seja à indicação de quando um efeito deverá ser executado [Timmerer et al., 2012], quanto a caracteri-

zação de um efeito na aplicação [Josué et al., 2018].

Para indicar quando um efeito sensorial deve ser executado, o autor deve sincronizar momentos de renderização[1] de efeitos sensoriais com momentos de apresentação de "componentes de cena" em um objeto de mídia. Neste trabalho, definimos um componente de cena como segue.

Definição 1 (Componente de Cena) *Um componente de cena representa*

(i) um elemento (e.g., pessoa, rocha, fogo); ou

(ii) um conceito (e.g., frio, noite, calmo, feliz)

que faz parte do conteúdo de um objeto de mídia e é apresentado em um ou mais pontos durante sua apresentação.

Esta definição se inspira em trabalhos relacionados ao reconhecimento do conteúdo de objetos de mídia visuais [Jain et al., 1977; Kunishige et al., 2011; Cao et al., 2012]. Porém, em nossa definição, conceitos podem surgir de características dos elementos apresentados em um objeto de mídia audiovisual. Por exemplo, a cor de um objeto pode sugerir um conceito de calor [Ziat et al., 2016] ou um período do dia [Tsoumakas and Katakis, 2007]. Tais componentes de cena, podem inclusive ser reconhecidos em mais de uma modalidade de um objeto de mídia, como em seu áudio e vídeo separadamente ou em conjunto. É importante notar, que essa definição não tem relação a definição de cenas audiovisuais do padrão MPEG-4 [ISO/IEC 14496-11:2015, 2015], no qual uma cena é definida por meio de um conjunto de componentes distribuídos no dispositivo de exibição.

Para sincronizar um efeito sensorial, por exemplo, com um determinado objeto de mídia, o autor da aplicação precisa observar todo o conteúdo do objeto de mídia buscando o início e fim da apresentação dos componentes de cena de interesse para então usar tais tempos para especificar a sincronização dos efeitos sensoriais. Essa tarefa é muito custosa e pode induzir a erros [Guedes et al., 2017]. Neste contexto, Waltl et al. [2013] indicam que uma forma de gerar tais marcações de maneira semiautomática incentivaria a adoção de tais aplicações pela comunidade.

O autor de uma aplicação deve ser capaz de especificar um efeito na própria aplicação. Isto permite que a sincronização do efeito seja definida por relações condicionais da aplicação e, consequentemente, a execução de efeitos sensoriais em aplicações interativas. Aplicações multimídia interativas despertam grande interesse da comunidade e da indústria [Wu et al., 2009]. O padrão MPEG-V, apesar de não ter sido definido especificamente para esse propósito, tem sido

[1] Renderizar é uma adaptação da palavra em inglês *Render*. Que significa, em termos gerais, apresentar ou prover algo. No contexto deste trabalho, o termo renderizar é usado para significar o ato de representar efeitos sensoriais de forma física para o usuário.

utilizado para especificar a sincronização de múltiplos efeitos sensoriais com o conteúdo de uma aplicação interativa[2] [Santos et al., 2015]. Por focar na comunicação da aplicação com sensores e atuadores no ambiente real, a especificação da interatividade em aplicações definidas usando o MPEG-V fica a cargo do autor, que deve produzir código imperativo para tratar a interatividade em sua aplicação.

Uma forma de diminuir o esforço de autoria de uma aplicação com efeitos sensoriais é utilizar uma abordagem declarativa. Assim, obtém-se uma clara separação entre a definição do efeito e detalhes de sua implementação. Existem propostas para a especificação de efeitos sensoriais em linguagens declarativas [Guedes et al., 2016a, 2017], porém tal especificação necessita que o autor enderece diretamente os atuadores responsáveis por renderizar um efeito sensorial. Um ponto negativo dessa proposta é que o código declarativo fica fortemente acoplado aos atuadores usados. Assim, diminui-se a portabilidade de tais aplicações, ou seja, que uma aplicação possa ser usada em outros ambientes sem que seja necessária uma alteração no seu código.

Uma maneira de atingir essa portabilidade é usar o padrão MPEG-V como base para execução de efeitos sensoriais nos dispositivos atuadores disponíveis no ambiente. O MPEG-V utiliza uma forma de ativação de atuadores baseado no seu posicionamento no ambiente de execução, sem endereçar especificamente qual atuador que será executado. Ainda assim, o método de posicionamento do MPEG-V pode se tornar um fator que dificultante da autoria por não permitir um controle preciso destas posições. Dificultando, por exemplo, representar pequenas variações na posição de um efeito ou até mesmo animações da mudança de posição de um efeito no ambiente.

Por fim, uma forma de apoiar o processo de autoria de conteúdo *mulsemedia* é utilizar ferramentas de autoria. Tais ferramentas podem permitir uma rápida prototipação da aplicação sendo criada, como é o caso com simuladores [Waltl et al., 2013]. No contexto de *mulsemedia*, entretanto, é importante que tal ferramenta dê suporte a execução de aplicações interativas e contendo efeitos sensoriais.

I.2 Objetivos

Diante do grande interesse pelo aumento da imersão, em particular pela inclusão de efeitos sensoriais, em aplicações multimídia e a dificuldade inerente do modelo de desenvolvimento sendo usado para criar tais aplicações, o principal objetivo desta dissertação é facilitar a autoria de efeitos sensoriais em aplicações multimídia interativas.

Uma abordagem para a autoria de efeitos sensoriais em aplicações interativas é reutilizar linguagens de autoria multimídia tradicionais, estendendo-as para este propósito. Dessa forma se ganha toda a capacidade de definição da sincronização de tais linguagens. Em particular, este

[2] É importante notar que tal sincronização tem sempre uma única mídia como referência.

trabalho foca na linguagem *Nested Context Language* (NCL) [ITU, 2009]. NCL pode facilitar o desenvolvimento de aplicações *mulsemedia* ao abstrair a sincronização de objetos de mídia, permitindo que os autores utilizem eventos e estruturas causais para especificar relações temporais entre tais objetos. Desta forma, em NCL é possível sincronizar efeitos sensoriais com conteúdo audiovisual ao criar relacionamentos de causa-e-efeito entre eventos.

Em NCL, uma abordagem usual para definir a sincronização entre objetos de mídia é criar âncoras temporais, indicando partes do objeto que são de interesse para a sincronização da aplicação e relacionamentos que associam a ocorrência de eventos dessas âncoras a eventos em outros objetos de mídia na aplicação. Assim, o início e o fim do evento de apresentação dessas âncoras, por exemplo, são utilizados para disparar relacionamentos temporais definidos na aplicação. Neste trabalho propomos o conceito de âncoras abstratas, cujo objetivo é aprimorar o processo de autoria de aplicações *mulsemedia* ao permitir a definição da sincronização de efeitos sensoriais com o conteúdo de mídias da aplicação. A partir de tal definição é feita a geração automática da sincronização por um software de reconhecimento que verifica dentro de cada mídia os momentos de apresentação dos componentes de cena relacionados a efeitos sensoriais.

Redes neurais podem ser utilizadas para realizar o reconhecimento de componentes cena relacionados a efeitos sensoriais. Porém, uma característica não explorada é o reconhecimento de efeitos sensoriais em ambas as modalidades de áudio e vídeo em conjunto. Neste trabalho, é proposta uma arquitetura de rede neural que realiza o reconhecimento de componentes de cena em ambas a modalidades conjuntamente.

NCL tem a vantagem de não levar em consideração o tipo de mídia que ela representa. As mídias em NCL podem ser do tipo áudio, vídeo, imagem, *scripts*, e mídias customizadas. De forma a permitir a especificação de efeitos sensoriais em aplicações NCL, este trabalho propõe ainda uma representação de tais efeitos similar à de objetos de mídia convencionais da aplicação. Assim, toda a capacidade de representação da sincronização e da interatividade de NCL pode ser reusada para a especificação de aplicações *mulsemedia*. A representação de um efeito sensorial utiliza uma extensão do método de posicionamento do MPEG-V para permitir a definição de coordenadas esféricas. Tais coordenadas permitem uma representação de posicionamento de efeitos sensoriais com maior precisão. Neste trabalho é apresentado um módulo que traduz tais descrições em NCL para descrições compatíveis no padrão MPEG-V e as envia pela rede.

Além de permitir a descrição de aplicações com efeitos sensoriais em linguagens multimídia interativas, este trabalho propõe um simulador 3D que permite visualização da ativação de efeitos sensoriais. O simulador recebe comandos de ativação pela rede e, desta forma, permite a visualização da execução de efeitos sensoriais integrada com a reprodução de aplicações interativas.

I.3 Contribuições

Esta seção apresenta as principais contribuições desse trabalho. As contribuições são divididas em três frentes, são elas:

1. Contribuições relacionadas a autoria semiautomática da sincronização baseada no conteúdo de mídia. Esta frente apresenta as seguintes contribuições:

   (a) A definição do conceito de âncoras abstratas. Uma âncora abstrata permite definir a sincronização de uma aplicação baseada em componentes de cena presentes no conteúdo de um objeto de mídia audiovisual. Ou seja, âncoras abstratas oferecem uma forma de descrever e sincronizar o conteúdo que não se encontra no documento, mas sim dentro de um objeto de mídia em si. Âncoras abstratas facilitam a autoria de aplicações com efeitos sensoriais por evitar que o autor precise identificar tempos de exibição de tais efeitos manualmente. Este processo é feito automaticamente de acordo com a identificação de componentes de cena em um objeto de mídia.

   (b) A extensão da linguagem de autoria multimídia NCL de forma a permitir definição de âncoras abstratas. Esta extensão permite integrar âncoras abstratas com as definições de relacionamentos em NCL para reprodução de efeitos sensoriais em uma aplicação multimídia.

   (c) A criação de um processador de âncoras abstratas (do inglês *Abstract Anchor Processor* - AAP), tendo NCL como linguagem alvo. A ferramenta reconhece componentes de cena no conteúdo dos objetos de mídia de uma aplicação multimídia e realiza a instanciação das âncoras abstratas e dos relacionamentos associados a essas âncoras para definir a sincronização da aplicação como um todo. Para isto, o AAP faz interface com softwares de reconhecimento baseados em redes neurais. A ferramenta foi desenvolvida de forma a ser extensível e permitir a fácil integração com diversos softwares de reconhecimento.

2. Contribuições relacionadas ao aprimoramento do reconhecimento de componentes de cena em conteúdo audiovisual. Esta frente apresenta as seguintes contribuições:

   (a) Proposta de uma arquitetura de rede neural bimodal como uma forma de otimizar o reconhecimento de componentes de cena relacionados a efeitos sensoriais. Tal proposta baseia-se na concepção de que as modalidades de áudio e vídeo podem ser usadas simultaneamente para identificar os momentos em que algum componente de cena é apresentado em uma mídia audiovisual. Resultados indicam que a arquitetura de rede

neural bimodal aprimora o reconhecimento de componentes de cenas relacionados a efeitos sensoriais.

(b) Criação de um conjunto de dados bimodal. Tal conjunto foi criado para avaliação da arquitetura proposta. Assim é apresentado um método de construção desse conjunto com base no conjunto AudioSet do Google.

3. Contribuições relacionadas a representação de efeitos sensoriais em aplicações interativas. Esta frente apresenta as seguintes contribuições:

   (a) Definição de um módulo de reprodução de efeitos sensoriais em NCL. O módulo de reprodução permite a criação de aplicações *mulsemedia* apoiando-se nas abstrações conceituais da linguagem NCL. O módulo, implementado em Lua [Ierusalimschy, 2006], é capaz de interpretar eventos gerados pelo interpretador da linguagem NCL e traduzi-los em descrições de comandos de efeitos sensoriais compatíveis com o padrão MPEG-V. Os comandos gerados são enviados pela rede local para que reprodutores compatíveis com o padrão MPEG-V possam recebê-los e renderizar os efeitos descritos. O módulo facilita a autoria e reprodução de uma aplicação *mulsemedia* interativa ao integrar suas definições em uma linguagem multimídia orientada a eventos.

   (b) Extensão do método de posicionamento de efeitos sensoriais do padrão MPEG-V. Tal extensão foi proposta visando aprimorar e tornar mais granular a definição da posição em que um efeito é executado. Esta extensão permite o uso de coordenadas esféricas, além das coordenadas tradicionais do MPEG-V, para definir a localização da ativação de efeitos sensoriais. Assim é permitindo ao autor um maior controle sobre o posicionamento de um efeito.

   (c) Definição de um simulador 3D de efeitos sensoriais. O simulador permite a fácil prototipação de uma aplicação contendo efeitos sensoriais. O simulador é capaz de representar uma sala tridimensional com atuadores de efeitos sensoriais e ativá-los de acordo com descrições do padrão MPEG-V. O simulador é capaz de apresentar o posicionamento de efeitos seguindo a extensão do método de posicionamento MPEG-V proposta neste trabalho.

I.4 Estrutura

O restante deste trabalho está estruturado da seguinte forma. O Capítulo II fornece informações básicas necessárias para entendimento deste trabalho e trabalhos relacionados. No capítulo, é apresentada uma descrição do padrão MPEG-V, com enfoque na Parte 3 “Informação

Sensorial". Também são apresentadas linguagens de autoria multimídia e o conceito de templates para estas linguagens. São apresentados modelos de sincronização multimídia e por fim uma breve discussão sobre a Qualidade da Experiência em aplicações *mulsemedia*.

No Capítulo III são apresentados trabalhos relacionados. Nele são apresentados trabalhos cujo foco seja tanto a criação quanto a execução de aplicações *mulsemedia*. São apresentados trabalhos de geração automática da sincronização de efeitos sensoriais, bem como métodos baseados em redes neurais para realizar reconhecimento de cena. O capítulo também apresenta softwares compatíveis com o padrão MPEG-V para criação, reprodução e simulação de aplicações com efeitos sensoriais. Este capítulo também apresenta esforços no sentido de permitir interatividade em aplicações com efeitos sensoriais.

No Capítulo IV é apresentado o conceito de âncoras abstratas como forma de permitir a autoria semiautomática da sincronização baseada no conteúdo de mídia. No capítulo é formalizada a definição de uma âncora abstrata e apresentada sua instanciação para a linguagem NCL com o Processador de Âncoras Abstratas. No capítulo também é mostrado o uso de uma rede neural para identificação de componentes de cena em vídeo. Por fim, um caso de uso é apresentado para ilustrar como o uso de âncoras abstratas facilita a autoria de uma aplicação multimídia com efeitos sensoriais.

No capítulo V é apresentada uma arquitetura de rede neural bimodal, buscando melhorar a capacidade de reconhecimento de componentes de cena. São descritas as formas de extração de características das modalidades de áudio e de vídeo e sua combinação. Para fins de avaliação, apresentamos um método de construção de um conjunto de dados bimodal com base no conjunto AudioSet do Google. São apresentados o método de extração de exemplos de treinamento do conjunto de dados, bem como os experimentos realizados para validar nossa arquitetura bimodal.

No Capítulo VI é apresentado um módulo acoplável a documentos NCL que permite a reprodução de efeitos sensoriais. É apresentada a sintaxe de uma descrição de efeito sensorial em NCL usando esse módulo e a geração da descrição equivalente para código compatível com o padrão MPEG-V. Neste capítulo também é apresentada a extensão do método de posicionamento do padrão MPEG-V para utilizar coordenadas esféricas e um simulador 3D para efeitos sensoriais.

Por fim, o Capítulo VII conclui este trabalho, apresentando contribuições, limitações e trabalhos futuros.

Capítulo II Conceitos básicos

Este capítulo apresenta alguns conceitos básicos relativos ao trabalho aqui proposto. O objetivo é permitir ao leitor uma maior compreensão a respeito do padrão MPEG-V (Seção II.1) e de modelos e linguagens tradicionais para a criação de aplicações multimídia (Seção II.2). Uma breve discussão sobre *templates* em linguagens multimídia (Seção II.3) e QoE em *mulsemedia* (Seção II.4) encerram este capítulo.

II.1 MPEG-V

O *Moving Picture Experts Group* (MPEG) produz diversos padrões para tecnologias relacionadas à compressão e transmissão de áudio e vídeo. Motivados pelo avanço de tecnologias ubíquas e por sua integração com conteúdo multimídia, nos últimos anos, o padrão MPEG-V [ISO/IEC 23005-1, 2016] foi estabelecido. Ele consiste de sete partes que são apresentadas na Tabela II.1. O padrão define metadados para a representação de diferentes tipos de comunicação entre mundos virtuais e o mundo real.

Tabela II.1: Partes do padrão MPEG-V com suas respectivas descrições

	Descrição
Parte 1	Apresenta a arquitetura do MPEG-V e descreve relações entre suas partes
Parte 2	Define uma sintaxe de controle de dispositivos, tais como capacidade de dispositivos e preferência do usuário
Parte 3	Especifica sintaxe e semântica para descrição de informações de efeitos sensoriais
Parte 4	Especifica uma representação de objetos virtuais (*avatars*) assim como a representação virtual de objetos físicos
Parte 5	Define uma sintaxe para descrever interações de dispositivos, comandos e informações obtidas de sensores
Parte 6	Define padrões de sintaxe e semântica, tipos de dados e ferramentas para as outras partes do MPEG-V
Parte 7	Provê softwares de referência e conformidade

Os mundos virtuais do MPEG-V podem ser jogos como *Second Life* e *World of Warcraft*, filmes ou simulações. O mundo real pode ser representado por sensores, atuadores e dados integrados ao mundo real (e.g., características físicas, preferências, etc.). O padrão também abrange especificações sobre dispositivos, características de objetos virtuais e formatos padrão para interação entre dispositivos.

Além disso, o padrão fornece meios para enriquecer conteúdo multimídia de forma a estimular sentidos adicionais do usuário, tais como olfato, tato e paladar. A finalidade desta padronização é mapear efeitos sensoriais que por sua vez são renderizados em dispositivos compatíveis (e.g., cadeiras de vibração, lâmpadas, ventiladores). Esse trabalho é baseado nesta parte da padronização, sendo assim, o restante desta seção se concentra nela.

II.1.1 A Parte 3 do MPEG-V

A parte 3 do padrão MPEG-V provê a linguagem de descrição *Sensory Effect Description Language* (SEDL), uma linguagem baseada em eXtensible Markup Language (XML) que permite descrever as características de efeitos sensoriais.

Estes efeitos são descritos por um vocabulário de descrição chamado *Sensory Effect Vocabulary* (SEV), que define uma sintaxe e semântica de cada tipo de efeito sensorial (e.g., vento, vibração e névoa) e seus respectivos atributos (e.g., velocidade e intensidade). Esta abordagem permite que o vocabulário seja estendido e oferece flexibilidade para que cada domínio de aplicação possa definir seus próprios efeitos sensoriais [ISO/IEC 23005-3, 2016]. De acordo com Timmerer et al. [2012], o vocabulário SEV define os efeitos de: luz (*LightType*), flash (*FlashType*), temperatura (*TemperatureType*), vento (*WindType*), vibração (*VibrationType*), spray de água, (*SprayingType*), névoa (*FogType*), aroma (*ScentType*) e correção de cor (*ColorCorrectionType*). Por fim são descritos um conjunto de efeitos de movimento que podem ser usados para dispositivos táteis: *RigidBodyMotionType*, *PassiveKinestheticMotionType*, *PassiveKinestheticForceType*, *ActiveKinestheticType* e *TactileType*.

Um arquivo de descrição em conformidade com o SEDL é chamado de *Sensory Effects Metadata* (SEM). Este arquivo é utilizado para gerir a reprodução de efeitos sensoriais em dispositivos compatíveis. A Figura II.1 ilustra este cenário.

Na figura, o provedor de conteúdo (*Source*) envia até o *Media Processing Engine* (MPE) o conteúdo de mídia em conjunto com um arquivo SEM. O MPE é responsável por reproduzir conteúdo de mídia e renderizar efeitos sensoriais descritos em arquivos SEM. Esta reprodução é feita para o consumidor (*User*) de maneira sincronizada com a mídia audiovisual. O efeito a ser reproduzido depende da capacidade dos dispositivos sensoriais disponíveis para o usuário, que é definida de acordo com a parte 5 do padrão MPEG-V.

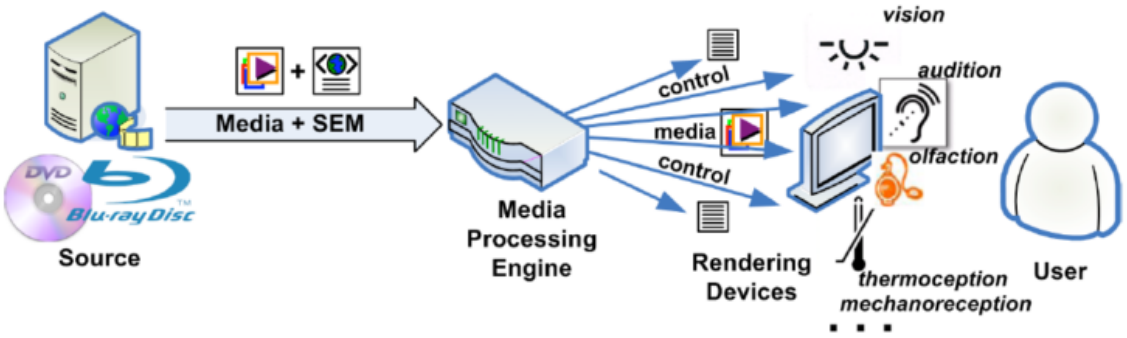

Figura II.1: Visão geral de sistemas de experiências sensoriais baseado em MPEG-V. Fonte: [Waltl et al., 2009]

O elemento `SEM` é o elemento raiz de um arquivo SEM. Ao longo dessa seção, tanto os atributos quanto demais elementos de um arquivo SEM são apresentados em uma notação baseada na notação *Extended-Backus-Naur Form* (EBNF). Nesta notação, elementos XML são representados entre os símbolos $\langle$ e $\rangle$, enquanto atributos são representados fora desses símbolos. Aqui apresentamos a notação adaptada de acordo com os trabalhos de Timmerer et al. [2012] e Ghinea et al. [2014], assim como a norma MPEG-V [ISO/IEC 23005-1, 2016].

$$\langle SEM \rangle ::= timeScale\ [autoExtraction]\ \langle DescriptionMetadata \rangle$$
$$(\langle Declarations \rangle \mid \langle GroupOfEffects \rangle \mid \langle Effect \rangle \mid \langle ReferenceEffect \rangle)^{+}$$

O elemento `SEM` contém um atributo `timeScale` que define a escala de tempo, ou seja, o número de instantes por segundo, utilizada para os efeitos sensoriais dentro dessa descrição. O atributo opcional `autoExtraction` é usado para sinalizar que a extração automática de efeitos sensoriais é preferível[1]. Esta extração automática pode ser por áudio, vídeo ou ambos. O elemento pode ainda conter a descrição de metadados sobre o arquivo através do elemento filho `DescriptionMetadata`. Por fim, para a declaração de efeitos sensoriais temos os elementos filhos de `SEM`: `Declarations`, `GroupOfEffects`, `Effect` e `ReferenceEffect`.

O elemento `ReferenceEffect` permite referenciar um efeito criado anteriormente ou um grupo de efeitos.

$$\langle ReferenceEffect \rangle ::= [id]\ uri\ BaseAttributes$$

Ele define um identificador (atributo `id`) opcional, a `uri` do elemento referenciado e, possivelmente, atributos do conjunto de atributos `BaseAttributes` discutido adiante nessa seção.

[1] Importante notar que o método para realizar esta extração está fora do escopo do padrão MPEG-V

O elemento `Declarations` é definido como segue.

$$\langle Declarations \rangle \ ::= \ (\langle GroupOfEffects \rangle \mid \langle Effect \rangle \mid \langle Parameter \rangle)^{+}$$

O elemento `Declarations` é usado para definir configurações comuns usadas pelos vários efeitos sensoriais e deve conter pelo menos uma das opções entre `GroupOfEffects`, `Effect` e `Parameter`. O elemento `Parameter` pode ser usado para definir configurações comuns usadas por vários efeitos sensoriais no mesmo ambiente.

O elemento `GroupOfEffects` define um grupo de efeitos como segue.

$$\langle GroupOfEffects \rangle \ ::= \ timestamp \ BaseAttributes \ (\langle EffectDefinition \rangle \mid \langle ReferenceEffect \rangle)^{2..n}$$

Ele define como atributo uma marca de tempo (*timestamp*) que indica quando o grupo de efeitos estará disponível para a aplicação. A marca de tempo poderá ser usada para fins de sincronização com a mídia associada. Ainda, um grupo de efeitos pode definir atributos do conjunto `BaseAttributes` e deve conter pelo menos duas definições de `EffectDefinition` ou `ReferenceEffect`.

O elemento `Effect` é usado para descrever um efeito sensorial como segue.

$$\langle Effect \rangle \ ::= \ id \ timestamp \ EffectDefinition$$

O elemento `Effect` é usado para descrever um único efeito sensorial (e.g., vento, calor) em uma determinada marca de tempo (`timestamp`). Ele define um identificador e uma marca de tempo que indica o tempo de início do efeito. Um elemento effect é composto de definições de efeito descritas no elemento `EffectDefinition`, que é definido como segue.

$$\langle EffectDefinition \rangle \ ::= \ BaseAttriutes \ \langle SupplementalInformation \rangle$$

O elemento `EffectDefinition` pode possuir o elemento `SupplementalInformation` que é usado para definir uma região para qual as informações de efeito podem ser extraídas caso a auto extração de efeitos (`autoExtraction`) esteja habilitada. São definidos atributos do conjunto

de atributos `BaseAttributes`, que compreende todas as informações relativas ao efeito sensorial.

$$BaseAttributes ::= [activate]\ [duration]\ [intensity - value]\ [intensity - range]\ [fade]\ [priority]\ [location]\ [alt]\ [adaptability]\ [autoExtraction]$$

Os atributos do conjunto possuem a seguinte semântica.

- `activate` descreve se o efeito deve ser ativado (*true* ou *false*).
- `duration` define a duração do efeito em segundos.
- `intensity-range` define uma escala entre os valores de intensidade mínima e máxima que o efeito pode ser executado.
- `intensity-value` define a intensidade do efeito de acordo com uma escala definida em `intensity-range`.
- `fade` Realiza aumento/diminuição gradual do efeito até intensidade alvo (`intensity-value`).
- `alt` descreve um efeito alterativo caso o original não possa ser processado.
- `priority` diz respeito a prioridade entre os efeitos de um mesmo grupo.
- `location` são termos nos eixos XYZ que definem um ponto ou local no espaço tridimensional onde o efeito será renderizado. O posicionamento é definido por uma concatenação de textos. Este atributo será descrito em mais detalhes na Seção VI.4.
- `adaptability` define tipos preferidos de adaptação do efeito correspondente.
- `autoExtraction` descreve se o efeito atual deve ser extraído automaticamente da mídia.

A Listagem II.1 apresenta um exemplo em XML da descrição de efeitos sensoriais por SEDL. Por brevidade, os prefixos indicativos de espaço de nomes (*namespace*) foram omitidos.

```
1 <Effect type="FlashType" intensity−value="20000.0" intensity−range="0.00001 32000.0"
    duration="5" frequency="10" pts="0"/>
```

Listagem II.1: Definição de efeito de luz de acordo com SEDL

O exemplo da Listagem II.1 mostra a descrição de um efeito de flash. A intensidade é de $20.000 lux$ (dentro de uma faixa de $10^{-5}\ lux$ e $32.000 lux$), com uma duração de 5 segundos. A luz cintila a uma frequência de 10 vezes por segundo. Note que o atributo `frequency` é específico de efeitos de luz, pois o SEDL pode ser estendido com atributos específicos a cada tipo de

efeito sensorial. Nesta descrição, o `timeStamp` do efeito é descrito como PTS (*Presentation Time Stamp*). O efeito começa no tempo 0 (`si:pts="0"`). PTS são unidades de tempo relacionadas a uma unidade de tempo padrão definida no programa. Neste caso o padrão MPEG-V utiliza o atributo *timescale* para definir uma unidade padrão de pts que representa o tempo de 1 segundo de execução. Em nosso trabalho 1 segundo é representado por $9.000pts$, seguindo o valor padrão da norma MPEG-V [ISO/IEC 23005-1, 2016].

Na Listagem II.2 são definidos dois efeitos de vento. O primeiro efeito inicia no tempo $9.000pts$ e é localizado no ponto `left:middle:front`, ou seja, no atuador que esteja na esquerda, região central e frente do ambiente. O segundo efeito possui o atributo `activate` igual a *false* e tempo $27.000pts$ com a localização `left:middle:front`. Esta segunda linha desativa os efeitos de vento de atuadores que estejam na posição descrita. Então de acordo com estas duas linhas de descrição, um efeito de vento ficará ativo entre os instantes $9.000pts$ e $27.000pts$.

```
1 <Effect type="WindType" activate="true" location="left:middle:front" intensity−value="3.0"  pts="
      9000"/>
2 <Effect type="WindType" activate="false" location="left:middle:front" pts="27000"/>
```

Listagem II.2: Definição de efeitos de vento de acordo com SEDL

Esta seção focou na apresentação do padrão MPEG-V e da linguagem SEDL. A seção a seguir apresenta o cenário existente das linguagens de autoria multimídia.

II.2 Linguagens de Autoria Multimídia

Linguagens de autoria multimídia são linguagens específicas de domínio que proveem facilidades para desenvolver aplicações multimídia. Estas linguagens geralmente especificam aplicações multimídia focando na definição da sincronização de objetos de mídia, independentemente do conteúdo deles. Exemplos de linguagens de autoria multimídia são SMIL (*Synchronized Multimedia Integration Language*) [W3C, 2008], NCL (*Nested Context Language*) [ITU, 2009] e HTML5 (*Hypertext Markup Language*) [W3C, 2017].

Estas linguagens são geralmente declarativas e baseadas em XML. As vantagens destes formatos são que eles são legíveis por humanos, fáceis de estender, analisar e gerar. Tais características são desejáveis para um autor de aplicações multimídia.

A forma como uma linguagem especifica a sincronização entre os objetos de mídia de uma aplicação depende do modelo de sincronização na qual ela é baseada. Neste trabalho, utilizamos a definição baseada nos trabalhos de Bulterman and Hardman [2005] e Hardman et al. [2000] que classificam os modelos de sincronização de documentos multimídia em relação a como eles especificam relações temporais entre objetos de mídia. As seções a seguir descrevem as principais características dos modelos de sincronização baseados em linha do tempo (*timeline-based*) e baseados em eventos (*event-based*).

II.2.1 Sincronização Baseada em Linha do Tempo

O conceito básico de uma linha do tempo é projetar eventos de sincronização temporal (tal como início e fim de mídia) em um eixo temporal. Usualmente este eixo é representado graficamente por uma linha horizontal em que o tempo é representado no eixo x e as mídias são distribuídas ao longo deste eixo. Caso haja uma sobreposição de eixos temporais entre 2 ou mais mídias, é comum distribuir as mídias ao longo do eixo y. É importante notar que a distribuição no eixo y não possui semântica associada, servindo apenas para facilitar a visualização dos objetos de mídia que compõem uma aplicação. Para tal paradigma, é comum o uso de uma escala temporal global como uma referência de ativação comum para todos os objetos de mídia.

Este modelo de sincronização é melhor adaptado para autoria de documentos de mídia com início e fim, de tal forma que diversas mídias possam ser executadas dentro de um período de tempo estipulado. A Figura II.2 representa graficamente este modelo de sincronização. Na Figura existem duas mídias, *mídia 1* e *mídia 2*. As mídias são sincronizadas em uma linha do tempo identificada por t. Note que os tempos de início e fim das mídias devem ser definidos sobre essa linha do tempo.

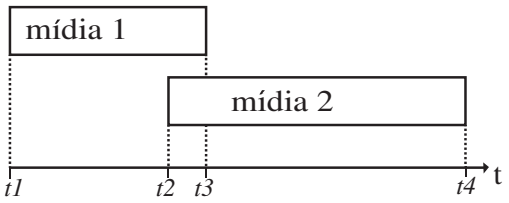

Figura II.2: Ilustração de modelo de sincronização baseado em linha do tempo

O paradigma baseado em linha do tempo tem algumas limitações. Segundo Bulterman and Hardman [2005], nesse paradigma, características importantes para um sistema de interação com o usuário, tal como a ativação condicional ou o comportamento assíncrono são quase impossíveis de modelar. Ainda, as condições de sincronização não podem ser expressas diretamente entre os objetos de mídia ou suas partes. Como resultado, se as durações dos objetos de mídia forem alteradas, o autor deve refazer o processo de definição de sincronização temporal entre as mídias. Um exemplo desta tarefa seria quando, por exemplo, uma sequência de vídeo é encurtada e os efeitos sensoriais anotados previamente devem ser novamente sincronizados com as partes do vídeo.

II.2.2 Sincronização Baseada em Eventos

A sincronização baseada em eventos permite a criação de relacionamentos entre os eventos que ocorrem durante a execução do documento. Uma aplicação é descrita por um grafo em que vértices representam objetos de mídia (também chamados de nós) e arestas representam

relacionamentos de causa-e-efeito entre estes objetos de mídia (também chamados de elos). A criação de uma aplicação orientada a eventos é semelhante a utilização de uma linguagem de programação procedural, porém existem linguagens declarativas que permitem a especificação de aplicações multimídia baseadas em eventos, tal como NCL[ITU, 2009] e SMIL[W3C, 2008].

De acordo com Hardman et al. [2000], a sincronização baseada em eventos permite a utilização de elementos com tempo indefinido. Estes elementos são definidos no documento, porém seus momentos de início e fim são determinados por algum evento em tempo de execução. Estes eventos podem estar relacionados a um determinado conteúdo (início e fim de âncoras) ou alguma forma de interação do usuário (como clicar em um botão).

A Figura II.3 representa graficamente este modelo de sincronização. Na Figura existem duas mídias. Ao término da *mídia 1* (evento *onEnd*), a *mídia 2* deverá ser iniciada (evento *Start*). Note que este evento não tem um tempo determinado. O tempo de início da apresentação da *mídia 2* depende da duração intrínseca da *mídia 1* ou de algum evento externo que leve *mídia 1* a parar.

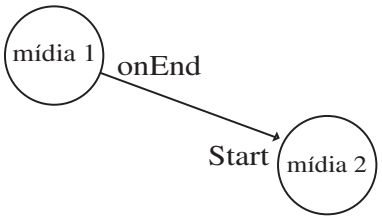

Figura II.3: Ilustração de modelo de sincronização baseado em eventos

Conforme indicado em [Blakowski and Steinmetz, 1996], esse tipo de especificação é facilmente estendido para novos tipos de sincronização. Uma extensão é o uso de eventos com durações, como em [Soares et al., 2005].

II.3 *Templates*

Linguagens de autoria multimídia permitem a criação de aplicações multimídia identificando o conteúdo da aplicação, i.e., os objetos de mídia, e relacionamentos entre eles. Apesar de simples, a definição de uma aplicação usando tais linguagens pode se tornar trabalhosa, especialmente quando a aplicação possui muitos objetos de mídia e relacionamentos.

Uma linguagem de *template* permite ao autor especificar componentes reutilizáveis (*placeholders*) que devem ser posteriormente substituídos por instâncias na linguagem alvo. Em linguagens multimídia, mais precisamente, os *templates* definem componentes genéricos e relações entre estes componentes e, possivelmente, mídias da aplicação. As relações que consideram componentes genéricos não identificam, portanto, os reais participantes da relação [Muchaluat-Saade and Soares, 2002]. Os componentes genéricos de uma aplicação usando *templates* são posteriormente instanciados para objetos de mídia ou outros componentes da linguagem alvo por

um processador de *templates*. Assim, o documento final pode ser executado em uma implementação padrão do formatador da linguagem alvo.

Em documentos multimídia, *templates* são muito utilizados para reduzir os esforço de autoria através do reuso [Santos and Muchaluat-Saade, 2012]. Entretanto, outro uso interessante de *templates* é enriquecer a linguagem alvo com semântica adicional. Por exemplo, algumas linguagens de *template* não só oferecem suporte aos *placeholders*, mas também loops e condições que geralmente faltam em linguagens multimídia declarativas [Bezerra et al., 2012].

II.4 QoE em *mulsemedia*

Qualidade de experiência (QoE) é uma avaliação da satisfação geral do usuário. A recomendação P.10/G.100 [ITU, 2008] define QoE como: "A aceitação global de uma aplicação ou serviço, como percebido subjetivamente pelo usuário final".

Uma aplicação *mulsemedia* deve preservar a QoE de forma que um efeito não seja renderizado em um momento inadequado e prejudique a experiência do usuário [Ghinea et al., 2014]. Portanto, os efeitos sensoriais (associados a conteúdo audiovisual) precisam ser renderizados por seus atuadores de forma apropriada e sincronizada com o conteúdo do objeto de mídia sendo apresentado.

Em pesquisa sobre sincronização de vídeo com efeitos hápticos (e.g vibração) e de vento, Yuan et al. [2014] concluiu que efeitos hápticos podem ser apresentados com atraso de $1,0s$ em relação a um vídeo e efeitos de vento podem ser apresentados tanto $3,0s$ antes até $5,0s$ após o conteúdo de vídeo para serem considerados aceitáveis, i.e., em sincronia, para a maioria dos usuários. Hülsmann et al. [2014] avaliou em um grupo de usuários o tempo de percepção e reação a efeitos de vento e calor em um sistema CAVE (*Computer Assisted Virtual Enviroment*) e reportou, para efeitos de vento tempos de reação de $\approx 3,1s$ para percepção da ativação do efeito e $\approx 1,3s$ para percepção da desativação. Para os efeitos de calor, reportou tempos de reação de $\approx 2,4s$ para percepção da ativação do efeito e $\approx 2,0s$ para percepção da interrupção do efeito.

Em outra pesquisa envolvendo sincronização de efeitos sensoriais, Ghinea and Ademoye [2010] apresentam limites temporais para reprodução de efeitos de aroma. No estudo os autores concluem que os atrasos para ativação/desativação de um efeito de aroma devem estar dentro de um intervalo temporal de $30s$ antes até $20s$ após o conteúdo do vídeo para serem considerados em sincronia.

Capítulo III Trabalhos Relacionados

A literatura é rica em trabalhos que apresentam abordagens para permitir o uso de efeitos sensoriais em aplicações multimídia. Estas abordagens podem ser diversas, portanto, separamos os trabalhos relacionados de acordo com a seguinte classificação: aqueles que focam na geração automática da sincronização de efeitos sensoriais com objetos de mídia; aqueles que utilizam redes neurais para reconhecimento de componentes de cena; aqueles que propõem ferramentas de autoria para aplicações com efeitos sensoriais; aqueles cujo objetivo é executar efeitos sensoriais em aplicações multimídia, seja em estrita sincronia com conteúdo audiovisual ou de forma a permitir interatividade com usuário; aqueles que implementam simuladores de efeitos sensoriais. As seções a seguir apresentam os trabalhos relacionados de acordo com essa classificação.

III.1 Geração Automática da Sincronização de Efeitos Sensoriais

A geração automática da sincronização de efeitos sensoriais é indispensável para a adoção de tais efeitos em aplicações multimídia [Timmerer et al., 2012]. O processo de autoria manual é custoso tanto em tempo como em esforço de autoria. Nesta direção, esforços têm sido feitos para desenvolver métodos de identificação de conteúdo audiovisual e definição dos momentos de execução de efeitos sensoriais, i.e., sua sincronização.

O padrão MPEG-V [ISO/IEC 23005-1, 2016] define, em seu formato de metadados de efeitos sensoriais, a extração automática de efeitos por meio do atributo `autoExtraction`. Este atributo indica se a extração é preferível para áudio, vídeo ou ambos. Embora suportado na descrição de um efeito no padrão MPEG-V, tal facilidade depende da implementação de softwares capazes de realizar essa extração automática.

Considerando isto, Waltl et al. [2009] descrevem meios para a extração automática de informações de cor a partir do conteúdo de quadros de vídeo. A informação extraída é utilizada para realizar a ativação de atuadores de luz em um quarto. Dois métodos de cálculo de cor são apresentados. (i) cálculo da média de cores no quadro, neste método são somados os valores de cores de cada *pixel* e no final dividido número pelo total de *pixels* no quadro. (ii) cálculo da cor dominante no quadro, neste método é calculada a cor dominante no quadro i.e., a cor que mais aparece em *pixels* no quadro. Em ambos os métodos, o cálculo de cores é feito a cada 100

milissegundos. A cor resultante dos cálculos é utilizada para definir a cor a ser utilizada pelos atuadores. Este método é, em um trabalho posterior, integrado a uma ferramenta de execução de efeitos sensoriais compatível com o padrão MPEG-V [Waltl et al., 2013]. Na ferramenta, divide-se o quadro do vídeo verticalmente em três partes iguais. Para cada parte, a média das cores é calculada e o resultado é renderizado no atuador de luz na posição correspondente. Ou seja, a parte esquerda do quadro é renderizada nos atuadores da esquerda do *display*, a parte central nos atuadores acima/abaixo do *display* e a parte direita nos atuadores à direita do *display*.

Kim et al. [2014] propõem um método para executar efeitos de temperatura automaticamente, também compatível com o padrão MPEG-V. O método consiste de estimar a temperatura através da cor de uma cena e mapear suas propriedades (e.g., intensidade) para os atributos de um efeito de temperatura. Para isto o algoritmo identifica as diferentes cenas do vídeo através de mudanças bruscas na cor dos quadros do vídeo. Para cada cena é calculada uma temperatura média considerando todos os quadros da cena. O número de quadros da cena é convertido na duração do efeito de temperatura e a temperatura média da cor é convertida para a sua intensidade.

Lee et al. [2016] apresentam algoritmos que analisam quadros de vídeo e calculam o movimento relativo da câmera. Em seguida, estes movimentos são mapeados para comandos de movimento em cadeiras capazes de renderizar efeitos de movimentação e vibração. São apresentados algoritmos para 3 classes de movimentos. Movimentos rápidos de câmera (`CF`), movimentos lentos de câmera (`CS`) e vibração (`V`). Os algoritmos para efeitos `CF` e `CS` usam o vídeo como fonte. Eles incluem um método de estimativa de movimento da câmera baseado em restrições entre dois quadros de vídeo. O algoritmo de síntese para efeitos `V` usa o som como fonte. São detectados efeitos `V` analisando as características perceptivas de um som, como sua intensidade.

Shin et al. [2016] apresentam a ferramenta *Real4DAExtractor*. A ferramenta foi desenvolvida para ser acoplada a uma ferramenta de autoria de aplicações com efeitos sensoriais, sendo compatível com o padrão MPEG-V. A ferramenta realiza a extração de movimentos hápticos, de luz e flash através de técnicas de análise dos quadros de um vídeo. As propriedades de cor de efeitos de luz e flash também podem ser extraídas. Além disso, esta ferramenta fornece as funções de análise de posicionamento de objetos específicos em quadros de vídeo, usando a técnica de reconhecimento de padrões nos quadros do vídeo.

É importante notar que, assim como identificado por Timmerer et al. [2012], a autoria completamente automática de efeitos sensoriais pode ser algo indesejável. Afinal, este é um processo artístico que depende da preferência de autores humanos. Por este motivo, o método proposto neste trabalho permite a autoria semiautomática destes efeitos. Dessa forma, o autor da aplicação pode decidir quais componentes de cena e efeitos sensoriais ele deseja utilizar na aplicação. Além disso, após o reconhecimento automático de efeitos, o autor também pode alterar manual-

mente a sincronização de efeitos individuais.

Como visto nesta seção, diferentes trabalhos relacionados apresentam abordagens para análise dos quadros de um vídeo e/ou amostras de um áudio para identificar os momentos de execução e outros atributos como cor e intensidade de efeitos sensoriais. As abordagens apresentadas não são gerais o suficiente para identificar componentes complexos que podem ser relacionados a efeitos sensoriais. Um exemplo é identificar a presença de uma flor nos quadros do vídeo para executar um efeito sensorial de aroma. Para estas tarefas mais gerais, redes neurais têm se mostrado boas candidatas, tanto para análise de vídeo [Karpathy et al., 2014] como para áudio [Piczak, 2015a]. Uma contribuição deste trabalho é empregar redes neurais para identificar componentes de cena que podem ser relacionados a efeitos sensoriais, permitindo assim a ativação destes efeitos em sincronia com o conteúdo audiovisual.

Alguns componentes de cena podem ser predominantemente encontrados no vídeo (e.g., relâmpago) ou no áudio (e.g., vento). Outros componentes (e.g., explosão) podem ser encontrados em ambas as modalidades do objeto audiovisual (i.e., áudio e vídeo). Em particular, propomos o uso de uma arquitetura de rede neural bimodal para aumentar a precisão do reconhecimento de componentes de cena especificamente para a tarefa de sincronizar efeitos sensoriais em aplicações *mulsemedia*. Nossa premissa é que combinar as duas modalidades na identificação pode produzir uma melhor precisão quando comparada à identificação de modalidades separadas.

III.2 Redes Neurais para Reconhecimento de Componentes de Cena

Nos últimos anos, várias arquiteturas de rede neural foram propostas como um modelo de fusão para dados de áudio e vídeo em diversos domínios. Uma das primeiras tentativas de usar redes neurais para explorar a correlação entre a informação sonora e visual foi feita por Ngiam et al. [2011]. Os autores aplicam seu modelo para melhorar a precisão do reconhecimento de fala usando não apenas o sinal de áudio, mas também os quadros de imagem dos lábios do locutor. Os autores mostram que melhores características para uma modalidade (e.g., vídeo) podem ser aprendidas se múltiplas modalidades (e.g., áudio e vídeo) são usadas no tempo de aprendizagem das características.

Também visando identificar imagens de pessoas falando, em Torfi et al. [2017], os autores usam redes neurais para realizar correspondência entre áudio com vídeo. Isto é feito identificando características espaço-temporais que são comuns a ambas as modalidades. Outra arquitetura para aprender um modelo para reconhecimento de fala é proposta em Feng et al. [2017]. Este modelo compreende duas arquiteturas de rede neural independentes para cada modalidade (áudio e vídeo), por fim as saídas de ambas as modalidades são fundidas por outra rede neural.

Em Yang et al. [2016] uma rede multimodal de fusão para classificação de vídeo é proposta.

Os autores usam quatro modalidades para extrair informações complementares em várias escalas temporais. Para a fusão de múltiplas camadas e modalidades, eles propõem um modelo de aprendizagem por reforço para aprender a melhor combinação de modalidades. Em contraste com o nosso trabalho atual, os autores não usam modalidade de áudio.

Em Aytar et al. [2016], SoundNet é apresentado, uma arquitetura para reconhecimento de sons. O SoundNet aprende representações de áudio diretamente de sinais de áudio não processados. Os autores treinam a rede de som transferindo conhecimento já adquirido de redes de reconhecimento visual. Os autores alcançam precisão superior em três conjuntos de dados de classificação de sons. Outra abordagem para reconhecer o som natural (ambiente) é apresentada em Boddapati et al. [2017], na qual as redes neurais convolucionais do autor são projetadas especificamente para reconhecimento de objetos em imagens, e podem ser treinadas com sucesso para classificar imagens espectrais de sons naturais. Similar ao nosso trabalho atual, Aytar et al. [2016] e Boddapati et al. [2017] tentam classificar sons naturais. No entanto, eles usam apenas a modalidade sonora, não explorando informações de outras modalidades para ajudar a identificar o som. Além disso, eles não pretendem auxiliar na tarefa de sincronização em aplicações *mulsemedia*.

III.3 Ferramentas de Autoria de Aplicações com Efeitos Sensoriais

No domínio *mulsemedia*, grande parte do esforço de autoria é para sincronizar conteúdos audiovisuais com efeitos sensoriais [Yoon et al., 2010]. Para facilitar a adesão de autores sem (ou com baixo) conhecimento de programação, a criação ou utilização de aplicações em conformidade com MPEG-V é apoiada pelo uso de ferramentas de autoria que permitem a manipulação de arquivos SEM de forma visual.

Considerando esta necessidade, Waltl et al. [2013] apresentam uma ferramenta de autoria projetada para a criação de aplicações *mulsemedia* chamada *SEVino* (*Sensory Effect Video Annotation*). *SEVino* fornece ao autor uma interface que apresenta uma linha de tempo do vídeo a ser usado como base para a aplicação. Esse vídeo representa o conteúdo audiovisual principal com o qual os efeitos sensoriais serão sincronizados. A Figura III.1 apresenta a interface do *SEVino*. A ferramenta permite criar intervalos de tempo que representam a execução de efeitos sensoriais (e.g., nevoeiro, vento, temperatura, etc). Para um determinado intervalo de tempo, os usuários podem indicar um efeito sensorial a ser executado. Após a fase de autoria, a ferramenta gera descrições compatíveis com o padrão MPEG-V indicando os tempos de início e fim dos efeitos criados. As descrições MPEG-V geradas pelo *SEVino* representam os efeitos sensoriais a serem executados em dispositivos físicos.

Shin et al. [2016] apresentam um ambiente de autoria de efeitos sensoriais. Os autores se

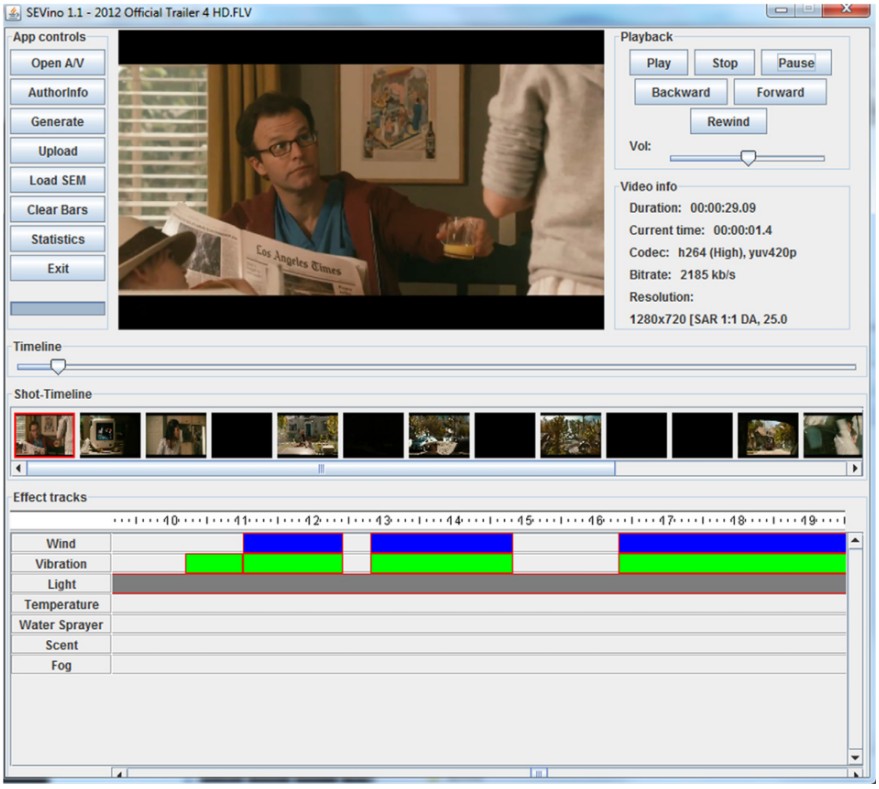

Figura III.1: Interface gráfica do SEVino. Fonte: [Waltl et al., 2013]

concentram em oferecer uma interface gráfica de fácil uso e permitir escalabilidade para uma variedade de aplicações de mídia usadas na indústria, tais como cinemas 4D. Duas ferramentas de autoria são propostas neste trabalho. A ferramenta *Real4DEMaker* permite a autoria de efeitos sensoriais individuais, enquanto a ferramenta *Real4DStudio* pode ser usada para autoria de diversos efeitos sensoriais em uma linha do tempo. O *Real4DStudio* pode ser usado para gerar um grupo de efeitos sensoriais e permite salvá-los para uso posterior.

Apesar dos avanços em ferramentas e modelos para facilitar o esforço de autoria, o processo de criação de uma aplicação *mulsemedia* ainda é um trabalho muito custoso em termos de es-

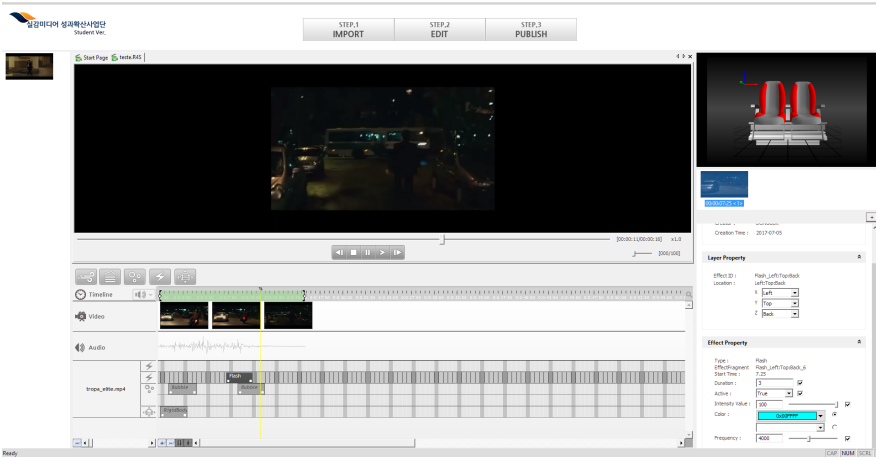

Figura III.2: Inteface gráfica do *Real4DStudio* - Student Version

força e tempo. Tal situação é agravada quando é necessário especificar a sincronização entre um grande número de objetos de mídia audiovisuais e efeitos sensoriais, bem como diferentes tempos de execução desses efeitos.

Diferente dos trabalhos relacionados, a proposta apresentada neste trabalho permite que a especificação dos momentos em que efeitos sensoriais serão executados em uma aplicação seja feita com base nos componentes de cena de interesse. Assim, o esforço de autoria independe do tamanho do objeto de mídia com o qual se quer sincronizar um efeito sensorial e do número de vezes que um determinado componente de cena aparece.

O uso de NCL, permite reusar os relacionamentos de causa-e-efeito da linguagem, permitindo assim definir a sincronização de efeitos sensoriais com os mais diversos objetos de mídia, inclusive considerando a interação do usuário.

III.4 Execução de Efeitos Sensoriais

Segundo o padrão MPEG-V, uma reprodução de efeito sensoriais depende de ferramenta capaz de analisar um arquivo SEM e gerar comandos de execução em dispositivos físicos. Esta ferramenta é chamada de *Media Processing Engine* (MPE), cuja definição está fora do escopo do padrão. Sendo assim diversos autores propõem ferramentas que sejam capazes de sincronizar efeitos sensoriais com objetos de mídia.

As ferramentas capazes de executar efeitos sensoriais sincronizados com conteúdo audiovisual são chamadas de reprodutores (*players*). Estes softwares reproduzem vídeos e renderizam efeitos sensoriais sincronizados com o conteúdo apresentado. Sob a padronização MPEG-V, estes reprodutores conseguem ler um arquivo SEM associado a um conteúdo audiovisual e enviar comandos de ativação para atuadores compatíveis.

No caso de aplicações interativas, tais como jogos e simuladores [Wu et al., 2009], a integração de efeitos sensoriais também pode ser muito vantajosa. Portanto, nesta seção iremos abordar reprodutores de vídeos capazes de indicar a renderização de efeitos sensoriais e também propostas que visam integrar a renderização de efeitos sensoriais em aplicações interativas.

Waltl et al. [2013] apresentam o reprodutor de efeitos sensoriais SEMP (*The Sensory Effect Media Player*). O SEMP é um reprodutor de mídia que oferece a possibilidade de reproduzir vídeos em conjunto com suas descrições de efeitos sensoriais compatíveis com MPEG-V (arquivo SEM). Essas descrições são processadas, sincronizadas com o vídeo e renderizadas em atuadores compatíveis (e.g., Philips amBX, Cyborg Gaming Lights). SEMP suporta a reprodução de efeitos de vibração, vento, luz, temperatura, *spray*, aroma e névoa. Este reprodutor conta com o algoritmo de cálculo de cor baseado em quadros de vídeo como mencionado na Seção III.1.

Saleme and Santos [2015] apresentam a plataforma *PlaySEM*, uma abordagem de reprodutor

de efeitos sensoriais compatível com MPEG-V que opera independente de um reprodutor de mídia audiovisual. A plataforma compreende duas ferramentas desenvolvidas em Java. Um *player* de vídeo e um renderizador de efeitos sensoriais (*SER* do inglês *Sensory Effects Renderer*). O *player* atua como um cliente para o SER, transmitindo o arquivo SEM além de comandos de início de reprodução, pausa e fim da reprodução do objeto de mídia. O SER é responsável por processar o arquivo SEM e ativar os atuadores no tempo dado na descrição do SEM. Para ativar os atuadores, o SER envia mensagens que são encaminhadas para um microcontrolador *Arduino*, que por sua vez aciona os atuadores responsáveis por renderizar efeitos sensoriais de vento, luz e vibração. Esta abordagem apresenta o SER desacoplado do *player* para permitir seu reuso por outras aplicações.

Um reuso desta plataforma foi apresentado em [Santos et al., 2015] em que os autores usam uma abordagem orientada a eventos para sincronizar efeitos sensoriais em ambientes interativos. Os autores criaram uma aplicação que identifica movimentos do usuário e ativa atuadores de efeitos sensoriais distribuídos na sala. Para esta aplicação, os comandos de efeitos sensoriais foram gerados previamente e guardados em disco. Foi definida uma base de movimentos pré-definidos que ativam estes comandos. No artigo, os autores usaram o *Kinect*[1] para identificar movimentos do usuário. Quando um movimento é reconhecido, a aplicação envia para o SER os arquivos SEM pré-definidos correspondentes ao movimento. O SER então faz a leitura do arquivo SEM e converte essas descrições em comandos para o microcontrolador responsável por acionar os dispositivos físicos.

Em outra proposta de utilização de efeitos sensoriais em aplicações interativas, Guedes et al. [2016b] investigam o suporte para sincronização de sensores e atuadores através de uma linguagem multimídia. Os autores utilizam a linguagem NCL para especificar a sincronização de sensores e atuadores, além dos objetos de mídia. Para isto, os autores propõem o uso de *scripts* Lua para controlar dispositivos IoT (*Internet of Things*) em aplicações NCL. Cada dispositivo é manipulado por um objeto de mídia Lua que se comunica com os serviços disponibilizados pelo dispositivo na rede local. Os autores utilizam REST (*Representational State Transfer*) para enviar comandos do *script* Lua para um código em *JavaScript* que é responsável por iniciar os atuadores. Um ponto negativo dessa proposta é que o código desenvolvido para a aplicação NCL foi extremamente acoplado aos atuadores usados, visto que os autores da aplicação precisavam endereçar diretamente os atuadores para os quais os comandos de ativação deveriam ser enviados. Este trabalho foi estendido em [Guedes et al., 2016a], em que os autores propõem uma série de extensões para a linguagem NCL com suporte a interações multimodais e multiusuário. Estas extensões visam permitir o acesso a sensores e atuadores pelo próprio formatador da linguagem.

[1] https://developer.microsoft.com/pt-br/windows/kinect

Grande parte das ferramentas de autoria e reprodução são desenvolvidas em código imperativo e permitem, de forma geral, que o usuário crie e reproduza vídeos em sincronia com efeitos sensoriais. No entanto, estas abordagens são restritas a apenas aplicações cujo foco é um conteúdo audiovisual acompanhado de efeitos sensoriais. Tais ferramentas seguem um paradigma de linha do tempo e não contam com suporte a interatividade do usuário. Elas também não permitem a extensão de suas funcionalidades, para permitir sua integração em novas aplicações.

Este trabalho visa viabilizar a criação de aplicações interativas com efeitos sensoriais, assim como os trabalhos mostrados nesta seção. Diferente do trabalho de [Guedes et al., 2016b], a abordagem aqui proposta favorece a interoperabilidade com vários atuadores, pois é usado o padrão MPEG-V como base para execução de efeitos sensoriais nos dispositivos atuadores disponíveis no ambiente. Ainda, a abordagem proposta não altera a linguagem NCL, permitindo que o atual formatador da linguagem possa ser utilizado para a execução de aplicações com efeitos sensoriais. A abordagem seguida é gerar código compatível com o padrão MPEG-V em tempo de execução por meio de um conjunto de parâmetros especificados em NCL que descrevem o efeito a ser executado.

Neste sentido, a abordagem seguida é similar àquela apresentada em [Santos et al., 2015]. Porém em [Santos et al., 2015] é necessário o uso de uma linguagem imperativa para desenvolver aplicações. Neste trabalho, a criação de aplicações é feita usando uma linguagem declarativa, diminuindo a dificuldade no desenvolvimento de tais aplicações e permitindo uma separação clara entre o código da aplicação e aquele relacionado a renderização de efeitos sensoriais. Ainda, o padrão MPEG-V é estendido para aprimorar a definição de posicionamento de efeitos sensoriais, através de pequenas modificações no seu sistema de referências espaciais.

III.5 Simuladores de efeitos sensoriais

Para facilitar ao autor visualizar a sincronização de efeitos sensoriais com conteúdo de vídeo, algumas ferramentas de autoria implementam simuladores de efeitos sensoriais. Essas ferramentas se mostram necessárias pois nem sempre é possível para o autor executar a aplicação no seu ambiente alvo para verificar seu comportamento.

O simulador *SESim* (*Sensory Effect Simulator*), desenvolvido por [Waltl et al., 2013], tem o intuito de simular os efeitos sensoriais criados na ferramenta de autoria SEVino. O simulador permite visualizar representações virtuais de atuadores num plano 2D sendo ativados em conjunto com a reprodução do conteúdo audiovisual. Os dispositivos atuadores virtuais são acionados de acordo com os efeitos descritos em um arquivo SEM. O simulador apresenta os valores assumidos por cada dispositivo em tempo de execução. Quando acionado um dispositivo, o retângulo que o inscreve assume uma borda na cor vermelha. Assim o autor de uma aplicação pode obser-

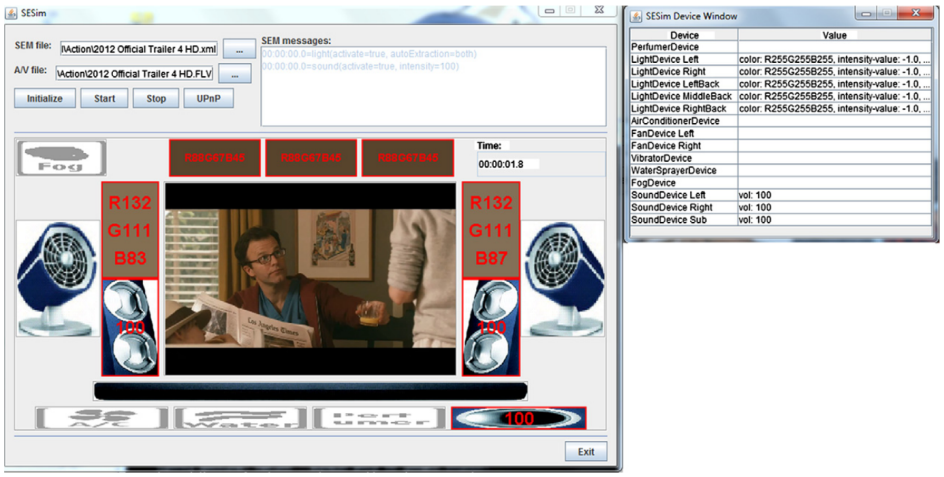

Figura III.3: Interface gráfica de SESim. Fonte:[Waltl et al., 2013]

Figura III.4: Interface gráfica de Sensible Media Simulator. Fonte:[Kim et al., 2013]

var a ativação e desativação dos dispositivos num determinado instante.

O simulador *Sensible Media Simulator* [Kim et al., 2013] foi desenvolvido com o intuito de representar os dispositivos atuadores e sensores presentes em automóveis. A interface gráfica do simulador é apresentada na Figura III.4. Em sua interface é possível visualizar efeitos de calor, vibração, luz e temperatura. O simulador também pode simular efeitos sensoriais para situações predefinidas de condução de veículo, como frenagem repentina, estacionamento em reverso e detecção de colisão. Além destas situações, o simulador também permite a simulação de um objeto de mídia acompanhado de efeitos sensoriais.

Este simulador é compatível com o padrão MPEG-V e têm como entrada um vídeo e um arquivo SEM com descrição de efeitos sensoriais. Quando um vídeo é reproduzido, os comandos de execução de efeitos a serem enviados para os atuadores são combinados com as preferên-

Figura III.5: Interface gráfica de Simulador 3D incorporado a *Real4DStudio*. Fonte:[Shin et al., 2016]

cias do usuário e as capacidades dos atuadores disponíveis. O simulador então apresenta os efeitos adaptados em sua interface gráfica. O simulador também possui comunicação com um microcontrolador que permite a transmissão destes comandos para a reprodução dos efeitos em atuadores físicos.

Shin et al. [2016] propõem um simulador para representar efeitos sensoriais incorporado a ferramenta de autoria *Real4DStudio*. A diferença para os demais é que este simulador apresenta um ambiente virtual tridimensional (3D). O simulador permite a visualização em 3D de diversos efeitos sensoriais, tais como névoa, efeitos de calor e vento. Sua interface gráfica é apresentada na Figura III.5. Cada atuador de efeito sensorial é representado como um modelo 3D de um atuador real. Os atuadores ficam fixos em suas posições. Os efeitos sensoriais são representados como animações de efeitos nestes atuadores, tal como um vento sendo gerado por seu atuador correspondente. A ferramenta permite a visualização do ambiente em 5 modos distintos: frente, trás, lado esquerdo, lado direito e topo.

Simuladores mostram-se essenciais para ajudar no processo de autoria. Um simulador permite que o autor possa desenvolver aplicações visando um ambiente específico ou testar sua aplicação em ambientes com números variados de atuadores. Entretanto, os simuladores atuais não exploram a capacidade de adição de novos atuadores ou movimentação de atuadores existentes em um ambiente tridimensional, limitando-se a apenas reprodução de efeitos. Por fim, uma característica importante é que os simuladores atuais não permitem que o autor veja o posicionamento de efeitos sensoriais, característica necessária para que o autor crie animações complexas de efeitos no ambiente.

Neste trabalho, é apresentado um simulador de efeitos sensoriais em 3D. O simulador também

é compatível com o padrão MPEG-V e, diferente dos trabalhos relacionados, permite a ativação de efeitos sensoriais de maneira orientada a eventos. Tal escolha dá suporte à reprodução de efeitos sensoriais em aplicações interativas. Além disso, no simulador, é permitida a configuração do ambiente para adicionar, remover ou movimentar atuadores. Adicionalmente, o simulador implementa uma extensão do padrão MPEG-V proposta neste trabalho que permite um endereçamento espacial de atuadores de forma mais precisa.

Capítulo IV Sincronização baseada no conteúdo de mídia

Neste capítulo é apresentado o conceito de âncoras abstratas como forma de permitir a autoria semiautomática da sincronização de elementos de uma aplicação baseada no conteúdo de uma mídia. Na Seção IV.1 é formalizada a definição do conceito de âncoras abstratas. Na Seção IV.2 este conceito é instanciado para a linguagem NCL. Na Seção IV.3 é apresentado um processador que permite a utilização de âncoras abstratas em NCL. Nesta seção também é apresentada a implementação do processador utilizando um software de reconhecimento baseado em redes neurais para realizar a identificação de componentes de cena em um vídeo. Por fim, na Seção IV.4 apresenta um caso de uso para ilustrar como o uso de âncoras abstratas facilita a autoria uma aplicação multimídia com efeitos sensoriais.

IV.1 Âncoras Abstratas

Como mencionado na Seção II.2, aplicações multimídia são descritas em documentos multimídia e a especificação de tais documentos é feita usando alguma linguagem de autoria multimídia. Para linguagens de autoria declarativas, as entidades comuns são *nós*, que representam o conteúdo e parte da estrutura do documento e elos (*links*), para representar os relacionamentos de sincronização a serem levados em conta durante a execução da uma aplicação.

Algumas linguagens de autoria, tal como NCL, fornecem ainda o conceito de âncoras para representar um pedaço de um conteúdo de nó. Este pedaço pode representar parte do conteúdo espacial ou temporal do nó. Âncoras espaciais representam uma região dentro de um nó. Por exemplo, uma área envolvendo uma forma específica em uma imagem. Já as âncoras temporais representam um pedaço de um conteúdo de nó no eixo do tempo. Por exemplo, uma sequência de quadros de um nó de vídeo ou uma sequência de amostras em um nó de áudio. Normalmente, as âncoras temporais são definidas por um tempo de começo e fim[1], em relação ao início da apresentação do conteúdo total do nó.

Conforme discutido no Capítulo II.3. Linguagens de autoria de *templates* permitem ao usuário abstrair algumas etapas do processo de autoria pelo uso de uma descrição mais genérica. Após a autoria, no tempo de processamento, um processador de *templates* é usado para “preencher

[1] É importante notar que mesmo quando uma âncora temporal é definida em função de outro atributo, como número de quadros ou amostras, este valor sempre pode ser mapeado para o eixo do tempo.

os espaços" deixados em branco no *template* com conteúdo específico da aplicação final.

A ideia de Âncoras Abstratas [Abreu and dos Santos, 2017a,b] se assemelha a definição de *templates*. Componentes genéricos são definidos, os quais são posteriormente instanciados por um processador. Neste trabalho, ao invés do componente genérico representar um objeto de mídia como um todo, ele representa um componente de cena que faz parte do conteúdo desse objeto de mídia. Assim, o autor de aplicações multimídia pode fazer uso de âncoras abstratas para representar pedaços do conteúdo de um nó, sem explicitamente descrever seus tempos de início e fim. Mais ainda, sem precisar indicar os tempos de início e fim de todas as ocorrências desse pedaço do conteúdo. O conceito de âncoras abstratas é definido como segue.

Definição 2 (Âncora Abstrata) *Uma Âncora Abstrata é uma entidade que representa, possivelmente, várias âncoras de um nó. Ela está relacionada a um componente da cena de modo que todas as suas instâncias representam momentos da apresentação de um nó quando este componente da cena é apresentado.*

A Figura IV.1 apresenta essa ideia, em que nós são representados como círculos e as âncoras desses nós são representadas como quadrados. Linhas tracejadas associam uma âncora a um nó e linhas sólidas representam relacionamentos (*links*) entre nós e, possivelmente, âncoras. A parte superior da Figura IV.1 apresenta um documento em que o objeto de mídia *video* possui três âncoras abstratas: *âncoraA*, *âncoraB* e *âncoraC*. Cada uma representando um determinado componente de cena. Os *links* entre tais âncoras e outros objetos de mídia definem quando tais mídias devem ser apresentadas em relação ao conteúdo do objeto de mídia *video*.

Antes da execução, um documento usando âncoras abstratas deve ser processado em um documento final na linguagem alvo. O processamento realizado é semelhante ao realizado para linguagens de *templates*. O primeiro passo do processo é instanciar as âncoras abstratas para os componentes da cena que eles especificam. O segundo passo é duplicar os links relacionados a essas âncoras abstratas para cada uma de suas instâncias. A visão geral desse processo é mostrada na Figura IV.1. A Seçao IV.2 apresenta a extensão da linguagem NCL para dar suporte a definição de âncoras abstratas.

A instanciação de âncoras abstratas é feita por um processador conforme será apresentado na Seção IV.3. O processador utiliza software de reconhecimento de cena para reconhecer os instantes de tempo em que um determinado componente de cena é apresentado no conteúdo da mídia. Então o processador cria instâncias das âncora abstratas contendo a informação temporal da presença ou não de tal componente. Portanto, a abordagem aqui proposta exige dos autores pouco (ou nenhum) conhecimento prévio sobre o conteúdo da mídia. A definição temporal das âncoras é realizada inteiramente com dados adquiridos pelo software de reconhecimento.

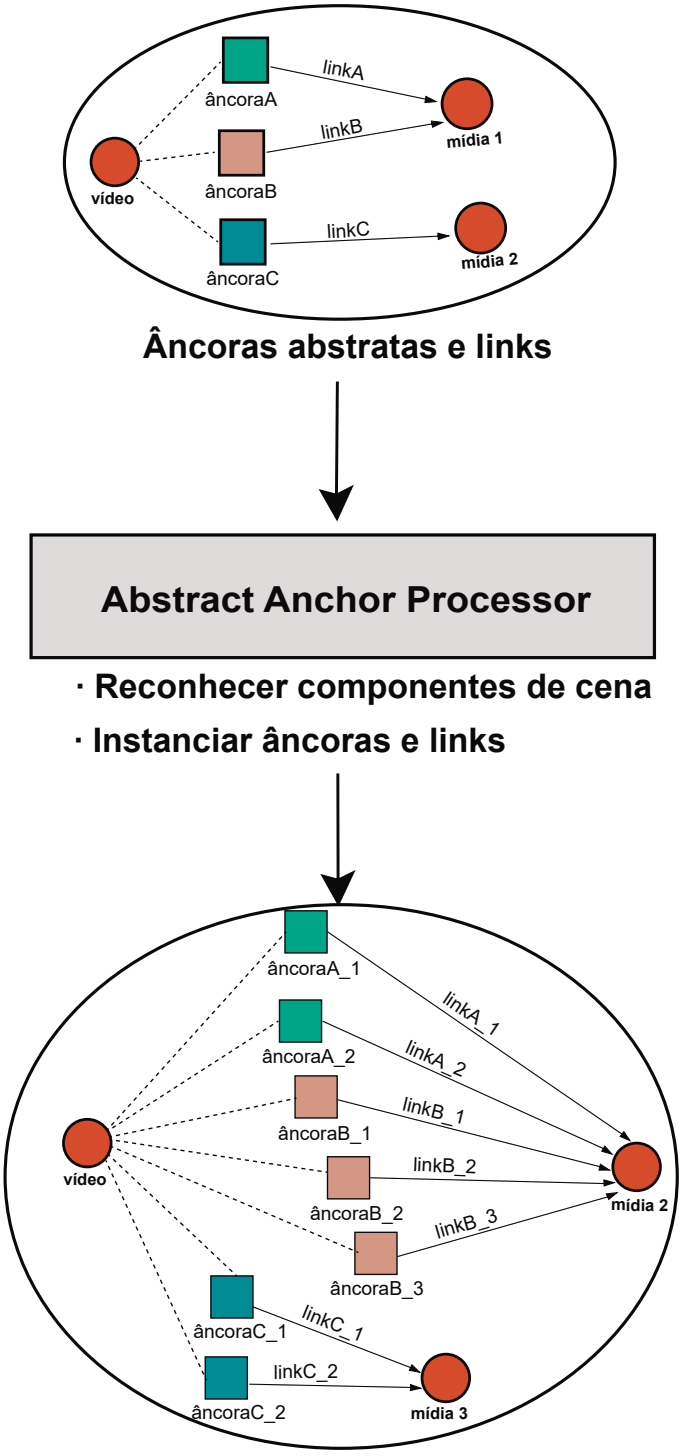

Figura IV.1: Definição e processamento de âncoras abstratas

IV.2 Extensão de NCL para uso de âncoras abstratas

NCL fornece o elemento `media` para definir nós que representam objetos de mídia. Ela também permite a definição de âncoras usando elemento `area`, filho do elemento mídia. A Listagem IV.1 apresenta um exemplo de especificação de mídia[2] com âncoras abstratas em NCL.

[2]Neste texto, o termo *mídia* é usado tanto para referir a um objeto de mídia quanto a um elemento `media` de NCL, sem prejuízo do entendimento. Quando necessário, os termos específicos são utilizados.

```
<media id="video1" src="video.mp4">
    <area tag="sea"/>
    <area tag="snow"/>
    <area tag="sun"/>
</media>
```

Listagem IV.1: Exemplo de especificação de mídia com âncoras abstratas em NCL

Para permitir a definição de âncoras abstratas, a linguagem NCL foi estendida de forma que os elementos `area` (âncora em NCL) possam definir um novo atributo `tag`. Esse atributo indica o componente de cena relacionado a essa âncora. No exemplo apresentado na Listagem IV.1, são criadas três âncoras abstratas, uma representando os instantes quando o mar aparece na mídia (*sea*), outra representando instantes quando a neve aparece na mídia (*snow*) e a terceira representando os instantes quando o sol aparece (*sun*).

Linguagens multimídia baseadas em eventos fornecem relações causais de modo que, quando um evento especificado como sua condição acontece, uma ou mais ações são acionadas. Em NCL isto é feito por meio da definição de relacionamentos. Os relacionamentos são definidos usando pares de elementos link-conector. Conectores [Muchaluat-Saade and Soares, 2002] definem uma relação geral que é instanciada por links para um determinado conjunto de participantes. A Listagem IV.2 apresenta um exemplo de especificação de conectores e links para as âncoras definidas na Listagem IV.1. É importante notar que os conectores são especificados apenas uma vez no cabeçalho do documento e, no corpo do documento, os links podem instanciar essas relações um número qualquer de vezes.

```
<!-- em head -->
<connectorBase >
    <causalConnector id="onBeginStart">
        <simpleCondition role="onBegin"/>
        <simpleAction role="start"/>
    </causalConnector>

    <causalConnector id="onEndStop">
        <simpleCondition role="onEnd"/>
        <simpleAction role="stop"/>
    </causalConnector>
</connectorBase>

<!-- em body -->
<link xconnector="onBeginStart">
    <bind role="onBegin"component="video1" interface="sea"/>
    <bind role="start" component="wind"/>
</link>
<link xconnector="onBeginStart">
    <bind role="onBegin"component="video1" interface="sun"/>
    <bind role="start" component="heat"/>
</link>
<link xconnector="onBeginStart">
    <bind role="onBegin" component="video1" interface="snow"/>
    <bind role="start" component="cold"/>
</link>
```

```
27
28 <link xconnector="onEndStop">
29     <bind role="onEnd"component="video1" interface="sea"/>
30     <bind role="stop" component="wind"/>
31 </link>
32 <link xconnector="onEndStop">
33     <bind role="onEnd"component="video1" interface="sun"/>
34     <bind role="stop" component="heat"/>
35 </link>
36 <link xconnector="onEndStop">
37     <bind role="onEnd" component="video1" interface="snow"/>
38     <bind role="stop" component="cold"/>
39 </link>
```

Listagem IV.2: Exemplo de especificação de conectores e links em NCL

IV.3 Processador de Âncoras Abstratas

A arquitetura do Processador de Âncoras Abstratas (*Abstract Anchor Processor* - Abstract Anchor Processor (AAP)) é apresentada na Figura IV.2.

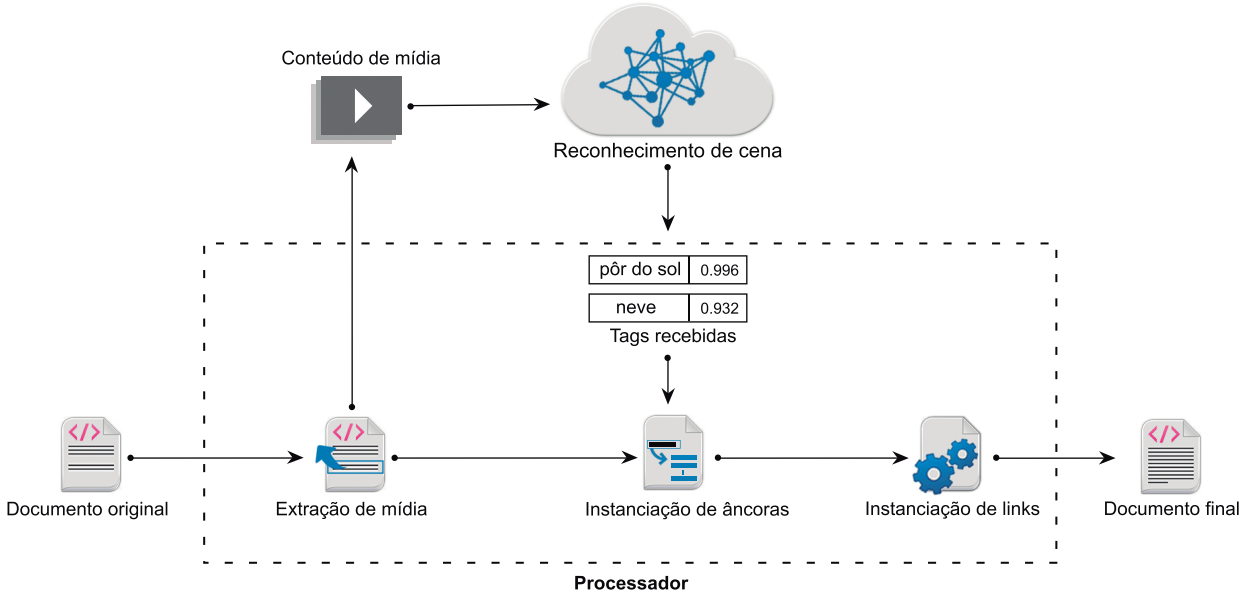

Figura IV.2: Arquitetura do Processador de Âncoras Abstratas

O AAP recebe como entrada um documento contendo âncoras abstratas definidas pelo autor. A ferramenta analisa o documento, identificando os nós que definem âncoras abstratas e links relacionados a elas. Nesta etapa, o processador também extrai o conteúdo de mídia desses nós. Para o exemplo na Listagem IV.1, o processador identifica o nó *video1* como um nó que define âncoras abstratas e deve extrair seu conteúdo (arquivo `video.mp4`).

O conteúdo de mídia extraído é enviado para um software externo para reconhecimento de cena. Como pode ser visto na Figura IV.2, o software de reconhecimento é desacoplado do processador. Essa abordagem dá mais liberdade ao autor, permitindo que se use diferentes

softwares de reconhecimento de cena. O passo de reconhecimento de cena resulta em um conjunto de *tags*[3] que são equivalentes às identificadas no atributo de mesmo nome nas âncoras abstratas definidas pelo autor. Essas *tags* representam os componentes da cena juntamente com informações temporais que indicam quando estes aparecem na mídia.

IV.3.1 Instanciação de Âncoras

Provido com as *tags* recebidas do software de reconhecimento de cena, o AAP instancia as âncoras. O processo de instanciação é realizado da seguinte forma. De acordo com as âncoras abstratas encontradas no documento, o processador verifica no conjunto de *tags* recebidas os instantes de tempo quando os componentes de cena descritos estão presentes. Quando uma primeira correspondência for encontrada, ou seja, quando encontrar a primeira *tag* que contenha o nome do componente de cena descrito na âncora abstrata, o AAP cria uma instância da âncora. Esta instância possuirá como *id* o nome do componente de cena em conjunto com o número da instância. O AAP também adiciona como início da âncora (atributo *begin* em NCL) a informação temporal associada com a *tag* recebida do software de reconhecimento. Por fim o AAP salva em uma lista o *id* da instância da âncora abstrata para ser usado futuramente para instanciar os *links* da aplicação.

O AAP então continua a busca por *tags* relacionadas a componentes de cena para o próximo passo de tempo. A partir deste momento, dois casos são possíveis.

- O passo de tempo já possui a mesma *tag* referente ao componente de cena. Nesse caso, o AAP vai prosseguir sua execução.
- O passo de tempo atual possui a *tag* referente ao componente de cena, porém o próximo passo de tempo não a possui. Neste caso, o AAP marca o tempo final da âncora como sendo o passo de tempo da última *tag* encontrada e salva a nova instância no documento NCL.

O AAP então continua esse processo, mas buscando uma nova ocorrência da *tag*. Caso seja encontrada, o AAP começa uma nova instância da âncora e realiza o processo acima novamente. Terminada a varredura das *tags*, o processador retorna o processo, passando a instanciar a próxima âncora abstrata encontrada no documento.

O processo continua até não existirem mais âncoras abstratas a serem instanciadas. O Algo-

[3] Usamos a mesma nomenclatura que o software de reconhecimento. Não deve ser confundido com tags XML.

ritmo 1 resume esse processo.

Algoritmo 1: INSTANCIAÇÃO DE ÂNCORAS ABSTRATAS EM DOCUMENTO NCL

Entrada: DOC : Documento NCL com âncoras abstratas

$tags$: Resultado do software de reconhecimento de cena

Saída: DOC : Documento NCL com instâncias de âncoras abstratas

$AAList$: Lista de âncoras abstratas e seus respectivos id's de instâncias

```
1  início
2      AAlist = BUSCAR_AAS(DOC)
3      para cada AA em AAList faça
4          acc = 0
5          para k = 0 até k < lenght(tags) faça
6              se AA.nome ∈ tags[k] então
7                  se k == 0 ∨ AA.nome ∉ tag[k − 1] então
8                      a = criarAncora(DOC)
9                      a.id = CONCATENAR(AA, acc)
10                     a.begin = k ∗ step
11                     AA.ids[acc] = a.id            // guardar o id de cada instância
12                 fim
13                 se AA.nome ∉ tags[k + 1] então
14                     a.end = k ∗ step
15                     acc = acc + 1
16                 fim
17             fim
18         fim
19     fim
20 fim
21 retorna DOC, AAList
```

Ao fim da execução, o AAP apaga as descrições de âncoras abstratas do documento, mantendo apenas suas instâncias. A Listagem IV.3 apresenta o resultado da etapa de instanciação das âncoras abstratas para o exemplo da Listagem IV.1.

```
<media src="video.mp4" id="video1">
    <area id="sea_1" begin="01s" end="09s"/>
    <area id="sea_2" begin="17s" end="19s"/>
    <area id="snow_1" begin="05s" end="16s"/>
    <area id="snow_2" begin="20s" end="26s"/>
    <area id="sun_1" begin="01s" end="19s"/>
    <area id="sun_2" begin="28s" end="32s"/>
</media>
```

Listagem IV.3: Resultado da etapa de instanciação das âncoras definidas no exemplo da Listagem IV.1

No exemplo apresentado na Listagem IV.3, o componente de cena `sea` foi identificado nos intervalos $[1, 9]$ e $[17, 19]$ segundos. Assim, duas instâncias de âncora foram criadas, `sea_1` para o primeiro intervalo e `sea_2` para o segundo. O mesmo é feito para o componente de cena `snow`, que foi identificado dentro de intervalos $[5, 16]$ e $[20, 26]$, gerando instâncias de âncora `snow_1` e `snow_2`. Por fim, o componente de cena `sun`, foi identificado dentro de intervalos $[1, 19]$ e $[28, 32]$, gerando instâncias de âncora `sun_1` e `sun_2`.

Vale ressaltar que, no documento resultante, o atributo `tag` foi removido das instâncias de âncoras abstratas. O atributo `id` das âncoras, que são obrigatórios em NCL, são criados pelo processador de acordo com o valor do atributo `tag`. Para manter a compatibilidade de saída com o padrão NCL, cada id de âncora também é incrementado para ser único em todo o documento.

IV.3.2 Instanciação de Links

Após o processo de instanciação de âncoras, o AAP é capaz de instanciar links que se referem a âncoras abstratas. O AAP percorre as a lista âncoras abstratas definidas no documento e examina se existem links cujos elementos *bind* referenciem essa âncora abstrata. Dois casos são possíveis.

- Não são encontrados tais *links*. Nesse caso, nada tem que ser feito.
- São encontrados um ou mais *links*. Nesse caso, cada *link* no documento deverá ser duplicado para cada instância da âncora abstrata criada na etapa anterior.

Esse processo continua até que nenhum *bind* de link esteja vinculado a uma âncora abstrata. O Algoritmo 2 ilustra esse processo.

A Listagem IV.4 apresenta o resultado da etapa de instanciação dos links para o exemplo na Listagem IV.2.

```
1 <link xconnector="onBeginStart">
2     <bind role="onBegin" component="video1" interface="sea_1"/>
3     <bind role="start" component="wind"/>
4 </link>
5 <link xconnector="onBeginStart">
6     <bind role="onBegin" component="video1" interface="sea_2"/>
7     <bind role="start" component="wind"/>
8 </link>
```

```
<link xconnector="onBeginStart">
    <bind role="onBegin" component="video1" interface="snow_1"/>
    <bind role="start" component="cold"/>
</link>
<link xconnector="onBeginStart">
    <bind role="onBegin" component="video1" interface="snow_2"/>
    <bind role="start" component="cold"/>
</link>
<link xconnector="onBeginStart">
    <bind role="onBegin" component="video1" interface="sun_1"/>
    <bind role="start" component="heat"/>
</link>
<link xconnector="onBeginStart">
    <bind role="onBegin" component="video1" interface="sun_2"/>
    <bind role="start" component="heat"/>
</link>
<link xconnector="onEndStop">
    <bind role="onEnd" component="video1" interface="sea_1"/>
    <bind role="stop" component="wind"/>
</link>
<link xconnector="onEndStop">
    <bind role="onEnd" component="video1" interface="sea_2"/>
    <bind role="stop" component="wind"/>
</link>
<link xconnector="onEndStop">
    <bind role="onEnd" component="video1" interface="snow_1"/>
    <bind role="stop" component="cold"/>
</link>
<link xconnector="onEndStop">
    <bind role="onEnd" component="video1" interface="snow_2"/>
    <bind role="stop" component="cold"/>
</link>
<link xconnector="onEndStop">
    <bind role="onEnd" component="video1" interface="sun_1"/>
    <bind role="stop" component="heat"/>
```

```
44 </link>
45 <link xconnector="onEndStop">
46     <bind role="onEnd" component="video1" interface="sun_2"/>
47     <bind role="stop" component="heat"/>
48 </link>
```

Listagem IV.4: Resultado da etapa de instanciação de link para o exemplo na Listagem IV.2

Algoritmo 2: INSTANCIAÇÃO DE LINKS EM DOCUMENTO NCL

Entrada: DOC : Documento NCL com instâncias de âncoras abstratas
$AAList$: Lista de âncoras abstratas e seus respectivos id's de instâncias
Saída: Documento com instâncias de links

```
1  início
2      para cada AA em AAList faça
3          links = BUSCAR_LINKS(DOC, AA.nome)
4          para cada l em links faça
5              para cada id em AA.ids faça
6                  instanciarLink(DOC, link = l, interface = id)
7              fim
8          fim
9      fim
10 fim
11 retorna DOC
```

No exemplo apresentado na Listagem IV.4, o primeiro link da Listagem IV.2 foi instanciado para ambas as instâncias da âncora abstrata `sea`. Os links resultantes agora têm como objeto as âncoras `sea_1` e `sea_2`, respectivamente. O mesmo processo foi feito para o segundo link da Listagem IV.2, que foi instanciado para âncoras `snow_1` e `snow_2` e para o terceiro link que foi instanciado para âncoras `sun_1` e `sun_2`. De forma análoga, os links responsáveis por parar os efeitos (`onEndStop`) também foram instanciados.

IV.3.3 Implementação

Seguindo a abordagem da arquitetura do AAP, a ferramenta foi implementada como três módulos principais. A Figura IV.3 ilustra este cenário. O primeiro módulo é um script Lua responsável por analisar um documento NCL e posteriormente instanciar âncoras abstratas e links, aqui chamado de *Processador*. O segundo é o módulo de comunicação com o software de reconhecimento, neste trabalho chamado de *Módulo Intermediário*. Conforme mencionado anteriormente, este módulo é desacoplado do processador pois é desejado que o processador possa se adaptar a diversos tipos de mídias e softwares de reconhecimento. Por fim, o terceiro módulo é o software de reconhecimento de cena que é implementado de forma independente. As seções a

seguir apresentam os módulos em mais detalhes.

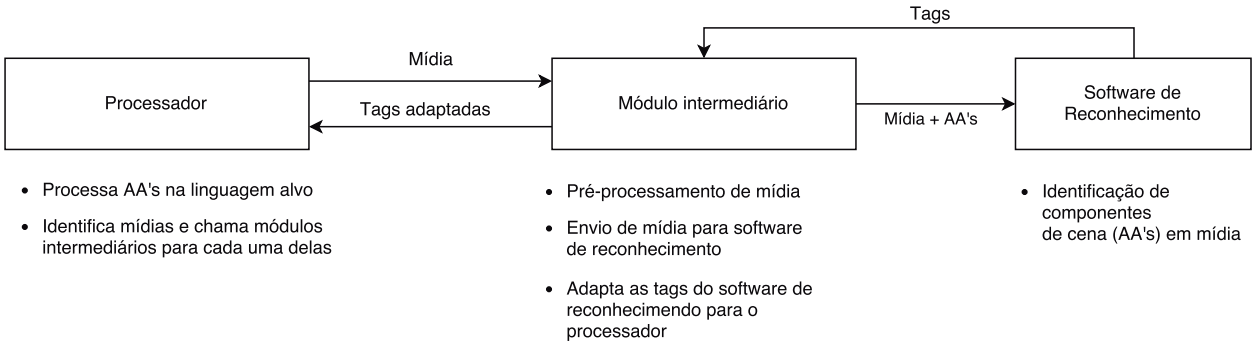

Figura IV.3: Módulos do AAP

Processador

Este módulo foi implementado como um *script* Lua que faz o *parsing* de um documento NCL contendo âncoras abstratas. O processador busca por mídias cujas âncoras (ao menos uma) possua um atributo `tag` e, então, para cada mídia, executa uma chamada a um módulo intermediário. O módulo intermediário utilizado irá depender do tipo da mídia identificada. Esta abordagem permite que novas ferramentas de reconhecimento de cena sejam adicionados ao AAP.

O processador também é responsável por receber um arquivo descritivo de *tags* vindas do módulo intermediário e transformar, com base nesta informação, o documento original em um documento NCL padrão. Após receber as *tags*, o processador realiza a instanciação de âncoras e links de acordo com o processo descrito nas Seções IV.3.1 e IV.3.2.

Módulo intermediário

A cada mídia identificada pelo processador, um módulo intermediário é chamado para tratar da comunicação com o software de reconhecimento. O módulo faz o pré-processamento da mídia para adaptar o conteúdo da mídia a um formato compatível com o software de reconhecimento. O módulo intermediário também pode adaptar características da mídia para ser enviada para um software de reconhecimento. Esta adaptação visa, em situações de comunicação de rede, diminuir o tráfego de dados da mídia para o software de reconhecimento. Por exemplo, para um reconhecimento de vídeo, o som é desnecessário, portanto este som pode ser removido para diminuir o tamanho do arquivo a ser enviado. Da mesma forma, para reconhecimento de som o vídeo é desnecessário, portanto o som pode ser extraído do arquivo de vídeo e enviado separadamente para o software de reconhecimento.

O módulo também recebe como resposta dos softwares de reconhecimento um conjunto de *tags* que descrevem os componentes de cena junto com probabilidades deste componente estar

presente na cena. Um arquivo com *tags* descreve também informações temporais que indicam o momento em que cada componente de cena aparece. Exemplos de tais *tags* podem ser vistas na Seção IV.3.3. Em nosso trabalho, por padrão, os trechos de informação temporal das *tags* são de um segundo. É importante notar que após a fase de instanciação, as instâncias das âncoras no documento final terão esta granularidade deste padrão de 1s (um segundo). A granularidade pode ser modificada nas configurações do AAP, podendo ter valores maiores que um segundo ou frações de um segundo. Dados os resultados sobre QoE apresentados na Seção II.4 ([Ghinea et al., 2014; Yuan et al., 2014; Hülsmann et al., 2014; Ghinea and Ademoye, 2010]), pode-se considerar que tal granularidade de um segundo para aplicações com efeitos sensoriais não deve representar uma ameaça para a qualidade de experiência do usuário.

Ao receber estas *tags*, o módulo intermediário compõe um arquivo de descrição de cena, expondo os componentes de cena identificados para cada trecho da mídia. A Listagem IV.5 apresenta um exemplo deste arquivo. Neste arquivo, é definida um objeto *json* [Crockford, 2006] que representa a mídia que foi reconhecida. Este objeto possui como atributo chave o nome da mídia foi reconhecida, neste caso `video.mp4`. O valor referente a chave é um *array* onde cada índice representa um trecho de informação temporal de reconhecimento. Caso um trecho de informação temporal não possua nenhuma informação de reconhecimento, este índice do *array* não terá valores, como visto na linha 4 da Listagem IV.5. Estes trechos são usados para guiar o AAP a percorrer o tempo total da mídia para definir as informações temporais das instâncias de âncoras.

A composição do arquivo foi pensada para evitar que alterações na saída do software de reconhecimento prejudiquem a interoperabilidade da ferramenta. Assim a integração com diversos softwares de reconhecimento é permitida, requerendo apenas uma adaptação do módulo intermediário para cada caso, mantendo ainda a compatibilidade com a etapa de instanciação do processador AAP.

Outra configuração padrão do AAP é selecionar apenas *tags* com probabilidade $\geq 0,9$ para compor este arquivo intermediário. Esta condição limita que conceitos de cenas que foram reconhecidos de forma incerta sejam atribuídos a efeitos sensoriais.

```
1 "video.mp4": [
2         ["musica_de_rua", "sirene","motor_de_carro"],
3         ["sirene","motor_de_carro"],
4         [],
5         ["ar_condicionado"]
6     ]
```

Listagem IV.5: Arquivo de descrição de cena compatível com o AAP

Reconhecimento de cena

Dado um conjunto de âncoras abstratas previamente definidas pelo autor, o AAP coleta as mídias relacionadas a estas descrições e as envia para análise. As *tags* resultantes devem ser instanciadas com informações temporais que identifiquem os momentos que essa *tag* apareceu na cena. Aqui chamamos esse processo de *Reconhecimento de Cena*.

O reconhecimento de cena é alcançado enviando o conteúdo de mídia para um software de reconhecimento, que é um software que emprega algoritmos que podem detectar componentes de cena no conteúdo da mídia audiovisual. Essas abordagens retornam um conjunto de *tags* que indicam a descrição de componentes em um determinado momento no conteúdo de mídia. Embora a mídia estática também possa ser analisada (imagem e texto), este trabalho se concentra em objetos de mídia contínua, que são frequentemente usados como base para a sincronização de efeitos sensoriais.

Em uma primeira implementação do AAP foi usada uma API (*Application Programming Interface*) de reconhecimento de vídeo[4] com base em redes neurais convolucionais [LeCun et al., 1989]. Essas redes neurais se mostram um método eficaz para analisar o conteúdo de imagem e vídeo de acordo com [Karpathy et al., 2014; Zeiler and Fergus, 2013]. A Figura IV.4 mostra o resultado do reconhecimento de imagem usando esse software.

Figura IV.4: Resultado de reconhecimento de imagem

O exemplo na Figura IV.4 retorna um conjunto de *tags* que indicam os componentes da cena presentes na imagem. Cada *tag* é seguida pela probabilidade de previsão da rede. A rede pode identificar tanto objetos (e.g., barco), bem como conceitos individuais (e.g., reflexo).

Para reconhecer o conteúdo do vídeo, a rede neural funciona de forma semelhante ao reconhecimento de imagem. Uma abordagem para isto é tratar o vídeo como uma série de imagens. No entanto, como apontado por [Ng et al., 2015], essa abordagem não trata as informações temporais entre quadros e pode levar a conceitos irrelevantes emergentes da cena. Por outro lado, uma vantagem deste método é requerer menos tempo de computação para analisar o vídeo.

Outra abordagem é considerar a relação temporal entre os quadros e inferir as *tags* analisando

[4] https://clarifai.com

as relações à medida que o tempo passa. Uma vantagem deste método é que ele diminui a probabilidade de retornar *tags* irrelevantes do vídeo e mantém apenas aquelas que persistiram durante todo o tempo [Ng et al., 2015].

Neste contexto, a API de reconhecimento de vídeo usada neste trabalho fornece a descrição de componentes de cena para cada segundo do vídeo. Como resultado, após a fase de instanciação, os tempos descritos no documento multimídia terão uma granularidade de um segundo, estando compatível com a configuração padrão do módulo intermediário.

O processador de âncoras abstratas foi implementado de forma transparente ao reconhecimento de cena utilizado. Neste trabalho também foram implementadas arquiteturas de redes neurais para o reconhecimento de áudio, vídeo e para o conteúdo audiovisual combinado. No Capítulo V cada uma dessas implementações é discutida em mais detalhes.

IV.4 Caso de Uso

Nesta Seção, é apresentado um cenário de uso para destacar como âncoras abstratas suportam o desenvolvimento de uma aplicação *mulsemedia*. Foi desenvolvida uma aplicação NCL que combina um vídeo com efeitos sensoriais para enriquecer a experiência do usuário. A aplicação chamada “ambientes ao redor do mundo”, consiste em cenas sobre diferentes ambientes que são apresentados ao usuário.

Uma representação da linha de tempo do conteúdo do vídeo e sua sincronização com efeitos sensoriais é apresentada na Figura IV.5. Ela apresenta um conjunto de quadros-chave do vídeo[5] e as três primeiras *tags* reconhecidas nessa parte do vídeo. No momento de cada cena, a aplicação NCL inicia um atuador para executar um efeito sensorial relacionado a essa cena. Na Figura IV.5 também é apresentada uma linha do tempo de execução de efeitos sensoriais esperada para o conjunto de quadros-chave apresentados.

Figura IV.5: Efeitos sensoriais gerados em linha do tempo

Para criar essa aplicação, um autor deve definir em NCL âncoras e *links* para especificar a sincronização de tais efeitos sensoriais. Como forma de nortear a autoria da aplicação em NCL,

[5]Imagens e vídeos são licenciados como Creative Commons CC0 e foram encontrados em Pixabay (https://pixabay.com).

a Tabela IV.1 descreve os efeitos sensoriais a serem sincronizados quando uma determinada *tag* é encontrada no vídeo. Isso varia de efeitos de aroma para efeitos de vento, calor e frio. Os efeitos também variam em intensidade de acordo com os componentes da cena. Deve-se notar que os efeitos podem ser executados ao mesmo tempo. Isso ocorre quando ambas as *tags* representativas de efeitos são encontradas na mídia ao mesmo tempo. Assim, os elementos `area` relacionados a essas *tags* estarão ativos e, como consequência dos links NCL, os efeitos sensoriais estarão ativados também.

Tabela IV.1: Efeitos sensoriais gerados por cada componente de cena

Tag	Efeito Sensorial
summer	vento 50%, calor 50%
snow	frio 100%
forest	aroma de floresta 100%, vento 25%
flower	aroma de flor 100%, vento 25%
storm	vento 100%, frio 50%, umidificador de ar 100%
sea	vento 50%, calor 50%, umidificador de ar 50%
hot	vento 50%, calor 100%

Seguindo as descrições na Figura IV.5 e na Tabela IV.1, um documento desta aplicação foi descrito em NCL usando âncoras abstratas para indicar os componentes da cena de interesse. A Listagem IV.6 apresenta a especificação das âncoras abstratas deste documento.

```
1 <media id="video" src="video.mp4">
2     <area tag="summer"/>
3     <area tag="snow"/>
4     <area tag="forest"/>
5     <area tag="flower"/>
6     <area tag="storm"/>
7     <area tag="sea"/>
8     <area tag="hot"/>
9 </media>
```

Listagem IV.6: Especificação das âncoras abstratas da aplicação “ambientes ao redor do mundo”

Após definir as âncoras abstratas, o autor deve relacioná-las com a execução de efeitos sensoriais. O comportamento da aplicação que dita a ativação de efeitos sensoriais é definido por um grupo de 7 elementos `link` (um para cada âncora abstrata). A Listagem IV.7 apresenta uma especificação de link para a âncora abstrata `summer`.

```
1 <link xconnector="onBeginStartSet">
2     <bind role="onBegin" component="video" interface="summer"/>
```

```
    <bind role="start" component="wind"/>
    <bind role="start" component="heat"/>
</link>
```

Listagem IV.7: Especificação do link para a âncora `summer`

O link apresentado na Listagem IV.7 sincroniza o componente de cena `summer` com os efeitos sensoriais `wind` e `heat`. Neste exemplo, considera-se que ambos os efeitos sensoriais são representados como nós de mídia na aplicação e são representados por *scripts* Lua que controlam os atuadores responsáveis por esse efeito. A Listagem IV.8 apresenta uma definição de *scripts* Lua representando os efeitos de vento e calor. São definidos parâmetros de localização do efeito, que indicam que estão localizadas em todo o ambiente (*:*:*) e parâmetros de intensidade, cujo valor será 50%. Neste exemplo a intensidade é expressa em uma porcentagem da intensidade máxima que o atuador é capaz de fornecer. Detalhes da execução de efeitos por parte destes *scripts* Lua são apresentados no Capítulo VI.

```
<media id="wind" src="wind.lua">
    <property name="location" value="*:*:*"/>
    <property name="intensity" value="50%"/>
</media>

<media id="heat" src="heat.lua">
    <property name="location" value="*:*:*"/>
    <property name="intensity" value="50%"/>
</media>
```

Listagem IV.8: Especificação dos efeitos de vento e calor

Como resultado do uso de âncoras abstratas, há uma diminuição do esforço de autoria. O autor desta aplicação, usando âncoras abstratas, deve declarar 7 âncoras abstratas e 7 links. A aplicação tem um total de 74 linhas de código para descrever seu comportamento. Após o processamento, para um vídeo de 80 segundos, o documento passa a possuir 45 instâncias de âncora e também 45 instâncias de `link`. O documento processado tem um total de 362 linhas de código para executar o mesmo comportamento descrito anteriormente, usando âncoras abstratas. Isto se dá porque os componentes de cena descritos por âncoras abstratas apareceram diversas vezes no conteúdo do vídeo e, consequentemente, cada instância de âncora e link representa o momento de apresentação de um componente de cena.

Como pode ser visto neste exemplo, usando âncoras abstratas, o autor teve que declarar

cerca de 15% do número resultante de âncoras e links e cerca de 20% das linhas de código resultantes. Além disso, sem o uso do AAP, o autor deveria, não só, definir as âncoras e os links, mas também observar cuidadosamente o vídeo para reconhecer os componentes da cena e seu tempo, a fim de descrever as âncoras e sua sincronização com os efeitos sensoriais. Como pretendido, podemos ver uma grande diminuição no esforço de autoria em relação à autoria manual.

Vale ressaltar que o mesmo código descrito usando âncoras abstratas é mantido mesmo no caso em que o tamanho da mídia muda. Dado que as âncoras abstratas não estão diretamente relacionadas ao comprimento da mídia (no tempo), mas apenas aos componentes da cena que ela possui, o código da aplicação não precisa mudar no caso de o tamanho da mídia mudar. Esse resultado também é favorável ao autor, pois o número de instâncias de âncora pode aumentar com o aumento do tamanho da mídia.

De acordo com a Listagens IV.6 e IV.7, com âncoras abstratas o autor precisa definir apenas 5 linhas de código para realizar a sincronização da ativação de efeitos sensoriais. Sendo uma linha para definir a âncora abstrata com o componente de cena que deseja-se reconhecer e 4 linhas para definição de links relacionados a esta âncora abstrata. Sendo assim, se uma âncora abstrata resultar em 10 novas instâncias, serão geradas 50 novas linhas de código. Da mesma forma, se uma âncora abstrata resultar em 100 novas instâncias serão geradas 500 linhas de código. Ou seja, a quantidade de linhas de código gerada cresce de forma linearmente proporcional a quantidade de momentos (não adjacentes) que cada componente de cena aparece.

Como foi feita pouca modificação na linguagem NCL, não foram realizados testes de usabilidade para avaliar o processo de autoria com âncoras abstratas em NCL. Porém um trabalho futuro interessante será avaliar a contribuição de âncoras abstratas no processo de autoria. Principalmente, se o processo de instanciação de âncoras corresponde ao processo manual feito pelo autor. Vale ressaltar que modificações por parte do autor após a etapa de instanciação são esperadas. Pois assim como identificado por Timmerer et al. [2012], a autoria completamente automática de efeitos sensoriais pode ser algo indesejável. Afinal, este é um processo artístico que depende da preferência de autores humanos.

Capítulo V Aprimoramento do Reconhecimento de Componentes de Cena

Redes neurais recentemente alcançaram um desempenho notável em muitos domínios de aplicação, especialmente para reconhecimento de conteúdo audiovisual (i.e., áudio e vídeo) [LeCun et al., 2015]. Isso nos motivou investigar o uso desse modelo de aprendizado para na Seção IV.3.3 apresentarmos um primeiro esforço de reconhecimento de componentes de cena utilizando uma API de reconhecimento apenas para a modalidade de vídeo.

Entretanto, como pode ser visto na Figura V.1, componentes de cena podem estar associados apenas à modalidade de vídeo, como a *tag* `nuvens` na Figura V.1(a). Esses componentes podem também estar associados apenas à modalidade de áudio, como as *tags* `vento` e `trovoada` na Figura V.1(b). Existe ainda o caso em que componentes de cena podem estar associados a ambas as modalidades, como é o caso da *tag* `tempestade` na Figura V.1(c).

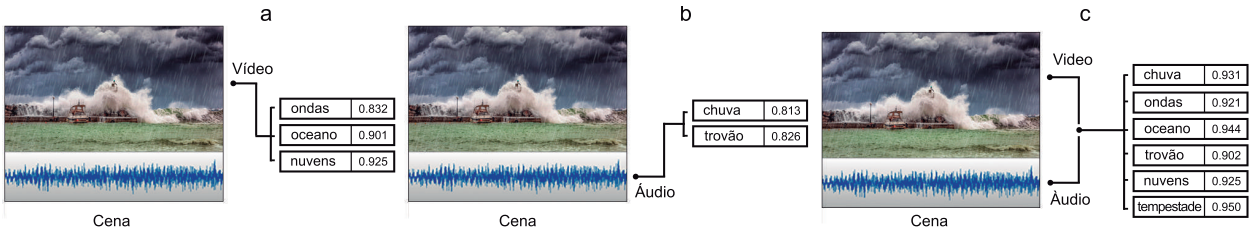

Figura V.1: Resultado de reconhecimento de cena em diferentes modalidades

Neste capítulo é apresentada uma arquitetura de rede neural capaz de reconhecer componentes de cena relacionados nas modalidades de áudio e video combinados, denominada de arquitetura de rede neural bimodal [Abreu et al., 2018]. Para esta tarefa, foi necessária a criação de arquiteturas de redes neurais para reconhecimento de vídeo (Seção V.1) e de áudio (Seção V.2). Na Seção V.3 é apresentada a arquitetura bimodal, que integra as duas arquiteturas anteriores.

A motivação para utilizar uma arquitetura de rede bimodal é que esse tipo de arquitetura tem conseguido desempenho equiparável ou superior a duas redes neurais separadas (uma para cada modalidade), precisamente por tirar vantagem da possível relação entre as duas modalidades que representam um componente de cena. Foram realizados experimentos para validar nossa arquitetura bimodal de aprendizado. Esses experimentos, junto com o conjunto de dados utilizado, são apresentados na Seção V.4.

V.1 Vídeo

Redes Neurais Convolucionais [LeCun et al., 1989] são propostas como forma de reconhecimento de imagem e vídeo. Essas redes são o atual o estado-da-arte para análise de tal conteúdo [Karpathy et al., 2014; Zeiler and Fergus, 2013; Mishkin et al., 2017; LeCun et al., 2015]. De maneira geral, estas redes são constituídas de camadas que realizam operações de aplicação sucessiva (convolução) de filtros em imagens. Cada filtro é responsável por identificar uma característica da imagem (e.g., arestas, ângulos, linhas horizontais). Cada nova camada da rede permite que os filtros detectem características mais genéricas da imagem (e.g., rostos, árvores, rodas). Com o treinamento da rede, os filtros são ajustados para poder identificar certas características da imagem que estejam relacionadas a componentes de cena que se pretende reconhecer [LeCun et al., 1989].

Como mencionado na Seção IV.3.3, existem duas maneiras de realizar tal reconhecimento. A primeira é inferindo componentes de cena para cada quadro do vídeo e a segunda é inferir componentes de cena para um conjunto de quadros do vídeo, levando em conta a relação temporal entre eles. Na arquitetura proposta é levado em conta a relação temporal entre quadros. Este processo é usualmente realizado com a combinação de Redes Neurais Convolucionais com Redes Neurais Recorrentes [Donahue et al., 2015]. Uma Rede Neural Recorrente realiza um processamento sequencial, de modo que consiga considerar a ordenação de *inputs* recebidos para inferir relações à medida que o tempo passa. Em nosso trabalho utilizamos o modelo de rede recorrente chamada LSTM (*Long Short-term Memory*) [Hochreiter and Schmidhuber, 1997].

A combinação de rede neural convolucional com rede LSTM permite um maior desempenho em tarefas de reconhecimento que necessitam de informações visuais ao longo do tempo (por exemplo, vídeos) [Donahue et al., 2015]. As camadas da rede convolucional extraem os recursos de cada quadro e alimentam a rede LSTM. Então, a rede LSTM será capaz de usar a ordenação dos recursos recebidos para aprender sobre o relacionamento de tempo entre eles.

A Figura V.2 apresenta uma visão geral desta arquitetura de aprendizagem. A rede extrai características de sinais visuais contidos em cada exemplo de treinamento (clipe de vídeo). Os vetores de características computados são usados para realizar a predição. A seguir, o processo de extração de características visuais é explicado.

Processamento de vídeo: Primeiramente, para reduzir o espaço do problema, é feita uma sub-amostragem de cada clipe de vídeo para o total de 32 quadros. Também é reduzido o tamanho de cada quadro para 32×32 *pixels* e os valores RGB são mantidos. O resultado é uma matriz de $32 \times 32 \times 32 \times 3$ para cada exemplo de treinamento de vídeo. Por fim, os valores de cada pixel são normalizados para o intervalo $[0, 1]$.

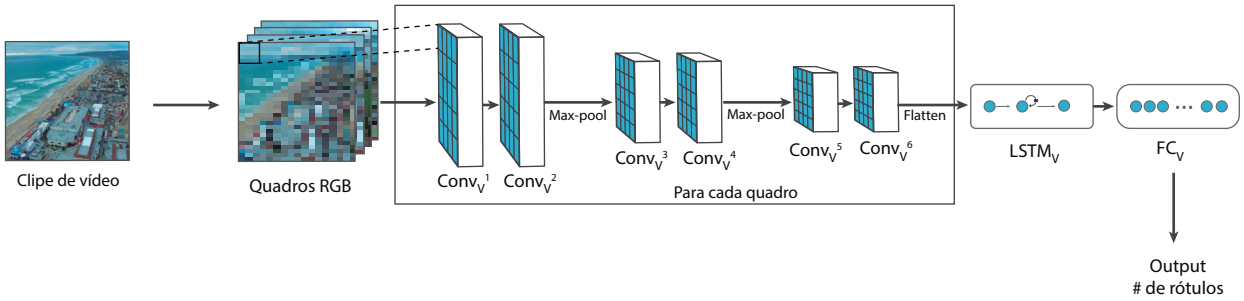

Figura V.2: Arquitetura de rede neural para reconhecimento de vídeo

Arquitetura: A arquitetura de rede usada neste trabalho é motivada por trabalhos relacionados na literatura [He et al., 2016; Mishkin et al., 2017; Krizhevsky et al., 2012]. A rede extrai características de uma sequência de quadros aplicando primeiro três estágios convolucionais ($Conv_V^i$, sendo $1 \leq i \leq 6$, na Figura V.2) em que cada estágio é composto por duas camadas convolucionais idênticas. A saída de cada estágio é normalizada por meio da operação de *batch normalization* [Ioffe and Szegedy, 2015]. Após cada estágio também é aplicada uma operação de *max-pooling* [Kalchbrenner et al., 2014] com dimensões (*shape*) de 2×2 e passo (*stride*) de 2×2 para reduzir a quantidade de parâmetros e computação na rede e, portanto, também evitar sobreajuste da rede para um exemplo de treinamento específico. Seguindo trabalhos na literatura [Krizhevsky et al., 2012], os filtros convolucionais foram definidos com campos receptivos de 3×3. A saída de cada cadamada é preenchida com zeros (*zero-padding*) [Goodfellow et al., 2016] para manter o mesmo tamanho de entrada em todos os estágios. Assim como em [He et al., 2016], a quantidade de filtros no primeiro, segundo e terceiro estágios convolucionais são 32, 64 e 128, respectivamente.

O objetivo desta rede é identificar componentes de cena ao longo de uma escala temporal. Por este motivo, todas as camadas convolucionais são também aplicadas na dimensão de tempo (i.e., para cada quadro na sequência) como forma da rede aprender características espaço-temporais.

As ativações da camada convolucional são serializadas para uma dimensão de 512×1. Esse processo acontece para cada quadro da sequência de entrada. Cada dimensão de quadro resultante é unida aos quadros subsequentes. A dimensão final de todos os quadros é ainda levada a uma camada LSTM com 256 unidades para aprender a estrutura temporal dos quadros. Por fim, esta camada é seguida por uma camada completamente conectada (FC_V) com 1000 unidades. Este processo tem por finalidade padronizar as dimensões de comprimento de características para 1000. Para evitar sobreajuste do modelo, uma taxa de *dropout* [Srivastava et al., 2014] de 50% é aplicada após o primeiro estágio convolucional, após a camada LSTM (LSTM_V) e após a última camada completamente conectada. Finalmente, mapeamos as previsões para os rótulos na camada de saída.

V.2 Som

No contexto de reconhecimento de som, o componente da cena que deseja ser reconhecido são sons do ambiente. No caso, um evento sonoro que possui características acústicas que possam ser identificadas, tal como fogo, vento, explosão, etc.

Embora primariamente utilizadas para reconhecimento visual, as redes neurais convolucionais têm sido aplicadas com sucesso para identificação de som. São aplicadas principalmente para o reconhecimento de fala [LeCun and Bengio, 1998; Abdel-Hamid et al., 2014] e análise musical [Van den Oord et al., 2013; Dieleman et al., 2011]. Recentemente, tais redes também têm sido empregadas para reconhecimentos de sons ambiente [Piczak, 2015b]. Redes convolucionais são capazes de aprender filtros ao longo do tempo e da frequência quando aplicados a entradas do tipo espectrograma [Zhang et al., 2015]. A construção do espectrograma é feito por meio da transformada de Fourier de termo curto [Allen, 1977] que gera uma representação do espectro do som que varia com o tempo. Os espectrogramas são utilizados para gerar uma forma de representação 2D que será enviada para a rede neural convolucional. A variação de frequência ao longo do tempo tem se mostrado uma característica importante para distinguir entre sons no ambiente utilizando redes neurais convolucionais [Salamon and Bello, 2015]. Seguindo a abordagem em [Piczak, 2015b] utilizamos espectrograma na escala mel. Esta escala mel representa frequências compreendidas pelo ouvido humano [Logan et al., 2000].

A Figura V.3 apresenta uma visão geral desta arquitetura de aprendizagem. A rede extrai características de sinais sonoros contidos em cada exemplo de treinamento. Estes sinais sonoros são convertidos para características do tipo espectrograma. Tal espectograma é utilizado como entrada para realizar a predição. A seguir, o processo de extração de características sonoras é explicado.

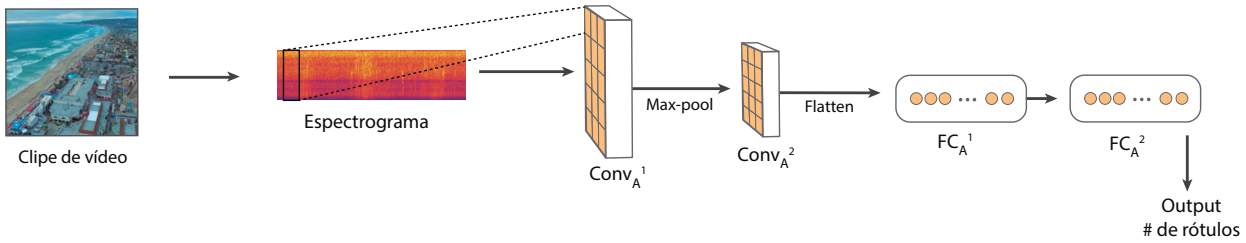

Figura V.3: Arquitetura de rede neural para reconhecimento de em áudio

Processamento de áudio: Neste trabalho foi usada a biblioteca *Librosa* [McFee et al., 2015] para extração de características acústicas representativas de som. Foi empregado um método similar ao apresentado por Piczak [2015b] para transformar o sinal de áudio de cada exemplo de treinamento. Primeiramente, todos os clipes de áudio foram reamostrados para a taxa de $22.050Hz$ e normalizados *min-max* para o intervalo entre -1 e 1. Espectrogramas na escala mel

foram extraídos de todas as gravações. Seguindo a implementação em Piczak [2015b], os espectrogramas foram formados usando a implementação disponível na biblioteca *Librosa* [McFee et al., 2015] com tamanho de janela (*window size*) de 1024, comprimento de salto (*hop lenght*) de 512 e 60 bandas mel. Tais representações de espectrogramas foram convertidos para a escala logarítmica. A principal diferença arquitetural da implementação em Piczak [2015b] é que neste trabalho foram usados os espectrogramas correspondentes a uma entrada de 1 canal em nossa rede, enquanto o autor mencionado adicionou seus deltas, formando uma entrada de 2 canais.

Arquitetura: A arquitetura de rede usada em nosso trabalho é semelhante à utilizada em [Piczak, 2015b]. A primeira camada convolucional ($Conv_A^1$) consiste de 80 filtros receptivos de $57 \times 6 \times 1$. Após a camada é aplicada uma operação de *max-pooling* com uma dimensão de 4×3 e passo de 1×3. Um dropout de 50% é aplicado após esta camada. A segunda camada convolucional ($Conv_A^2$) consiste de 80 filtros com campos receptivos de 1×3. Outra operação de *max-pooling* é aplicada com uma dimensão de 1×3 e passo de 1×3. A operação de *Batch Normalization* [Ioffe and Szegedy, 2015] é aplicada após cada camada convolucional.

O vetor de ativação resultante de $Conv_A^2$ é serializado em uma dimensão de 3600×1 e posteriormente usado como entrada a um estágio completamente conectado com duas camadas (FC_A^1 and FC_A^2). A primeira camada completamente conectada possui 5000 unidades, a segunda camada possui 1000 unidades. Este processo tem por finalidade padronizar as dimensões de comprimento de características para 1000. Uma taxa de *dropout* de 50% é aplicada após cada camada completamente conectada (FC_A^1 e FC_A^2). Finalmente, as previsões são mapeadas para os rótulos na camada de saída.

V.3 Arquitetura Bimodal

Propomos o uso de uma arquitetura de rede neural bimodal [Abreu et al., 2018] para aumentar a precisão do reconhecimento de componentes de cena especificamente para a tarefa de sincronizar efeitos sensoriais em aplicações mulsemedia. Nossa premissa é que combinar as duas modalidades na identificação pode produzir uma melhor precisão quando comparada à identificação de modalidades separadas.

A Figura V.4 apresenta uma visão geral da nossa arquitetura de aprendizagem, que compreende dois componentes: a rede bimodal de extração de características e a rede de fusão. A rede bimodal extrai características de sinais sonoros e visuais contidos em cada exemplo de treinamento (clipe de vídeo). Os vetores de características computados são concatenados e o vetor resultante usado como entrada para duas camadas totalmente conectadas, que realizam a predição.

Nossa arquitetura de aprendizagem bimodal compreende três módulos (retângulos tracejados

na Figura V.4). Os primeiros módulos são as redes individuais apresentadas nas seções acima, porém sem o passo final de predição após suas últimas camadas (FC_V para vídeo e FC_A^2 para áudio). Cada rede irá realizar a extração de características e terá como saída dois vetores com tamanho 1000.

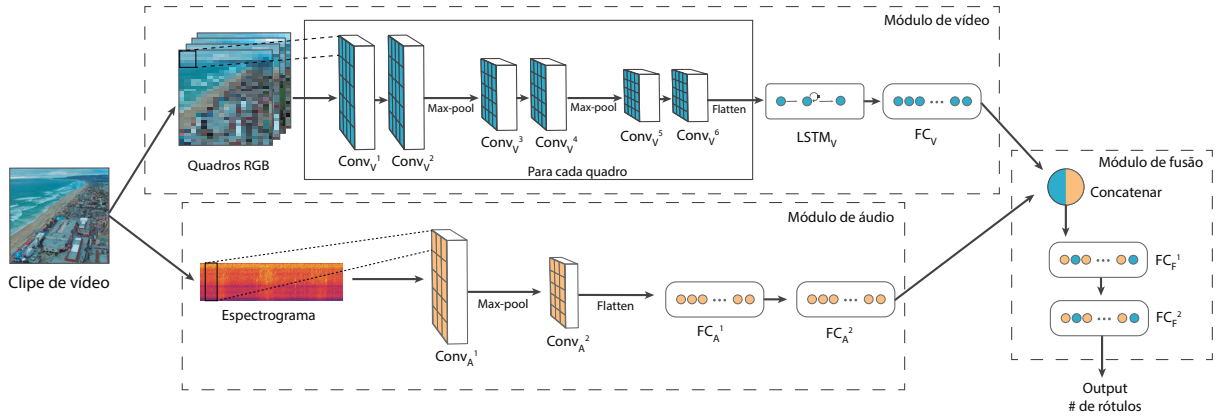

Figura V.4: Ilustração de nossa arquitetura de aprendizado

Arquitetura: A rede de fusão é modelada para executar a tarefa de predição de rótulos tendo como entrada as características identificadas pelos módulos de áudio e vídeo. Para tal, inicialmente concatenamos os dois vetores resultantes dos módulos de áudio e vídeo. Essa concatenação nos deixa com um vetor de características de dimensões 2000×1, que é usado como entrada para o módulo de fusão. O módulo de fusão é uma rede totalmente conectada, que faz a extração de características dos vetores recebidos pelas redes de áudio e vídeo e os relaciona com a previsão dos rótulos. A rede é composta por duas camadas ocultas, cada uma delas com 500 unidades. Uma taxa de 50% de *dropout* é aplicada após cada camada oculta.

V.4 Experimentos

Esta seção apresenta experimentos realizados em um conjunto de dados para validar nossa arquitetura de aprendizado[1]. A Seção V.4.1 descreve o conjunto de dados utilizado e técnicas para sua preparação. A Seção V.4.2 descreve os experimentos conduzidos para validar a arquitetura de aprendizado, junto com análise dos resultados.

V.4.1 Conjunto de Dados

Com o propósito de ajustar os parâmetros de nosso modelo de rede neural bimodal, construímos nosso próprio conjunto de treinamento com base em um conjunto de dados do Google chamado AudioSet [Gemmeke et al., 2017]. O AudioSet é uma extensa coleção de segmentos

[1]O código para construção do conjunto de dados e treinamento de redes pode ser baixado de https://github.com/MLRG-CEFET-RJ/bimodal_audioset

(clipes) de 10 segundos de som que pertencem a vídeos do YouTube. Cada segmento é rotulado de acordo com os componentes de cena de áudio encontrados nele. Audioset contém 632 rótulos e mais de 2 milhões de clipes.

Embora o AudioSet não venha com o conteúdo de vídeo, cada entrada na coleção faz referência ao um número identificador (ID) de vídeo correspondente, cujo conteúdo pode ser acessado pelo YouTube. O AudioSet então fornece um arquivo CSV que vincula o ID do vídeo do YouTube aos rótulos dos componentes de cena encontrados no segmento. Por exemplo, a entrada `7Zdx0YrzHVk,20.000,30.000,‘‘/m/02_41,/m/0838f’’` no arquivo CSV significa que o vídeo do YouTube cujo ID é `7Zdx0YrzHVk`, a partir de `20s` até `30s`, apresenta as classes de “Fire” (`/m/02_41`) e “Water” (`/m/0838f`). Com essas informações, podemos então selecionar os vídeos relacionados a um segmento de som e baixá-los (do YouTube), para cada classe em nosso subconjunto escolhido do AudioSet.

O conjunto de dados de treinamento foi construído a partir do conjunto de treinamento desbalanceado fornecido pelo AudioSet. Este subconjunto foi construído selecionando e baixando segmentos que podem ser associados a efeitos sensoriais. Selecionamos apenas exemplos provenientes dos seguintes 7 rótulos do AudioSet: *Wind*, *Thunder*, *Rain*, *Ocean*, *Fire*, *Explosion* e *Gunshot, gunfire*.

O processo de construção do conjunto de treinamento se aproveita da informação contida no CSV do AudioSet. De posse de informações sobre os sons contidos em cada clipe, nós podemos recuperar vídeos do youtube que também estão associados com estes sons.

No total, foram baixados 10.950 segmentos distintos com um total de 12.829 rótulos. A Figura V.5 mostra a distribuição resultante dos rótulos recuperados para o nosso conjunto de dados de treinamento para os rótulos *Wind* (20.17%), *Gunshot, gunfire* (14.55%), *Ocean* (14.89%), *Rain* (17.1%), *Explosion* (15.05%), *Fire* (9.89%) e *Thunder* (8.35%).

Para construir nosso conjunto de dados de avaliação, extraímos segmentos de todos os clipes mencionados no conjunto de avaliação do AudioSet. O conjunto de avaliação total possui 532 segmentos com 657 rótulos. Este conjunto fornece pelo menos 53 exemplos para cada rótulo.

É importante observar que o Audioset é um conjunto de dados *multi-label*, o que significa que um exemplo de treinamento pode ter vários rótulos associados a ele [Zhang and Zhou, 2014]. Portanto, devido à co-ocorrência de rótulos para cada exemplo de treinamento, o número real de rótulos é maior que os segmentos.

V.4.2 Resultados e Análise

Seguindo abordagens da comunidade [Ruder, 2016], os modelos de redes neurais aqui propostos foram ajustados utilizando o método Gradiente Descendente (*Gradient Descent*) com *Nes-*

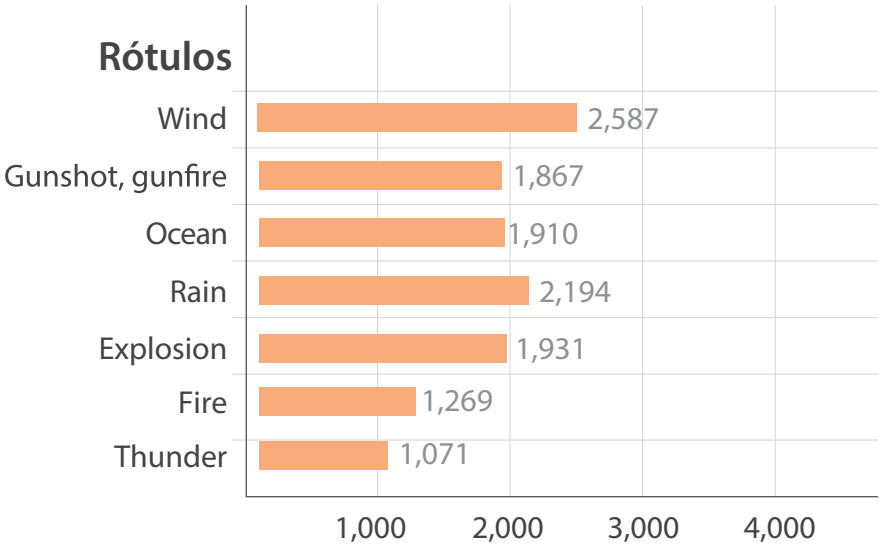

Figura V.5: Distribuição de rótulos em nosso conjunto de treinamento

terov momentum de $0,9$ e taxa de aprendizado de $0,01$. Os modelos foram treinados por 40 épocas com implementação *mini-batch* com tamanho de batch de 32. Nossa arquitetura foi implementada usando Keras [Chollet et al., 2015] com TensorFlow [Abadi et al., 2016] como *backend*. Em todas as arquiteturas, as camadas escondidas foram equipadas com não-linearidades do tipo ReLU [Nair and Hinton, 2010]. Devido a limitações de tempo e de disponibilidade de plataforma experimental, não foi realizada otimização de hiperparâmetros da rede.

Em nossos modelos, mapeamos as predições para os rótulos usando uma camada de saída cujas unidades estão equipadas com a função de ativação sigmóide. A razão desta escolha reside no fato de o conjunto de dados utilizado ser *multi-label*. Desta forma, precisamos produzir uma saída independente para cada rótulo. Por esta razão, decidimos usar a não-linearidade sigmoide na camada de saída, uma vez que, para esta função, as saídas previstas para cada rótulo estão no intervalo contínuo $[0, 1]$, em que um valor próximo de 1 significa a presença do componente correspondente ao rótulo, e um valor próximo de 0 significa sua ausência.

Para validar a arquitetura de aprendizagem proposta, foram investigadas três configurações experimentais diferentes: foram treinados e avaliados os modelos somente em áudio, somente em vídeo e por fim o modelo bimodal. Em cada modelo foi empregada uma função de custo de treinamento *multi-label*, a qual calcula o erro entre os resultados previstos em comparação com os resultados reais [Goodfellow et al., 2016]. O objetivo da rede, portanto, é minimizar o custo total. Para isto foi usada a função *binary cross-entropy* disponível no framework Keras [Chollet et al., 2015]. Esta função é uma representação da função *cross-entropy* para treinamento *multi-label*, adequada para essa configuração de classificação [Lapin et al., 2018]. Na Figura V.6 são mostradas as curvas de aprendizado para todas as configurações. Pode ser visto que o treinamento em nossa arquitetura bimodal resulta em uma convergência mais rápida, ou seja, a rede

alcança um custo menor mais rapidamente.

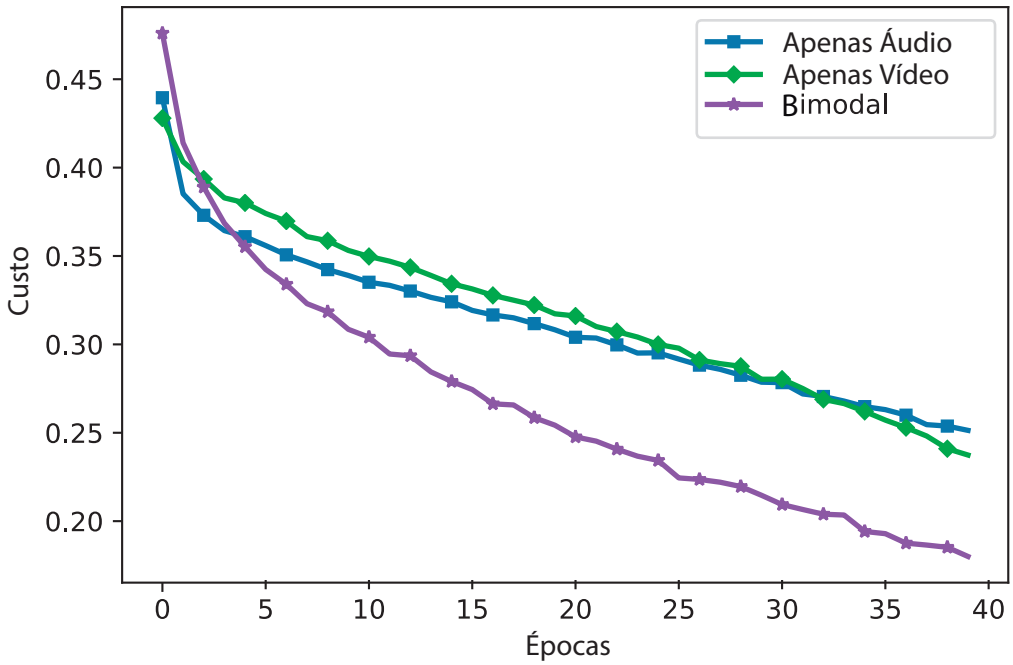

Figura V.6: Curvas de aprendizado para configurações unimodais e bimodais

Para medir a precisão dos modelos treinados, utilizamos a métrica *Hamming Loss*, que representa quantas vezes, em média, um rótulo é incorretamente previsto [Tsoumakas and Vlahavas, 2007]. Essa métrica de avaliação é adequada para a configuração de classificação do AudioSet que é *multi-label*.

A métrica *Hamming Loss* é apresentada na Equação V.1, no qual $|L|$ é o número total de rótulos, e $|D|$ é o número de exemplos no conjunto de avaliação. Nessa equação, x_i é o valor previsto para um dado exemplo, e y_i é o valor de referência correspondente. A operação de disjunção exclusiva (xor) é usada para computar a diferença simétrica de valores para cada rótulo. A operação xor retorna 0 se o rótulo é igual em x_i e y_i. Em caso contrário, essa operação retorna 1.

$$\mathrm{HammingLoss}(x_i, y_i) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{\mathrm{xor}(x_i, y_i)}{|L|} \tag{V.1}$$

Resultados do experimento são apresentados na Tabela V.1. Chamamos a rede treinada apenas com características de áudio de *Apenas Áudio*, a rede treinada apenas em vídeo de *Apenas Vídeo* e por fim a arquitetura de rede bimodal de *Bimodal*. Resultados indicam que a rede Bimodal obteve maior desempenho que as redes unimodais.

Na Seção IV.3 é mostrado que o AAP por padrão assume que os trechos de informação

Tabela V.1: Resultados de avaliação para modelos Unimodais e Bimodal.

Tipo de Modelo	**Hamming Loss**
Apenas Áudio	0.145811
Apenas Vídeo	0.174544
Bimodal	0.123523

temporal das *tags* são de um segundo. Ou seja, o AAP espera que o software de reconhecimento retorne *tags* para cada segundo do conteúdo audiovisual. Desta forma para ter compatibilidade com AAP é necessário que a rede neural seja capaz de realizar reconhecimento de cena para cada segundo.

Embora treinada em um conjunto de dados com exemplos de treinamento de 10 segundos, a arquitetura desenvolvida aqui pode ser aplicada a segmentos de qualquer duração. A princípio redes neurais são robustas suficiente para reconhecer componentes em diferentes variações temporais. De acordo com [Ng et al., 2015], podem ser treinadas e avaliadas com conjuntos de dados que possuem tamanho de exemplos variados. Um exemplo de conjunto de dados é o UCF-101 que tem tamanho de clipes entre ≈ 1s e ≈ 71s [Soomro et al., 2012].

Nos experimentos foi utilizada uma máquina CPU Intel(R) Core(TM) i7-6700 @ 3.40GHz com 32GB de memória e sistema operacional Ubuntu 16.04.4. Ainda, a máquina possui placa de vídeo NVIDIA GeForce GTX 1080 com 8GB de memória. Para realizar a preparação dos conjuntos de treinamento foram necessárias $\approx 10h$. O treinamento de redes durou, em média, 30 minutos. A predição de rótulos do conjunto de validação (total de 532 segmentos) durou $\approx 8ms$ para a rede de áudio, $\approx 36ms$ para a rede de vídeo e $\approx 41ms$ para a rede bimodal.

Capítulo VI Representação de Efeitos Sensoriais em Aplicações Interativas

No Capítulo IV, foi apresentado um exemplo de documento NCL que sincroniza a execução de efeitos sensoriais com trechos de um vídeo. Na apresentação desse exemplo, foi indicado que os efeitos sensoriais a serem apresentados foram representados da mesma forma que um objeto de mídia, ou seja, como um nó da aplicação. Portanto, a proposta de tratar um efeito sensorial em uma aplicação multimídia da mesma forma que um objeto de mídia tradicional tem como objetivo aproveitar a capacidade de NCL definir a sincronização entre nós, sem se preocupar com os detalhes de como eles devem ser apresentados.

Ao longo deste capítulo, a Seção VI.1 descreve como o formatador da linguagem NCL trata a execução de objetos de mídia e os eventos associados a uma mídia. Em seguida, na Seção VI.2 é apresentada a proposta de integração de efeitos sensoriais em aplicações NCL. Esta proposta se baseia no uso de *scripts* Lua que convertem uma descrição NCL para o padrão MPEG-V, bem como em um componente processador que ajusta uma descrição contendo efeitos sensoriais para uma seguindo o padrão NCL. Por fim, na Seção VI.4 é apresentado um simulador de aplicações *mulsemedia*, capaz de compreender descrições de efeitos sensoriais em MPEG-V e simular seu funcionamento em um ambiente tridimensional.

VI.1 Tratamento de eventos em NCL

NCL [ITU, 2009] define um evento multimídia como "*Uma ocorrência no tempo que pode ser instantânea ou ter uma duração mensurável*". Dentre os eventos suportados por NCL, existem os eventos de apresentação, seleção e atribuição. O evento de apresentação indica o estado da exibição de um nó ou de uma âncora. O evento de seleção indica a ocorrência de interação do usuário com um nó ou âncora. O evento de atribuição indica a modificação do valor de uma propriedade de um nó. Todo evento NCL está associado a uma máquina de estados composta dos estados: ocorrendo (*occurring*), pausado (*paused*) e dormindo (*sleeping*). A Figura VI.1 apresenta a máquina de estados de eventos de NCL.

As relações causais de NCL são definidas por meio de conectores, que relacionam transições na máquina de eventos em diferentes participantes da relação (nós/âncoras). Como discutido brevemente no Capítulo IV, a relação definida em um conector é instanciada por elos (links) que

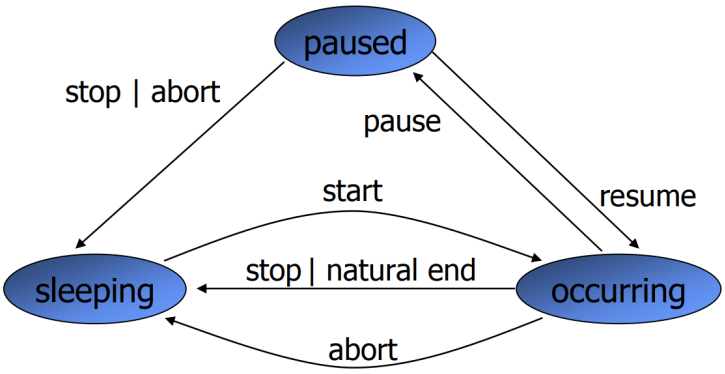

Figura VI.1: Máquina de estados de eventos em NCL. Fonte: [Soares and Barbosa, 2012]

representam relacionamentos específicos entre um conjunto de nós/âncoras. No caso de um evento de apresentação, um autor pode definir links NCL que manipulem a execução de mídias. Por exemplo, podemos ter um link que especifica que quando uma âncora da mídia *A* iniciar, a mídia *B* irá iniciar. Links podem também considerar a ocorrência de interação do usuário, por exemplo, especificando que uma determinada mídia será pausada quando o usuário a seleciona. Por fim, através de um evento de atribuição, um autor pode tanto definir links que são disparados quando uma mudança no valor de uma propriedade acontece ou que mudam o valor de uma propriedade quando algum outro evento acontece.

A linguagem NCL não leva em consideração o tipo e conteúdo de um objeto de mídia ou de qualquer outro nó da aplicação. A arquitetura do formatador da linguagem NCL segue essa mesma filosofia da linguagem. A responsabilidade pela execução de um nó é conferida ao *player* específico capaz de renderizar aquele tipo de nó. Ao formatador, cabe a responsabilidade de capturar eventos que tenham ocorrido em um objeto de mídia, reportados pelos *players*, e disparar novos eventos de acordo com os relacionamentos descritos em um documento.

Cada *player* de mídia deve seguir uma interface padronizada a fim de se interligar ao formatador NCL. Quando a interface do *player* não está de acordo com a definida pelo formatador, um adaptador deve ser usado. De acordo o padrão da linguagem NCL [ABNT, 2011], a interface do formatador com um *player* funciona por meio da troca de eventos, seguindo a máquina de estados de evento de NCL vista na Figura VI.1.

Para ilustrar esse funcionamento, suponha uma aplicação contendo uma mídia de vídeo. Quando a execução da aplicação atinge o ponto em que essa mídia deve iniciar sua apresentação, o formatador faz uma chamada ao *player* disponível capaz de apresentar vídeos, reportando uma transição de *start* do vídeo. O formatador passa ao *player* o conteúdo da mídia (o objeto de vídeo) em questão e informações sobre suas características de apresentação (sua posição, transparência, volume do som, etc) como definidas na aplicação. Dado que o *player* foi capaz de iniciar a apresentação do vídeo, o formatador transita a máquina de estados do evento de

apresentação dessa mídia de *sleeping* para *occurring*. Quando a mídia termina sua execução pela sua duração, o *player* reporta uma transição de *stop* da apresentação ao formatador.

Supondo que durante a apresentação dessa mídia o usuário interaja com ela, o evento de seleção é reportado pelo *player* ao formatador através de transições de *start* e *stop* do evento de seleção. O formatador, por sua vez, transitará a máquina de estados do evento de seleção dessa mídia de *sleeping* para *occurring* e de volta para *sleeping*, dado que eventos de seleção são instantâneos. É importante notar que ao reportar um evento, o *player* também reporta quaisquer informações relevantes a ele relacionadas. No caso da seleção, a tecla pressionada, por exemplo.

O mesmo comportamento ocorre para eventos de atribuição, em que o *player* indica o novo valor para o qual uma propriedade da mídia foi modificada. Quando a mudança de valor parte da aplicação, o formatador passa ao *player* uma indicação da ocorrência de uma transição de *start* do evento de atribuição, junto do novo valor a ser atribuído. Caso a mudança seja possível, o *player* indica o fim da atribuição retornando uma transição de *stop* do evento de atribuição.

A extensão de NCL para atuação em conjunto com a linguagem Lua segue a mesma abordagem, permitindo que um *script* Lua possa ser considerado como um objeto de mídia qualquer da aplicação. Eventos relativos a uma mídia Lua são repassados ao interpretador Lua, que pode ser inicializado, pausado e parado por NCL. Da mesma forma um código Lua pode gerar eventos NCL relativos ao nó Lua, suas âncoras e propriedades, que são reportados pelo interpretador Lua ao formatador NCL.

VI.2 Integrando efeitos sensoriais em aplicações NCL

Aproveitando-se das características de extensão de NCL com Lua, a abordagem seguida neste trabalho é utilizar nós Lua para representar efeitos sensoriais. Esta abordagem é baseada em [Josué et al., 2018], onde um nó de efeito sensorial especifica um conjunto de propriedades que caracterizam esse efeito. Na proposta aqui apresentada, tais propriedades são mapeadas para um *script* Lua que controla a apresentação deste efeito.

As propriedades usadas para caracterizar um efeito sensorial são baseadas naquelas propostas pelo padrão MPEG-V. Em particular são consideradas propriedades referentes aos atributos do elemento `effect` da linguagem SEDL, vistos na Seção II.1. Para renderizar efeitos sensoriais no ambiente, uma descrição SEDL correspondente é gerada e enviada na rede local. Um *Media Processing Engine* (SEDL) na rede local, com base nessa descrição, ativa/desativa atuadores presentes no ambiente que estejam relacionados a esse efeito. A escolha pelo padrão MPEG-V tem por objetivo manter a compatibilidade da proposta com outros *players* de efeitos sensoriais disponíveis na literatura.

A Figura VI.2 ilustra o cenário conceitual da reprodução de efeitos sensoriais integrado à

linguagem NCL. Conforme um nó de efeito sensorial recebe eventos da aplicação, descrições SEDL correspondentes são geradas e enviadas para um *Media Processing Engine*(MPE). Para isso, propriedades da mídia representando um efeito sensorial em NCL são automaticamente mapeadas para especificações MPEG-V por meio de um componente tradutor. Esse componente tradutor é implementado em Lua, permitindo sua utilização em aplicações sem necessitar uma modificação do formatador de NCL.

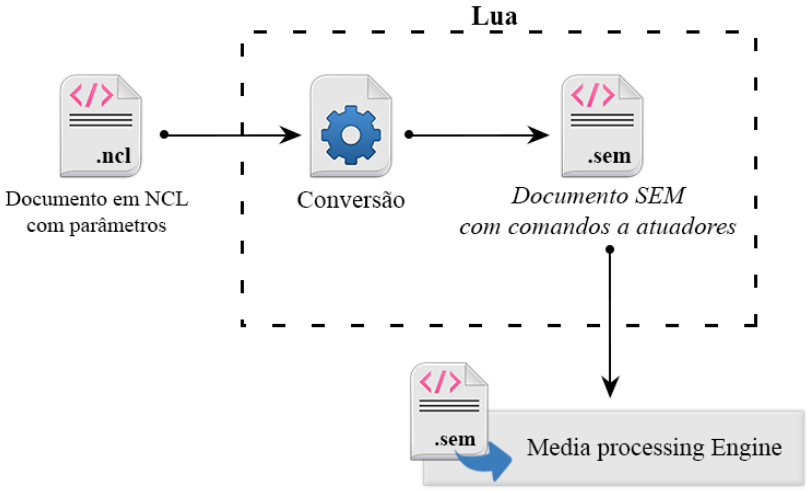

Figura VI.2: Diagrama conceitual de reprodução de efeitos sensoriais em NCL

A Listagem VI.1 apresenta uma descrição de um efeito sensorial em NCL por meio da definição de uma mídia com propriedades. No exemplo, é definido um nó de efeito sensorial de vento com id `vento01`. As características do efeito sensorial são definidas por meio de elementos `property`, que especificam características específicas de um nó de mídia. A propriedade `duration` indica a duração em segundos do efeito. A propriedade `intensity` define a intensidade do vento na escala *beaufort*[1]. A propriedade `location` define quais dos atuadores presentes no ambiente serão usados para renderizar um dado efeito. No exemplo, o efeito deve ser ativado no atuador que esteja na posicionado na frente do usuário, na região central-esquerda.

```
1 <media id="vento01" type="x-sensory-effect/x-windType">
2     <property name="location" value="left:middle:front"/>
3     <property name="duration" value="5.0"/>
4     <property name="intensity" value="3.0"/>
5 </media>
6
7 <link id="efeitoVento01" xconnector="onBeginStart">
8     <bind component="video" interface="ancora01" role="onBegin"/>
```

[1]Uma intensidade também pode ser descrita em porcentagem. Neste caso a porcentagem indicada deverá ser traduzida para um valor na escala *beaufort* entre 0-12.

```
9     <bind role="start" component="vento01"/>
10 </link>
```

Listagem VI.1: Especificação de efeito sensorial de vento em NCL

No exemplo, assume-se a existência de uma mídia `vídeo` que define uma âncora `ancora01`. É especificado um link com identificador `efeitoVento01`, o qual inicia o efeito `vento01` quando a âncora `ancora01` for ativada. Para facilitar a definição de efeitos sensoriais, a linguagem NCL foi estendida de forma que os elementos `media` tenham um novo tipo MIME [Freed and Borenstein, 1996]. Um tipo MIME é uma *string*, padronizada ou não, que define a classe da mídia e seu tipo de codificação. Os tipos MIME padronizados são controlados pela IANA (*Internet Assigned Numbers Authority*), já os não padronizados são definidos por acordo comum. Em geral, um tipo MIME não padronizado tem um prefixo `"x-"` [Freed and Borenstein, 1996]. Neste trabalho é proposto um tipo MIME que represente a classe de mídias de efeito sensorial, por meio da *string* `x-sensory-effect`. Subtipos de efeitos sensoriais são especificados como o tipo de codificação de uma mídia. Cada tipo de efeito segue a descrição do vocabulário SEV do padrão MPEG-V, visto na Seção II.1.1.

No exemplo apresentado na Listagem VI.1, uma mídia é declarada como sendo um efeito de vento, representado pelo tipo `x-sensory-effect/x-windType`. É importante notar que nesta descrição de efeito não é necessária a definição do atributo `src`, pois posteriormente um *script* Lua que controla a apresentação deste efeito sensorial será gerado automaticamente. Este *script* realizará a tradução dos atributos da mídia para comandos SEDL equivalentes. Por simplicidade, chamaremos esse *script* de TES (Tradutor de Efeitos Sensoriais). Tal tradução será descrita em mais detalhes na Seção VI.2.1.

Seguindo essa abordagem orientada a eventos, não é necessário um atributo de tempo de início para o efeito, visto que o início de um efeito sensorial é dado pela ocorrência de uma transição na máquina de estados do evento de apresentação, conforme a máquina de estados na Figura VI.1. No exemplo da Listagem VI.1, o efeito ativado se manterá ativo pelo tempo especificado em sua propriedade `duration`, caso especificado, ou, caso contrário, por tempo indeterminado, até receber um comando de desativação.

No contexto da descrição de um efeito sensorial na linguagem SEDL, um comando de ativação é um comando que atribui o valor `true` para o atributo `activate` de um elemento `effect`, enquanto um comando de desativação atribui o valor `false` para o atributo `activate` de um elemento `effect` que englobe a posição onde o efeito atual está executando. No caso do exemplo, tanto os comandos de ativação e desativação utilizarão a localização `left:middle:front`.

Uma vez disparado o link `efeitoVento01` da Listagem VI.1, o evento de *start* da apresentação

do nó `vento01` será encaminhado para o TES. A Listagem VI.2 apresenta o código SEDL gerado pelo TES ao receber esse evento.

```
<Effect type="windType" activate="true" location="left:middle:front"
   intensity-value="3.0"/>
```

Listagem VI.2: Definição de efeito de vento de acordo com SEDL

De acordo com a linguagem SEDL, duas informações temporais podem ser definidas: *Duration* e PTS. O atributo *Duration* define a duração de um efeito, neste caso o efeito ficará ativo durante um tempo estipulado. Enquanto PTS define o tempo de início de um efeito.

De acordo com essas definições temporais, três cenários para desativação do efeito sensorial apresentado na Listagem VI.2 são possíveis. (i) Atribuir uma duração para o efeito através do atributo `duration`. (ii) Definir um segundo efeito com a mesma informação de localização do efeito ativo, com os atributos `activate` igual a *false* e `pts` equivalente a duração do efeito. (iii) Realizar um novo comando para definir um efeito com a mesma informação de localização do efeito ativo e com o atributo `activate` igual a *false*.

Dado que se segue uma abordagem orientada a eventos, caso uma informação de duração seja adicionada, o efeito passará a contar seu tempo ativo a partir do momento que foi iniciado e deverá parar assim que sua duração chegar ao fim. O efeito definido na Listagem VI.2 não possui qualquer informação temporal, sendo seu término definido por um comando de desativação a ser enviado em um momento posterior. Dessa forma, o TES tem maior controle sobre a apresentação do efeito e sua máquina de estados de apresentação.

Em um cenário orientado a eventos, caso um reprodutor receba um efeito com o atributo PTS, isto resultará em um atraso na ativação equivalente ao tempo de PTS. Por outro lado, NCL permite a definição de atraso na ativação de *links* através do atributo *delay*. Na qual o formatador NCL só gera a transição após aguardar o tempo especificado no atributo *delay*. Por conta da integração do formatador NCL com o interpretador Lua, a informação de *delay* não é passada pelo formatador NCL para o interpretador Lua. O que ocorre é que o formatador aguarda o tempo definido em *delay* antes de enviar a transição de *start* para o interpretador. Portanto, para passar uma informação de *delay* para o TES transformá-la em PTS seria necessário alterar os *links* do documento incluindo uma ação de *set*, para cada transição contendo um *delay*. Essa ação de *set* seria usada para passar o valor do *delay* antes de ser feito o *start* na âncora do TES. O mesmo problema é observado quando se faz uma atribuição com duração em NCL. Os valores dos atributos *duration* e *by* que especificam a atribuição com duração não são passados para o interpretador Lua. Por conta de tais características, na abordagem aqui proposta, tanto atributos que especificam o *delay* de uma ação, quando uma atribuição com duração, não são traduzidos

pelo TES para se equivalente em SEDL. Espera-se que tais características sejam definidas na própria aplicação NCL.

É importante notar que a descrição de mídia de efeito sensorial com propriedades apresentadas na Listagem VI.1 não é uma descrição padrão NCL. A tradução de eventos NCL para efeitos SEDL é feita em uma sequência de etapas que visam (i) adequar o documento NCL ao padrão da linguagem, bem como (ii) instanciar um TES genérico para um determinado efeito. A Figura VI.3 apresenta os passos dessa tradução.

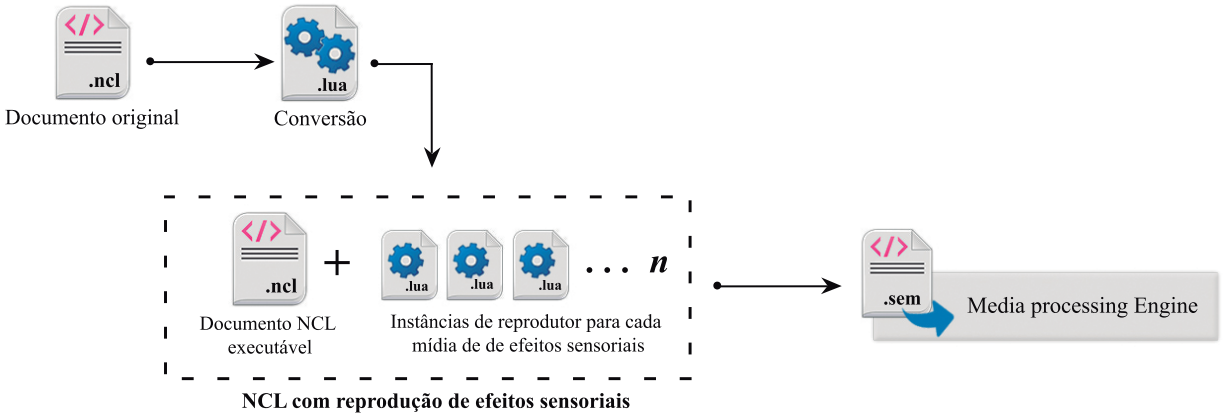

Figura VI.3: Cenário de conversão de uma aplicação NCL para comandos SEDL

Na figura, um documento NCL é criado com relações entre mídias audiovisuais e nós que representam efeitos sensoriais. Antes da execução, o documento original deve ser processado em um documento final seguindo o padrão NCL. Este processamento gera instâncias do TES para cada mídia de efeitos sensoriais no documento. Durante a reprodução do documento NCL, cada instância do TES fica responsável por um efeito sensorial. As instâncias fazem a tradução em tempo de execução para a linguagem SEDL e enviam a informação para um MPE presente na rede local. As seções a seguir descrevem em mais detalhes os passos de processamento e tradução de efeitos para SEDL.[2]

VI.2.1 Etapa de processamento

Provido de um documento NCL contendo mídias representando efeitos sensoriais, o processador coleta as informações referentes às propriedades desses efeitos sensoriais visando instanciar o TES genérico para esse efeito. Ainda, por conta de decisões de projeto da implementação do formatador NCL, alterações nos nós e links do documento também são necessárias. O processo de tradução é feito em três etapas: (i) gerar automaticamente instâncias do TES genérico, (ii) realizar alterações de nós no documento para integrar a mídia com a instância do TES e (iii)

[2]O código para processador e TES pode ser acessado em https://github.com/GPMM/TES

realizar alterações nos links no documento para redirecionar eventos para o TES. Os parágrafos a seguir apresentam cada uma dessas etapas em detalhes.

Na primeira etapa é feito o *parsing* do documento, identificando as mídias que definem efeitos sensoriais. A identificação é feita verificando aquelas que possuem no atributo `type` a informação MIME `x-sensory-effect`. As propriedades dessas mídias, bem como a informação do tipo de efeitos sensorial contido no atributo `type` são coletadas gerando uma tabela que armazena esses valores no TES gerado para uma dada mídia. A Listagem VI.3 apresenta um trecho do TES para a mídia *vento01* apresentado na Listagem VI.1.

```
1 local tr = require('SEtranslator')
2
3 local effect = {
4   type = 'windType'
5   location = 'left:middle:front',
6   duration = 5,
7   intensity = 3
8   }
9 local t = tr:new(effect)
```

Listagem VI.3: Instância de tradutor de efeitos

Na Listagem VI.3 as linhas 3-8 representam as propriedades do efeito sensorial. Essas propriedades foram coletadas diretamente do documento NCL na etapa de *parsing*. As linhas 1 e 9 mostram o carregamento e instanciação da classe `SEtranslator` com as propriedades de efeito sensorial.

Como discutido no início dessa seção, o *script* responsável pela exibição de uma determinada mídia é instanciado quando essa mídia inicia sua apresentação. Isto significa que o interpretador Lua somente irá executar o *script* quando a mídia que representa esse *script* sofrer um *start*. Portanto, qualquer evento que possa ocorrer com essa mídia não é capturado enquanto esta não é iniciada. Nas etapas seguintes, o documento NCL é ajustado de forma a fazer com que o TES seja iniciado assim que a aplicação for iniciada. Dessa forma, eventos realizados sobre o efeito sensorial durante a execução da aplicação podem ser recebidos pelo TES. Ainda, é preciso diferenciar eventos que são realizados sobre o efeito daqueles realizados sobre o TES, ou seja, iniciando-o quando a aplicação inicia e terminando-o quando ela termina.

Na segunda etapa, o nó que representa o efeito é modificado para referenciar o TES gerado. O atributo `src` é inserido e como valor para esse atributo é informada a localização do TES. O atributo `type` é removido, visto que a identificação do tipo de efeito será feita pelo TES. O resultado

da modificação é apresentado na Listagem VI.4.

```
<media id="vento01" src="vento01_windTypePlayer.lua">
    <area id="effectInterface"/>
    <property name="location" value="left:middle:front"/>
    <property name="duration" value="5.0"/>
    <property name="intensity" value="3.0"/>
</media>
```

Listagem VI.4: Resultado do ajuste da mídia *vento01* durante o processamento do documento

Na Listagem VI.4 é possível notar que uma âncora foi adicionada à mídia *vento01*. A âncora `effectInterface` é utilizada para diferenciar eventos que devem ocorrer sobre o efeito sensorial daqueles que devem ocorrer sobre o TES. Dessa forma, eventos que ocorram sobre o nó de efeito sensorial, são redirecionados para essa âncora durante o ajuste dos links.

Na terceira etapa do processamento, são feitas as mudanças nos links do documento que possuam algum *bind* fazendo referência à mídia de efeito sensorial. *Binds* associados a eventos de apresentação são redirecionados para a âncora `effectInterface`, adicionando-se o atributo *interface* no *bind* e apontando para essa âncora. Binds associados a eventos de atribuição e seleção permanecem inalterados. Por fim, uma nova porta é adicionada no documento fazendo com que o TES possa ser iniciado assim que a aplicação começar sua apresentação. O resultado é apresentado na Listagem VI.5.

```
<port id="pvento01" component="vento01"/>

<media id="vento01" src="vento01_windTypePlayer.lua">
    <area id="effectInterface"/>
    <property name="location" value="left:middle:front"/>
    <property name="duration" value="5.0"/>
    <property name="intensity" value="3.0"/>
</media>

<link id="efeitoVento01" xconnector="onBeginStart">
    <bind component="video" interface="ancora01" role="onBegin"/>
    <bind role="start" component="vento01" interface="effectInterface"
   />
</link>
```

Listagem VI.5: Resultado do ajuste do link *efeitoVento01* durante o processamento do documento

Na Listagem VI.5, o link agora efetua uma transição de *start* na interface `effectInterface` (linha 12), que por sua vez vai determinar a execução do efeito no *script* Lua.

A Listagem VI.6 apresenta o *script* completo do TES. Nele é definida uma função *handler* que fica responsável por receber eventos de NCL (linhas 11 a 24). Se o tipo de evento for do tipo apresentação (`presentation`) e esta ação seja sobre a âncora `effectInterface` então o TES irá iniciar o processo de tradução das propriedades e envio do comando (linha 15). Caso o evento seja de atribuição (`attribution`), ou seja, uma alteração de valores, então o TES atualiza o valor da propriedade alterada na tabela `effect` (linhas 19 a 21). As tarefas de tradução realizadas pelo TES são apresentadas em mais detalhes na seção a seguir.

```
1  local tr = require('SEtranslator')
2
3  local effect = {
4    type = 'windType'
5    location = 'left:middle:front',
6    duration = 5,
7    intensity = 3
8  }
9  local t = tr:new(effect)
10
11 function handler(evt)
12   if (evt.class ~= 'ncl') then return end
13   if (evt.type == 'presentation') then
14     if (evt.label == 'effectInterface') then
15       p:translate(evt.action)
16     else
17       -- iniciou/parou o script Lua
18     end
19   elseif (evt.type == 'attribution') then
20     effect[evt.name] = evt.value
21   end
22 end
23
24 event.register(handler)
```

Listagem VI.6: Instância de tradutor de efeitos sensoriais com comunicação NCL

VI.2.2 Tradução de efeitos para SEDL

O TES é o componente central na integração de efeitos sensoriais com aplicações NCL. Nele estão funções de tradução dos atributos NCL em efeitos SEDL e comunicação com o MPE. O TES também mantém a máquina de estados do NCL, permitindo que cada transição aconteça somente se o efeito estiver no estado correspondente.

Um diagrama de sequência que caracteriza a comunicação entre NCL, tradutor e MPE é apresentado na Figura VI.4. Na figura, para ativar um efeito sensorial, um link NCL age sobre a âncora *effectInterface* que por sua vez inicia o passo de tradução de propriedades desse efeito, gerando um SEDL.

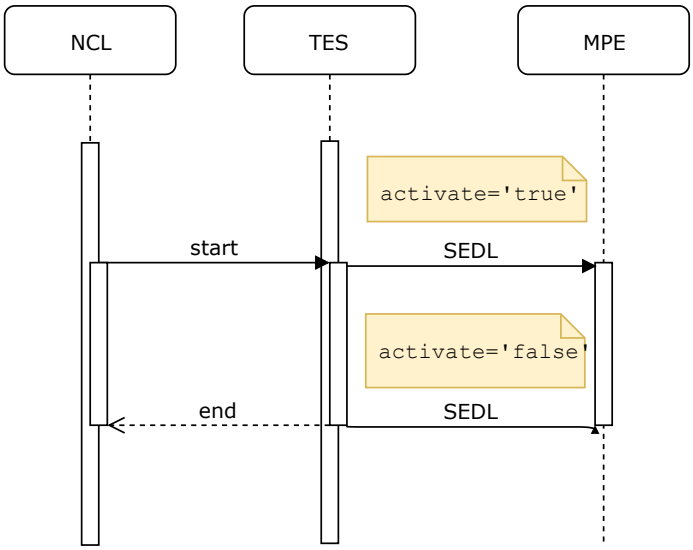

Figura VI.4: Diagrama de sequência de reprodução de efeitos com duração

No exemplo da Figura VI.4 o efeito iniciado possui uma duração de 5 segundos. Dado que é iniciada a apresentação do efeito, o SEDL gerado terá o atributo `activate="true"`. Em NCL a âncora *effectInterface* ficará como *occurring*. O *script* TES também manterá esse mesmo valor, evitando que mudanças que não correspondem a máquina de estados de NCL sejam feitas. Dado que o efeito possui uma duração, o TES manterá uma contagem do tempo em que esse efeito está ativo. Quando sua duração chega ao fim, o SEDL gerado é enviado, mas agora tendo o atributo `activate="false"`. A seguir o TES irá reportar para NCL o término do efeito sinalizando uma transição de *stop* sobre a âncora *effectInterface*. Desta forma, a aplicação em NCL pode ativar quaisquer *links* que dependam do término desse efeito.

A Figura VI.5 apresenta de ativação de um efeito sem duração especificada. Neste caso o SEDL será gerado pelo tradutor e será enviado para o *MPE* da mesma forma anterior. Porém desta vez por faltar uma informação temporal, o TES não realizará o término automático desse efeito. A desativação deste efeito deverá vir de links NCL que farão o TES criar um efeito com o

atributo `activate="false"` e o enviar para a *MPE*.

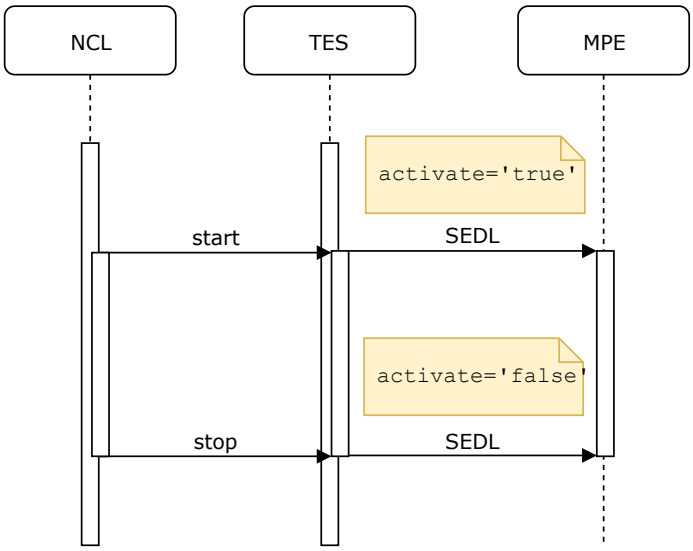

Figura VI.5: Diagrama de sequência de reprodução de efeitos sem duração

É importante notar, que essa mesma situação pode ocorrer quando um efeito é parado antes do término da sua duração. Nesse caso, a contagem da duração do efeito é interrompida e o TES cancela o envio de qualquer informação pendente para o MPE, como, por exemplo, um comando de desativação do efeito previamente enviado.

A Figura VI.6 apresenta um caso em que um efeito tem sua apresentação parada e, posteriormente, continuada. Isso ocorre devido a recepção de ações de *pause* e *resume* para o efeito, vindos da aplicação NCL.

Uma vez recebido um comando de *pause*, o TES irá enviar para o MPE um efeito com o atributo `activate="false"`, parando a execução do efeito. Adicionalmente o TES irá alterar o estado do efeito para *paused* e parará a contagem da duração desse efeito. É importante ressaltar que a parada na contagem de tempo pelo TES fará com que o envio de qualquer informação pendente seja congelado, até que o efeito volte a ser executado. A recepção de um comando de *resume* posterior, fará com que o TES envie para o MPE um novo efeito, agora com `activate="true"`, e retorne a contagem da duração.

Como pode ser observado a partir da descrição do TES apresentada nesta seção, o TES mantém uma fila de efeitos SEDL a serem enviados para o MPE. Cada efeito SEDL nessa fila possui um contador que determina o momento em que este será enviado para o MPE. Por exemplo, dado que o efeito sensorial seja definido em NCL com uma duração, o SEDL que produz sua parada será colocado nessa fila, com um contador iniciado com a duração especificada em NCL. Essa abordagem evita que o MPE execute mudanças no ambiente que não refletem o especificado no documento. Por exemplo, se o MPE solicitar o término de um efeito que já foi parado, isto pode causar o término de algum outro efeito que esteja atualmente executando naquela posição.

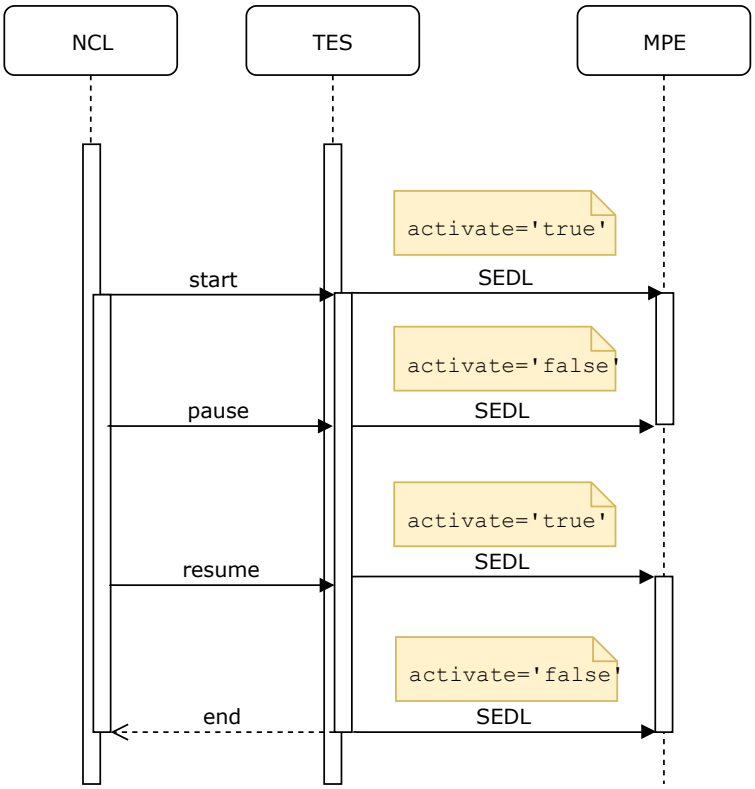

Figura VI.6: Diagrama de sequência de reprodução de efeitos com *pause* e *resume*

VI.3 Posicionamento de Efeitos Sensoriais

A parte 3 do padrão MPEG-V possui um método de posicionamento de efeitos sensoriais baseados em pontos nos eixos X, Y e Z de um espaço tridimensional. Um ponto é representado como a concatenação de 3 palavras que indicam a localização de ativação de um efeito. Por exemplo, a posição `center:top:right` indica que um efeito será renderizado em um ponto na área central da sala, no topo a direita. O símbolo "`*`" pode ser usado para se referir a um conjunto de localizações. Por exemplo, a posição `*:top:right` indica que o efeito será renderizado em todos os atuadores que estejam no topo da sala a direita. Na Figura VI.7 é mostrado um ambiente com todas as posições dentro do padrão de localização do MPEG-V. Na figura, cada ponto vermelho indica uma posição possível para um efeito de acordo com o padrão.

O método de posicionamento adotado pelo padrão MPEG-V pode não ser ideal para representar pequenas variações de posicionamento de efeitos ou até mesmo para realizar animações em que um efeito muda de posição ao longo do tempo. Para sanar esse problema, é proposta uma extensão da descrição de posicionamento do MPEG-V para contar com o posicionamento de acordo com um sistema de coordenadas esféricas. Em um sistema de coordenadas esféricas, pode-se definir qualquer outro ponto em uma esfera ao escolher valores de ângulos para apenas 2 eixos, um eixo horizontal (φ) e um vertical (θ). Adicionalmente também é possível especificar uma área de um efeito. Esta área é a indicação de um retângulo no ambiente no qual os efeitos

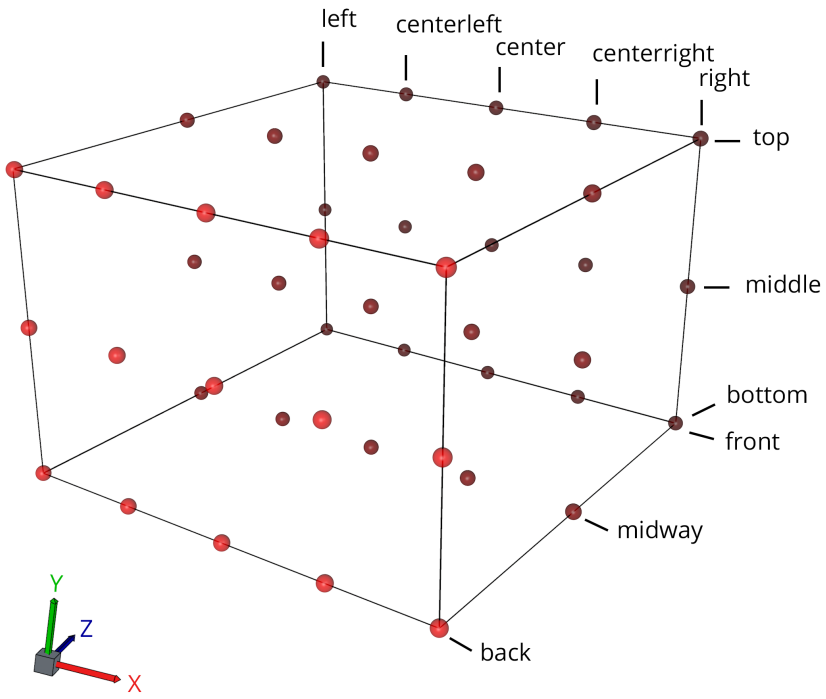

Figura VI.7: Pontos de posicionamento no ambiente de acordo com o padrão MPEG-V

poderão ser ativados. A especificação da região de um efeito compreende sua altura `h` e largura `w`, em graus.

A Figura VI.8 ilustra um cenário com definição de efeito sensorial por coordenadas esféricas. Na figura, o efeito está localizado no ponto `P`, que é definido pela angulação $[-45°, 45°]$. O efeito compreende uma área de ativação de $5°$ no eixo φ e $5°$ no eixo θ a partir do ponto `P`, ou seja, em todos os atuadores que estejam de $-45°$ até $-50°$ na angulação horizontal e de $45°$ até $40°$ na angulação vertical.

É importante notar que a escolha por um sistema de coordenadas esféricas facilita a definição de animações da posição de efeitos sensoriais. A movimentação de um efeito nesse sistema é uma mudança linear no valor de uma, ou das duas, coordenadas angulares. A Listagem VI.7 apresenta dois efeitos de vento definidos usando coordenadas esféricas. O primeiro efeito é localizado no ponto $[-45°, 45°]$ (φ, θ) e têm uma área de ativação de $[5° \times 5°]$ a partir do ponto. Na linha seguinte o efeito é desativado no tempo $27000pts$, usando a localização $[-40°, 40°]$ com uma área de $[15° \times 15°]$. Note que esta localização engloba a posição do efeito definido na linha anterior e resultará na desativação do mesmo.

```
1 <Effect type="WindType" activate="true" location="−45,45,5,5" intensity−value="3.0"/>
2 <Effect type="WindType" activate="false" location="−40,40,15,15" pts="27000"/>
```

Listagem VI.7: Definição de efeito de vento utilizando coordenadas esféricas

O sistema de coordenadas esféricas apresentado permite usar ângulos para definir a des-

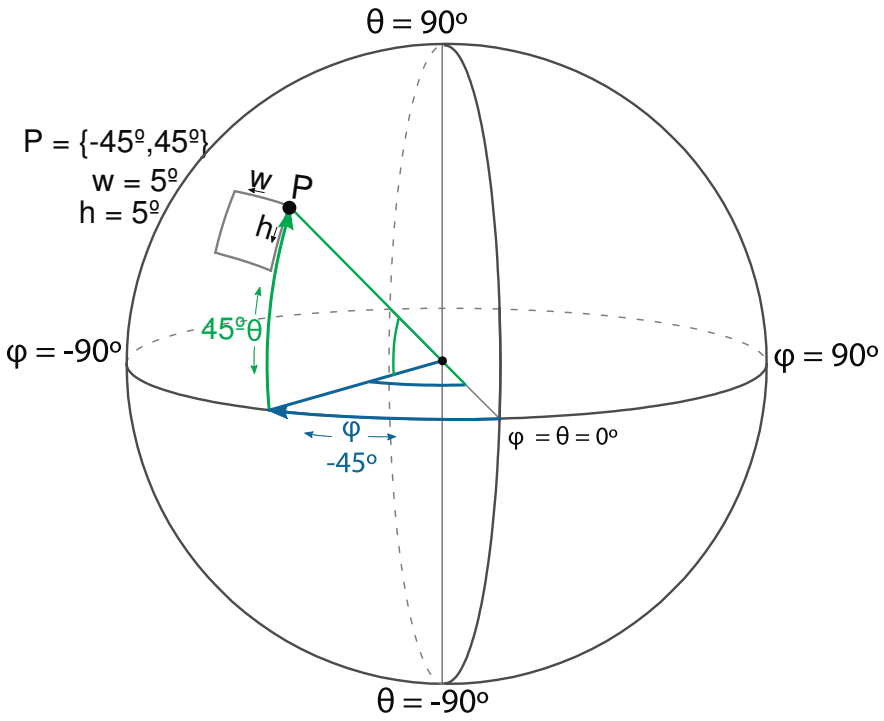

Figura VI.8: Sistema de coordenadas esféricas para posicionamento de efeitos sensoriais

crição do posicionamento de atuadores e efeitos sensoriais em uma direção nos eixos horizontal e vertical. Uma possível consequência negativa deste sistema é que não podem ser representados dois atuadores que estão numa mesma direção, porém em distâncias diferentes. Por exemplo, caso quiséssemos descrever o ponto`centerleft:middle:midway`, utilizamos a angulação $[-45°, 0°]$. Porém, esta angulação também é utilizada para descrever o ponto `left:middle:midway`. Uma forma de resolver este problema é adicionar, além da angulação horizontal e vertical, uma distância do ponto central até o atuador. Esta distância é normalmente denominada *raio* (r) em um sistema de coordenadas esféricas.

VI.4 Simulador

O principal objetivo do simulador é permitir a prototipação rápida de aplicações multimídia com efeitos sensoriais. Com o uso do simulador, o autor de uma aplicação pode testar sua aplicação interativa em tempo real sem precisar organizar um ambiente físico de testes. Ainda, a simulação permite que um mesmo documento possa ser testado em ambientes com diferentes configurações.

A abordagem seguida para implementação do simulador é baseada no cenário de reprodução de efeitos sensoriais em NCL apresentada neste capítulo. O simulador, portanto, recebe um arquivo SEM pela rede e reage aos comandos contidos nesse arquivo em tempo de execução. Além desta função, o simulador permite a adição de novos atuadores e movimentação de atuadores

existentes em um ambiente tridimensional.

O simulador é baseado em software de modelagem 3D[3] existente. A utilização de softwares de modelagem 3D facilitam ao autor organizar objetos físicos no ambiente, pois eles suportam o paradigma *What You See is What You Get* (WYSIWYG) de criação de cenas [Hanrahan and Haeberli, 1990], tornando possível prototipar ambientes complexos em um período curto de tempo. Tais softwares também suportam a adição de funcionalidades por *scripts*. Por meio desta funcionalidade nós estendemos o software para implementar a simulação de atuadores de efeitos sensoriais.

Para realizar uma simulação é necessário um ambiente tridimensional configurado com atuadores. Tal ambiente foi desenvolvido, como pode ser visto na Figura VI.9. O ambiente consiste em um modelo simplificado de uma sala, com um ponto central e objetos dispersos que representam os atuadores. Na figura, os atuadores são distribuídos nas paredes e teto da sala. Atuadores são representados por círculos pretos com linhas pretas em forma de cone. O cone de um atuador indica sua direção de atuação. No simulador proposto, a direção de atuação dos atuadores é sempre o ponto central do ambiente, que é representado como um quadrado com hastes azul, vermelha e verde. Estas hastes representam, respectivamente, os eixos X, Y e Z. Na Figura VI.9 também pode ser vista a interface do software 3D utilizado. A direita da figura (retângulo a) pode ser observado o painel de objetos, neste painel estão todos os objetos relativos ao cenário, incluindo todos os modelos 3D do cenário (e.g., sofá e TV) e todos os atuadores. Para criar um novo atuador é necessário apenas duplicar um atuador existente e movimentá-lo para outra posição no ambiente 3D.

VI.4.1 Uso do Simulador

O simulador é inicializado ao carregar o ambiente no software 3D. Este ambiente contém os modelos de sala e atuadores (com restrições aplicadas) e contém código de simulação. A execução de uma simulação pode ser dividida em três etapas: (i) o simulador recebe um arquivo SEM via rede, (ii) realiza a leitura do arquivo recebido e inicialização de variáveis para simulação, e (iii) executa os comandos de ativação/desativação nos atuadores correspondentes. Os parágrafos a seguir explicam em mais detalhes os passos de simulação.

A comunicação entre a aplicação e o simulador é feita pela rede utilizando o protocolo UDP (*User Datagram Protocol*). Escolheu-se fazer a comunicação entre a aplicação NCL e o Simulador via rede por permitir que aplicações desenvolvidas em qualquer linguagem de programação possam acessar o Simulador, bastando apenas ter suporte à criação de “socket’s UDP”. Uma vantagem desta abordagem é que aplicações podem ser executadas no mesmo computador do

[3]Maxon Cinema 4D : https://www.maxon.net/cinema-4d

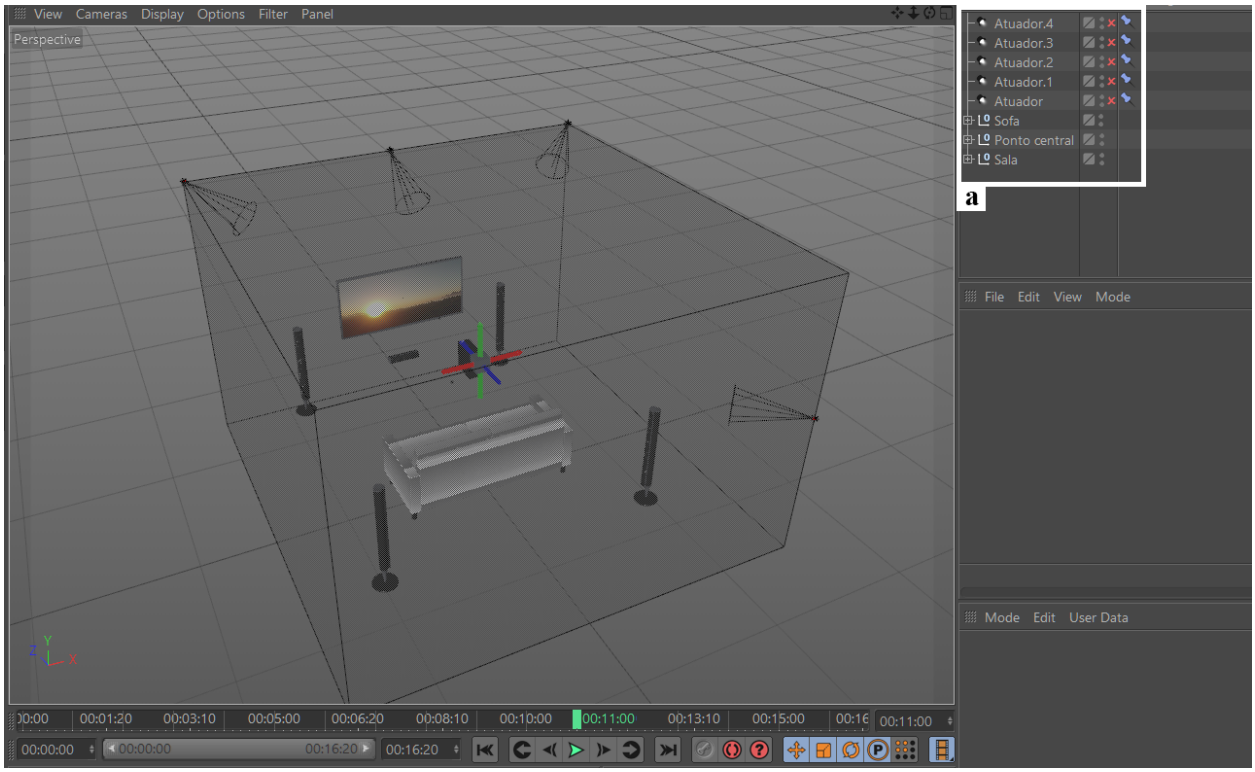

Figura VI.9: Visão do ambiente 3D e interface do software

simulador ou até em diferentes máquinas conectadas pela rede. O protocolo UDP foi selecionado para tornar a transmissão de um arquivo SEM mais rápida, proporcionando assim um menor tempo de resposta da simulação.

Ao receber um arquivo SEM o simulador faz o *parsing* do XML e guarda variáveis relativas à apresentação temporal, espacial e de intensidade dos efeitos. Os atributos temporais que o simulador identifica são `pts` e `Duration`. O atributo espacial identificado é o `Location` que pode ser descrito tanto de acordo com o padrão MPEG-V ou utilizar a extensão para coordenadas esféricas proposta neste trabalho. O simulador também identifica o atributo `intensity-value` para especificar a intensidade de como o efeito deve ser representado no ambiente tridimensional. Este valor é traduzido para a intensidade da cor do atuador. A Figura VI.10 apresenta a interface do simulador em execução. Os atuadores ativos são destacados em vermelho. As intensidades de seus efeitos estão em 100%, ou seja, os atuadores estão com sua máxima cor.

Visando maior compatibilidade com descrições de efeitos sensorias em MPEG-V disponíveis na literatura, o simulador suporta a reprodução de linha do tempo da seguinte forma. Caso o efeito sensorial tenha o atributo `pts`, o simulador irá criar um contador com o valor desse atributo e seguir sua execução até o fim desse contador. Nesse momento o efeito será ativado. Ainda, caso o efeito tenha informações de duração pelo atributo `duration`, um novo contador será criado para controlar a duração do efeito. No momento que o contador chegar a 0, o efeito será desativado.

As informações espaciais são lidas e tratadas de acordo com padrão utilizado. Ao ler a espe-

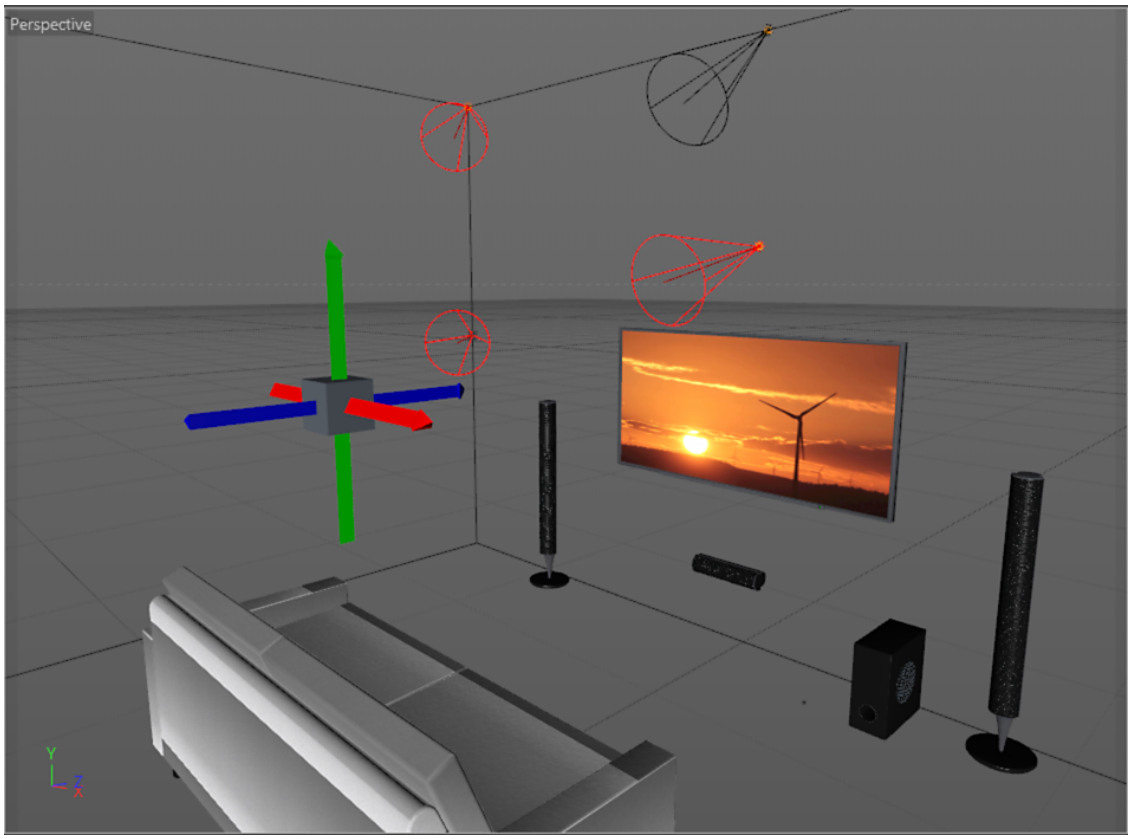

Figura VI.10: Ambiente de simulação em execução com atuadores ativos

cificação de coordenadas de um efeito, é verificado se o método de posicionamento é do padrão MPEG-V ou por coordenadas esféricas.

O simulador usa o método de posicionamento por coordenadas esféricas apresentado na Seção VI.3. Para possibilitar o uso de coordenadas esféricas em um ambiente tridimensional, é necessário um ponto de referência. Este ponto pode ser o usuário do sistema ou um objeto (e.g., sofá). Consequentemente, os atuadores do ambiente devem estar direcionados para este ponto. Esta alegação não se afasta das implementações atuais de sistemas com efeitos sensoriais, como o caso de cinemas 4D [Bartocci et al., 2015], em que os efeitos sensoriais de vento, calor e cheiro são executados em dispositivos direcionados a cadeiras individuais. Cada cadeira representa um ponto de referência único (i.e., para um único usuário) e utiliza-se um conjunto distinto de atuadores.

Para fixar a orientação dos atuadores no ponto de referência, utilizamos a restrição "*look-at*" [Blinn, 1988] do software 3D utilizado. Esta restrição certifica que um objeto (um atuador) situado em qualquer ponto alinhe sua direção de visão (usualmente o eixo Z) para um ponto de referência. O atuador ainda pode se mover livremente nos planos que compõem os limites do ambiente. Para satisfazer essa restrição, o *solver* de restrições do software 3D aplica rotações ao atuador para garantir que sua direção seja alinhada com o ponto de referência.

A Figura VI.11 mostra a utilização da restrição *look-at* para identificar atuadores no ambiente.

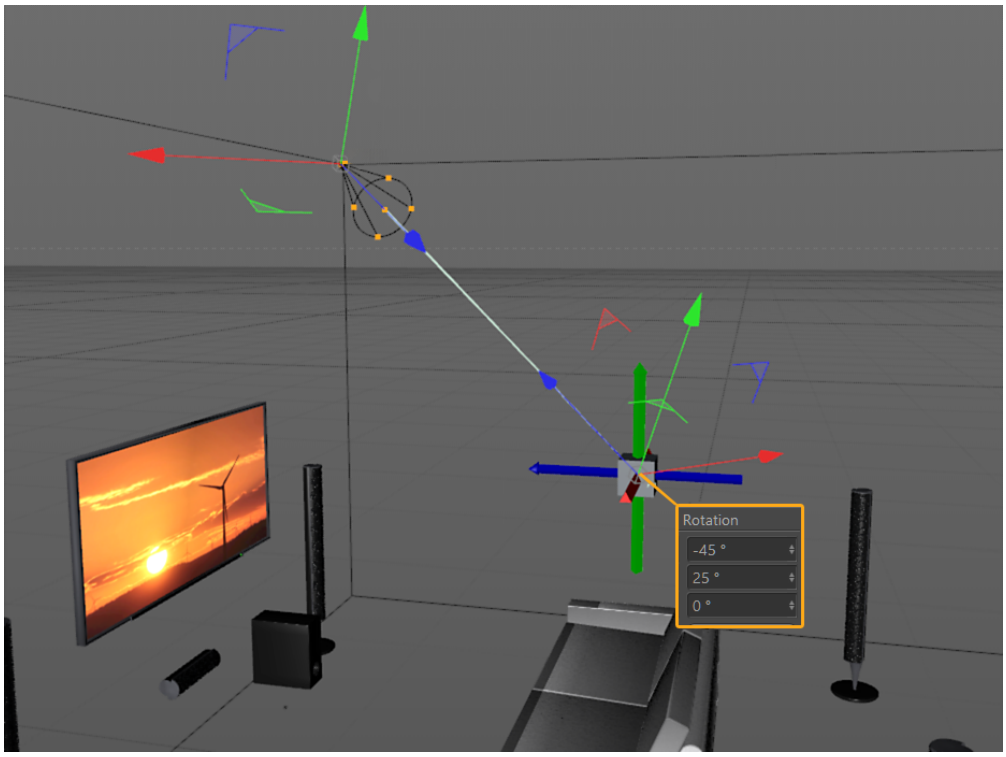

Figura VI.11: Orientação de ponto de referência para atuador no ambiente

Na figura, é desenhada uma linha partindo da orientação do eixo Z do ponto de referência. Partindo deste ponto, a linha tem a angulação de -45 no eixo horizontal e 25 no eixo vertical. Se esta angulação for de encontro com a coordenada Z de algum atuador no topo da sala, neste caso a angulação $[-45°, 25°]$ será usada para endereçar este atuador.

É importante notar que o simulador utiliza internamente apenas o método de posicionamento por coordenadas esféricas. Para suportar as descrições de acordo com MPEG-V, um dicionário de coordenadas foi criado para mapear todas as descrições de pontos tridimensionais para ângulos horizontais e verticais. Um trecho deste dicionário é mostrado na Listagem VI.8.

```
{
    "left:bottom:back": [135, -25],
    "left:middle:back": [135, 0],
    "left:top:back": [135, 25],
    "centerleft:bottom:back": [157.5, -33],
    "centerleft:middle:back": [157.5, 0],
    "centerleft:top:back": [157.5, 3],
    "center:bottom:back": [180, -33],
    ...
}
```

Listagem VI.8: Dicionário de posições MPEG-V em ângulos

Caso o método de posicionamento descrito no efeito seja do padrão MPEG-V, o simulador verifica se a descrição existe no dicionário. Então, o ângulo correspondente à coordenada é utilizado. Caso o método de posicionamento do efeito seja por coordenadas esféricas, o simulador apenas analisa os ângulos descritos no SEM e salva seus valores. São também analisados os valores do tamanho da área do efeito (largura e altura). Se uma descrição de ângulos não tiver um tamanho de área, esta área é assumida como sendo um único ponto no espaço. Importante notar que o dicionário de coordenadas atual não dá suporta a referência de um conjunto de localizações por meio do símbolo “$*$”.

Após ter guardado as informações de posicionamento de efeitos, o simulador percorre todos os atuadores do ambiente, verificando quais estão na angulação descrita pelos efeitos. Caso um atuador esteja de acordo com a descrição, ele é ativado. Os atuadores são ativados seguindo os atributos especificados para o efeito, e.g., sua intensidade.

Capítulo VII Conclusão

Este trabalho teve como principal motivação facilitar a autoria de efeitos sensoriais em aplicações multimídia interativas. Foi proposta uma abordagem de autoria de forma semiautomática ao descrever uma aplicação usando âncoras abstratas para realizar a sincronização de efeitos sensoriais com o conteúdo de mídia. As âncoras abstratas representam intervalos quando um determinado componente de cena é apresentado no conteúdo do nó de mídia. Assim, um autor de uma aplicação *mulsemedia* não precisa ter um conhecimento completo de um conteúdo do nó para definir sua sincronização com outros conteúdos.

Âncoras abstratas auxiliam no processo de autoria ao abstrair as definições de sincronização temporal de efeitos sensoriais. Esta tarefa então deixa de ser primariamente do autor e passa a ser feita por um processador que integra um software de reconhecimento baseado em redes neurais. O método proposto neste trabalho permite a autoria semiautomática destes efeitos. Pois o autor da aplicação pode decidir quais componentes de cena e efeitos sensoriais ele deseja utilizar na aplicação. Além disso, após o reconhecimento automático de efeitos, o autor também pode alterar manualmente a sincronização de efeitos individuais. Essa abordagem destina-se a um contexto *mulsemedia*, onde é comum realizar sincronização de efeitos sensoriais em relação ao conteúdo audiovisual. A abordagem, no entanto, não está restrita a ele e pode ser usada para especificações de aplicações multimídia tradicionais.

Juntamente com as âncoras abstratas, o Processador de Âncoras Abstratas (AAP) permite a geração automática de âncoras de um nó com base em seu conteúdo. Ele capta informações do documento e usa softwares de reconhecimento de cena para identificar as informações temporais para instanciar as âncoras abstratas corretamente. Um método de reconhecimento de cena já está integrado por padrão no AAP e utiliza uma API de reconhecimento de vídeo baseada em redes neurais.

Como forma de aprimorar o reconhecimento de componentes de cena relacionados a efeitos sensoriais, neste trabalho foi proposta uma arquitetura de rede neural bimodal. Esta arquitetura pode ser treinada para identificar componentes de cena relacionados a efeitos sensoriais em um objeto de mídia audiovisual. A arquitetura bimodal proposta compreende três módulos, dois deles para extrair recursos de áudio e vídeo, e um terceiro, chamado módulo de fusão, que combina os dois resultados para fazer a previsão dos rótulos relacionados a componentes da

cena. Para validar esta arquitetura, foi utilizado um subconjunto do conjunto de dados AudioSet. A arquitetura proposta foi validada em comparação com a predição de modalidades individuais de áudio e vídeo. Os experimentos mostraram um melhor desempenho da arquitetura bimodal com relação ao reconhecimento feito em cada modal individual.

Ainda, de forma a permitir a representação de efeitos sensoriais em aplicações interativas, foi apresentada uma proposta de integração de NCL com a reprodução de efeitos sensoriais pelo padrão MPEG-V. Esta forma de integração facilita a autoria de aplicações interativas pois evita que o autor precise criar código imperativo para definir relações entre eventos e ativações de efeitos sensoriais. Foi proposta ainda uma extensão do modelo de posicionamento do MPEG-V. Esta extensão permite o uso de coordenadas esféricas para definir a localização da ativação de efeitos sensoriais. Ao utilizar coordenadas esféricas, um autor de aplicação pode ter maior controle sobre o posicionamento de efeitos e realizar animações com maior granularidade. Foi apresentada a arquitetura e implementação de um módulo Lua que converte descrições em NCL para comandos em MPEG-V (na forma de arquivos SEM) e envia-os pela rede para reprodutores compatíveis. O protótipo apresentado neste trabalho considerou apenas a tradução de comandos relacionados a apresentação do efeito sensorial.

Por fim, visando permitir a prototipação rápida de uma aplicação, foi proposto um simulador 3D para efeitos sensoriais. Este simulador recebe pela rede comandos descritos em SEDL e realiza a ativação de efeitos sensoriais em um ambiente 3D. Neste simulador é possível adicionar, remover ou movimentar atuadores no ambiente.

VII.1 Trabalhos Futuros

No contexto de âncoras abstratas, um trabalho futuro é realizar uma avaliação experimental do impacto da utilização da ferramenta no processo de autoria de aplicações *mulsemedia*. Principalmente para avaliar se o processo de instanciação de âncoras estará de acordo com o esperado pelo autor da aplicação. Outro trabalho futuro é aprimorar o *AAP* com a capacidade de inferir sinônimos das palavras usadas para descrever componentes de cena. A abordagem atual para identificar componentes de cena pode ser propensa a erros, visto que o autor pode identificar um componente com uma nomenclatura diferente daquela usada pelo software de reconhecimento. Tal fato pode se agravar dado que existem diversas implementações de softwares de reconhecimento disponíveis e eles podem não seguir um padrão comum para nomenclatura de componentes de cena.

No contexto da rede neural bimodal, um trabalho futuro é verificar se redes treinadas com o conjunto de dados AudioSet mantêm seu desempenho ao serem aplicadas para reconhecimento de cena para clipes de 1 segundo. O passo de tempo de 1s foi selecionado como padrão

para o AAP pois de acordo com resultados apresentados por trabalhos publicados na literatura apresentados na Seção II.4 este passo está de acordo com uma QoE aceitável pelo usuário.

Outro trabalho futuro importante é verificar o tempo mínimo para reconhecimento de componentes de cena da modalidade de áudio em diferentes passos de tempo, visto que componentes de cena de áudio precisam de uma certa duração para serem reconhecidos. Mais um trabalho futuro neste contexto é construir um conjunto de dados apropriado para reconhecimento de componentes de cena relacionados a efeitos sensoriais. Este conjunto deve conter exemplos de treinamento adequados para reconhecimento de componentes de cena relacionados a efeitos sensoriais.

No contexto da reprodução de efeitos sensoriais, um trabalho futuro é estender o módulo de forma a controlar animações da posição de um efeito. Esta extensão visa se aproveitar da definição de posicionamento por coordenadas esféricas proposta na Seção VI.3. Sendo assim, o módulo pode utilizar-se das descrições de animação existentes em NCL para gerar animações de intensidade ou posicionamento de efeitos sensoriais.

No contexto do simulador 3D, um trabalho futuro é aprimorar o ambiente 3D do simulador para representar atuadores como objetos similares aos físicos. Por exemplo um atuador de vento pode ser representado por um ventilador. Também é necessário aprimorar o código do simulador para poder ativar atuadores de acordo com o tipo de efeito sensorial descrito. Além do mais, o simulador foi desenvolvido em um software de modelagem 3D proprietário. Portanto um trabalho futuro neste contexto é desenvolvê-lo em softwares gratuitos como, por exemplo, *Blender*.[1]

Como mencionado ao final da Seção VI.3, apesar do modelo de posicionamento apresentado permitir uma maior granularidade no posicionamento dos atuadores, este posicionamento não é tão expressivo quanto o modelo tridimensional do MPEG-V a respeito da distância de atuadores. Esta expressividade pode ser alcançada ao estender a proposta de posicionamento para contar, além da angulação horizontal e vertical, com uma distância do ponto central até o atuador. Portanto um trabalho futuro é implementar esta modificação.

[1] https://www.blender.org

Referências Bibliográficas

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Abdel-Hamid, O., Mohamed, A.-r., Jiang, H., Deng, L., Penn, G., and Yu, D. (2014). Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545.

ABNT (2011). Digital terrestrial television - data coding and transmission specification for digital broadcasting - part 2: Ginga-ncl for fixed and mobile receivers - xml application language for application coding. ABNT NBR 15606-2:2011 standard.

Abreu, R. and dos Santos, J. (2017a). Using abstract anchors for automatic authoring of sensory effects based on ambient sound recognition. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, WebMedia '17, pages 437–440, New York, NY, USA. ACM.

Abreu, R. and dos Santos, J. (2017b). Using abstract anchors to aid the development of multimedia applications with sensory effects. In *Proceedings of the 2017 ACM Symposium on Document Engineering*, DocEng '17, pages 211–218, New York, NY, USA. ACM.

Abreu, R., dos Santos, J., and Bezerra, E. (2018). A bimodal learning approach to assist multisensory effects synchronization. In *International Joint Conference on Neural Networks*, IJCNN '18. IEEE.

Allen, J. (1977). Short term spectral analysis, synthesis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3):235–238.

Aytar, Y., Vondrick, C., and Torralba, A. (2016). Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems 29*, pages 892–900. Curran Associates, Inc.

Bartocci, S., Betti, S., Marcone, G., Tabacchiera, M., Zanuccoli, F., and Chiari, A. (2015). A novel multimedia-multisensorial 4d platform. In *2015 AEIT International Annual Conference (AEIT)*, pages 1–6.

Bezerra, D. H. D., Sousa, D. M. T., Filho, G. L. d. S., Burlamaqui, A. M. F., and Silva, I. R. M. (2012). Luar: A language for agile development of ncl templates and documents. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web*, WebMedia '12, pages 395–402, New York, NY, USA. ACM.

Blakowski, G. and Steinmetz, R. (1996). A media synchronization survey: reference model, specification, and case studies. *IEEE Journal on Selected Areas in Communications*, 14(1):5–35.

Blinn, J. (1988). Where am i? what am i looking at?(cinematography). *IEEE Computer Graphics and Applications*, 8(4):76–81.

Boddapati, V., Petef, A., Rasmusson, J., and Lundberg, L. (2017). Classifying environmental sounds using image recognition networks. *Procedia Computer Science*, 112:2048 – 2056. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference, September 2017, Marseille, France.

Bulterman, D. C. A. and Hardman, L. (2005). Structured multimedia authoring. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(1):89–109.

Cao, L., Mu, Y., Natsev, A., Chang, S.-F., Hua, G., and Smith, J. R. (2012). Scene aligned pooling for complex video recognition. In *Computer Vision–ECCV 2012*, pages 688–701. Springer.

Chollet, F. et al. (2015). Keras. https://github.com/fchollet/keras.

Crockford, D. (2006). The application/json media type for javascript object notation (json). Technical report.

Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C. (1992). The cave: Audio visual experience automatic virtual environment. *Commun. ACM*, 35(6):64–72.

Dieleman, S., Brakel, P., and Schrauwen, B. (2011). Audio-based music classification with a pretrained convolutional network. In *12th International Society for Music Information Retrieval Conference (ISMIR-2011)*, pages 669–674. University of Miami.

Dinh, H. Q., Walker, N., Hodges, L. F., Song, C., and Kobayashi, A. (1999). Evaluating the importance of multi-sensory input on memory and the sense of presence in virtual environments. In *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, pages 222–228.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Feng, W., Guan, N., Li, Y., Zhang, X., and Luo, Z. (2017). Audio visual speech recognition with multimodal recurrent neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 681–688.

Freed, N. and Borenstein, N. (1996). Multipurpose internet mail extensions (mime) part one: Format of internet message bodies. Technical report.

Freina, L. and Ott, M. (2015). A literature review on immersive virtual reality in education: State of the art and perspectives. *eLearning & Software for Education*, (1).

Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M. (2017). Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA.

Ghinea, G. and Ademoye, O. A. (2010). Perceived synchronization of olfactory multimedia. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 40(4):657–663.

Ghinea, G., Timmerer, C., Lin, W., and Gulliver, S. R. (2014). Mulsemedia : State of the Art, Perspectives, and Challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(1s):1–23.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Guedes, A. L., de Albuquerque Azevedo, R. G., Colcher, S., and Barbosa, S. D. (2016a). Extending ncl to support multiuser and multimodal interactions. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web*, Webmedia '16, pages 39–46, New York, NY, USA. ACM.

Guedes, A. L., de Cunha, M., Fuks, H., Colcher, S., and Barbosa, S. D. J. (2016b). Using ncl to synchronize media objects, sensors and actuators. In *Proceedings of 1st International Workshop on Synchronism of Things (WSoT)*, Webmedia '16.

Guedes, Á. L. V., de Albuquerque Azevedo, R. G., and Barbosa, S. D. J. (2017). Extending multimedia languages to support multimodal user interactions. *Multimedia Tools and Applications*, 76(4):5691–5720.

Hanrahan, P. and Haeberli, P. (1990). Direct wysiwyg painting and texturing on 3d shapes. *ACM SIGGRAPH computer graphics*, 24(4):215–223.

Hardman, L., Schmitz, P., van Ossenbruggen, J., ten Kate, W., and Rutledge, L. (2000). The link vs. the event: activating and deactivating elements in time-based hypermedia. *New Review of Hypermedia and Multimedia*, 6(1):89–109.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hülsmann, F., Mattar, N., Fröhlich, J., and Wachsmuth, I. (2014). Simulating wind and warmth in virtual reality: conception, realization and evaluation for a cave environment. *Journal of Virtual Reality and Broadcasting*, 11(10).

Ierusalimschy, R. (2006). *Programming in lua*. Roberto Ierusalimschy.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

ISO/IEC 14496-11:2015 (2015). Information technology – Coding of audio-visual objects – Part 11: Scene description and application engine. Standard, International Organization for Standardization, Geneva, CH.

ISO/IEC 23005-1 (2016). Information technology – Media context and control – Part 1: Architecture. Standard, International Organization for Standardization, Geneva, CH.

ISO/IEC 23005-3 (2016). Information technology – Media context and control – Part 3: Sensory information. Standard, International Organization for Standardization, Geneva, CH.

ITU (2008). Telephone transmission quality, telephone installations, local line networks - vocabulary for performance and quality of service. https://www.itu.int/rec/T-REC-P.10-200807-I!Amd2. ITU-T Recommendation P.10/G.100.

ITU (2009). Nested context language (ncl) and ginga-ncl for iptv services. http://www.itu.int/rec/T-REC-H.761-200904-S. ITU-T Recommendation H.761.

Jaimes, A. and Sebe, N. (2007). Multimodal human–computer interaction: A survey. *Computer Vision and Image Understanding*, 108(1–2):116 – 134. Special Issue on Vision for Human-Computer Interaction.

Jain, R., Militzer, D., and Nagel, H.-H. (1977). Separating non-stationary from stationary scene components in a sequence of real world tv-images. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'77, pages 612–618, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Josué, M., Abreu, R., Barreto, F., Mattos, D. P., Amorim, G. F., dos Santos, J. A. F., and Muchaluat-Saade, D. C. (2018). Modeling sensory effects as first-class entities in multimedia applications. In *ACM Multimedia Systems Conference*.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA. IEEE Computer Society.

Kim, S. K., Joo, Y. S., and Lee, Y. (2013). Sensible media simulation in an automobile application and human responses to sensory effects. *ETRI Journal*, 35(6):1001–1010.

Kim, S. K., Yang, S. J., Ahn, C. H., and Joo, Y. S. (2014). Sensorial information extraction and mapping to generate temperature sensory effects. *ETRI Journal*, 36(2):224–231.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kunishige, Y., Yaokai, F., and Uchida, S. (2011). Scenery character detection with environmental context. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1049–1053. IEEE.

Lapin, M., Hein, M., and Schiele, B. (2018). Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1533–1554.

LeCun, Y. and Bengio, Y. (1998). The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

Lee, J., Han, B., and Choi, S. (2016). Motion effects synthesis for 4d films. *IEEE Transactions on Visualization and Computer Graphics*, 22(10):2300–2314.

Logan, B. et al. (2000). Mel frequency cepstral coefficients for music modeling. In *ISMIR*, volume 270, pages 1–11.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25.

Mishkin, D., Sergievskiy, N., and Matas, J. (2017). Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*, 161:11–19.

Muchaluat-Saade, D. C. and Soares, L. F. G. (2002). Xconnector & xtemplate: Improving the expressiveness and reuse in web authoring languages. *The New Review of Hypermedia and Multimedia Journal*, 8(1):139–169.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Ng, J. Y.-H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., and Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4694–4702. IEEE.

Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 689–696, USA. Omnipress.

Oviatt, S. (2003). In Jacko, J. A. and Sears, A., editors, *The Human-computer Interaction Handbook*, chapter Multimodal Interfaces, pages 286–304. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Piczak, K. J. (2015a). Environmental sound classification with convolutional neural networks. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6.

Piczak, K. J. (2015b). Environmental sound classification with convolutional neural networks. In *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*, pages 1–6. IEEE.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Salamon, J. and Bello, J. P. (2015). Feature learning with deep scattering for urban sound analysis. In *Signal Processing Conference (EUSIPCO), 2015 23rd European*, pages 724–728. IEEE.

Saleme, E. B. and Santos, C. A. S. (2015). PlaySEM: A Platform for Rendering MulSeMedia Compatible with MPEG-V. *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*, pages 145–148.

Santos, C. A. S., Neto, A. N. R., and Saleme, E. B. (2015). An event driven approach for integrating multi-sensory effects to interactive environments. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 981–986.

Santos, J. A. and Muchaluat-Saade, D. C. (2012). Xtemplate 3.0: Spatio-temporal semantics and structure reuse for hypermedia compositions. *Multimedia Tools Appl.*, 61(3):645–673.

Shin, S. H., Ha, K. S., Yun, H. O., and Nam, Y. S. (2016). Realistic media authoring tool based on MPEG-V international standard. *International Conference on Ubiquitous and Future Networks, ICUFN*, 2016-Augus:730–732.

Soares, L. F. G. and Barbosa, S. D. J. (2012). Programando em ncl 3.0. *Desenvolvimento de Aplicações para o Middleware Ginga, TV digital e Web*, 1.

Soares, L. F. G., Rodrigues, R. F., and de Resende Costa, R. M. (2005). Nested context model 3.0. *Monographies in Computer Sciences*, 18(05):35.

Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Sulema, Y. (2016). Mulsemedia vs. multimedia: State of the art and future trends. In *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–5.

Timmerer, C., Waltl, M., Rainer, B., and Hellwagner, H. (2012). Assessing the quality of sensory experience for multimedia presentations. *Signal Processing: Image Communication*, 27(8):909–916.

Torfi, A., Iranmanesh, S. M., Nasrabadi, N. M., and Dawson, J. M. (2017). 3d convolutional neural networks for cross audio-visual matching recognition. *IEEE Access*, 5:22081–22091.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Tsoumakas, G. and Vlahavas, I. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *European conference on machine learning*, pages 406–417. Springer.

Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651.

W3C (2008). Synchronized multimedia integration language - smil 3.0 specification. http://www.w3c.org/TR/SMIL3. World-Wide Web Consortium Recommendation.

W3C (2017). Html 5.1 - hypertext markup language. http://www.w3c.org/TR/HTML. World-Wide Web Consortium Recommendation.

Waltl, M., Rainer, B., Timmerer, C., and Hellwagner, H. (2013). An end-to-end tool chain for Sensory Experience based on MPEG-V. *Signal Processing: Image Communication*, 28(2):136–150.

Waltl, M., Timmerer, C., and Hellwagner, H. (2009). A test-bed for quality of multimedia experience evaluation of sensory effects. In *2009 International Workshop on Quality of Multimedia Experience*, pages 145–150.

Wu, W., Arefin, A., Rivas, R., Nahrstedt, K., Sheppard, R., and Yang, Z. (2009). Quality of experience in distributed interactive multimedia environments: toward a theoretical framework. *ACM Multimedia Conference*, pages 481–490.

Yang, X., Molchanov, P., and Kautz, J. (2016). Multilayer and multimodal fusion of deep neural networks for video classification. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 978–987, New York, NY, USA. ACM.

Yoon, K., Choi, B., Lee, E. S., and Lim, T. B. (2010). 4-d broadcasting with mpeg-v. In *2010 IEEE International Workshop on Multimedia Signal Processing*, pages 257–262.

Yuan, Z., Chen, S., Ghinea, G., and Muntean, G.-M. (2014). User Quality of Experience of Mulsemedia Applications. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 11(1s):1–19.

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

Zhang, H., McLoughlin, I., and Song, Y. (2015). Robust sound event recognition using convolutional neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 559–563. IEEE.

Zhang, M.-L. and Zhou, Z.-H. (2014). A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Ziat, M., Balcer, C. A., Shirtz, A., and Rolison, T. (2016). A century later, the hue-heat hypothesis: does color truly affect temperature perception? In *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*, pages 273–280. Springer.