Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Comunicações

TV Interativa Baseada na Inclusão de Informações Hipermídia em Vídeos no Padrão MPEG

Autor: André Leon Sampaio Gradvohl

Orientador: Prof. Dr. Yuzo Iano

Tese de Doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.
Área de concentração: **Telecomunicações e Telemática**.

Banca Examinadora

Yuzo Iano, Dr. ..... DECOM/FEEC/UNICAMP
Álvaro Luiz Stelle, PhD. ..... CEFET-PR
Maurício Silveira, Dr. ..... DTE/INATEL
João Baptista Tadanobu Yabu-uti, Dr. ..... DECOM/FEEC/UNICAMP
Luiz César Martini, Dr. ..... DECOM/FEEC/UNICAMP
Vicente Idalberto Becerra Sablón, Dr. ..... UNISAL

Campinas, SP
Março/2005

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G755t

Gradvohl, André Leon Sampaio
TV interativa baseada na inclusão de informações hipermídia em vídeos no padrão MPEG / André Leon Sampaio Gradvohl. - -Campinas, SP: [s.n.], 2005.

Orientador: Yuzo Iano.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de hipermídia. 2. Televisão digital. 3. Vídeo interativo. 4. Multimídia interativa. I. Iano, Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Interactive television based on insertion of hypermedia information on MPEG standard video.
Palavras-chave em Inglês: Hypermedia systems, Digital TV, Digital television, Interactive video e Interactive media.
Área de concentração: Telecomunicações e Telemática
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Álvaro Luiz Stelle, Mauricio Silveira, João Baptista Tadanobu Yabu-uti, Luiz César Martini e Vicente Idalberto Becerra Sablón
Data da defesa: 18/03/2005

Resumo

TV interativa é uma das grandes promessas no campo dos aparelhos eletrônicos para o consumidor. Entretanto, ainda não há um padrão definitivo para implementar esse novo tipo de dispositivo. Por isso, pesquisas são conduzidas para desenvolver um padrão barato, simples e, ao mesmo tempo, o mais funcional possível. Seguindo essa linha de raciocínio, este trabalho propõe um modelo para a inserção de informações hipermídia em vídeos em um padrão amplamente conhecido e já implementado nos televisores digitais atuais: o padrão MPEG.

O modelo proposto é baseado na elaboração de um conceito próprio de hipervídeo, no qual são embutidas referências a informações complementares ao conteúdo sendo exibido. Tais referências e as coordenadas dos objetos em cena, que atuam como âncoras para as informações complementares, formam o que se define como informações hipermídia. Além da concepção do modelo, este trabalho também apresenta uma prova de conceito para demonstrar a viabilidade das idéias aqui descritas.

Palavras-chave: hipervídeo, padrão MPEG, sistemas hipermídia, TV digital, TV interativa.

Publicações:

A. L. S. Gradvohl, Y. Iano. "An approach for interactive televison based on insertion of hypermedia information on MPEG standard video". In: *P*roceedings of IEEE International Symposium on Consumer Electronics. Reading, United Kingdom. Páginas 25-30. 1 a 3 de Setembro de 2004.

A. L. S. Gradvohl, Y. Iano. "An approach for interactive televison based on insertion of hypermedia information on MPEG standard video". Submetido para apreciação na IEEE Transactions on Consumer Electronics.

A. L. S. Gradvohl, Y. Iano. "Hypervideo meets Interactive TV". Submetido para apreciação na IEEE Transactions on Multimedia.

Abstract

The interactive TV is one of the great promises on Consumer Electronics field. However, there is no definitive standard to implement this new kind of device. For this reason, researches are conducted to develop a standard cheap, simple, and also the most functional as possible. Tracing this reasoning, this work proposes a model to insert hypermedia information on videos in a widely known and already implemented standard in current digital television sets: the MPEG standard.

The proposed model is based on the creation of a particular concept of hypervideo, which references to complementary information are embedded on the content being presented. Such references and the coordinates of the objects in scene, which act as anchors to complementary information, both form what is defined as hypermedia information. Beyond the conception of the model, this work also presents a concept proof to demonstrate the viability of the ideas here depicted.

Keywords: digital TV, hypermedia systems, hypervideo, interactive TV, MPEG standard.

Publications:

A. L. S. Gradvohl, Y. Iano. "An approach for interactive televison based on insertion of hypermedia information on MPEG standard video". In: *P*roceedings of IEEE International Symposium on Consumer Electronics. Reading, United Kingdom. Pages 25-30. September 1-3, 2004.

A. L. S. Gradvohl, Y. Iano. "An approach for interactive televison based on insertion of hypermedia information on MPEG standard video". Submitted to IEEE Transactions on Consumer Electronics.

A. L. S. Gradvohl, Y. Iano. "Hypervideo meets Interactive TV". Submitted to IEEE Transactions on Multimedia.

Dedicatória

à minha querida Silvia
e aos nossos filhos que hão de vir.

Agradecimentos

"Não vos esqueçais da hospitalidade,
porque por ela alguns, sem o saberem, hospedaram anjos."
Hebreus 13:2.

Antes de tudo, agradeço a Deus por ter me iluminado, dado saúde e paciência para superar todas as adversidades.

Agradeço também a minha esposa, Silvia, por ter estado sempre ao meu lado, com seu companherismo ímpar e seu amor incondicional.

Ao meu orientador, Prof. Yuzo Iano, pelos valiosos conselhos dados no decorrer desse trabalho, pela paciência e pelo incentivo dados em todas as horas difíceis ou não.

Aos meus pais e irmãos, por serem a base da minha educação e por fazerem com que eu acredite no meu potencial. Aos avós, tios, meus sogros e demais familiares que sempre estiveram na torcida por mim.

Aos companheiros da pós-graduação da FEEC, os Imortais, agradeço pela parceria na busca pelo desenvolvimento científico, tecnológico e cultural do nosso país, bem como pelos intermináveis cafés às 16:00h. Aos colegas do CENAPAD-SP, pelo suporte dado a mim no decorrer da tese.

Enfim, a todos aqueles que contribuíram de uma forma ou de outra para que eu possa ter concretizado este trabalho.

Sumário

Lista de Símbolos

Os símbolos matemáticos a seguir foram utilizados no decorrer desse texto, em particular no Capítulo 3. As demais abreviaturas e definições estão na seção "Definições e Abreviaturas", na página 70.

Símbolo	**Significado**
(i, j)	Coordenadas espaciais no domínio da transformada.
(k, l)	Coordenadas da freqüência no domínio da amostra.
A	Alfabeto de símbolos.
$Dct[i, j]$	Matriz de saída da função DCT.
$H_{N-1}, H_{N-2}, \ldots, H_1$	Símbolos hipotéticos cujas probabilidades são a soma das probabilidades dos símbolos que os geraram.
$P(A)$	Conjunto de probabilidades de ocorrência dos símbolos do alfabeto.
$p_1, p_2, \ldots, p_N$	Probabilidades de ocorrência de cada um dos símbolos do alfabeto.
q_{kl}	Elemento da linha k, coluna l da matriz de quantização.
Q	Matriz de quantização.
Q^{-1}	Matriz de quantização inversa.
$s_1, s_2, \ldots, s_N$	Símbolos do alfabeto.
t_i	Tamanho da palavra código para o símbolo s_i.

Continuação da lista de símbolos matemáticos usados nesse texto.

Símbolo	**Significado**
T	Tamanho da maior palavra código.
$[T]$	Matrriz de vetores base da DCT.
v_1 e pv_1	Vetores de movimento associados ao campo ímpar.
v_2 e pv_2	Vetores de movimento associados ao campo par.
$\delta(x)$ e $\delta(y)$	Vetores de movimento diferencial nos eixos x e y respectivamente.
x_{ij}	Intensidade do "*pixel*" ("*picture element*") na linha i, coluna j.
$[X]$	Matriz de coordenadas espaciais no domínio da transformada DCT.
y_{kl}	Coeficiente DCT na linha k, coluna l da matriz DCT.
$[Y]$	Matriz de coordenadas ded freqüência no domínio da amostra.
z_{kl}	Quantização uniforme de y_{kl}.
$\hat{z}_{kl}$	Quantização inversa de z_{kl}.

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

"*Lasciate ogni speranza ch'entrate.*[1]"
Dante Alighieri - Inferno, Canto III, 9.

O primeiro capítulo deste texto se inicia explicando a motivação para a execução do trabalho. Em seguida, formaliza-se a hipótese que conduziu o trabalho e que, no final do texto, pretende-se comprovar. Depois, para tornar mais claros os resultados esperados, são traçados os objetivos do trabalho. Finalmente, a estrutura do texto é detalhada para melhor guiar a leitura.

1.1 Motivação para o Projeto

O modo como se assiste à televisão não mudou muito nos últimos anos. Desde que foi inventado, o aparelho de TV é um meio de comunicação que expõe o telespectador a uma infinidade de informações visuais e auditivas, geralmente superficiais. Em outras palavras, raramente o telespectador pode obter mais informações sobre o conteúdo assistido no momento em que é transmitido.

A principal motivação deste projeto é a proposta de uma nova forma de se assistir TV. Ou seja,

[1] "Deixai toda esperança, ó vós que entrais."

observou-se que em várias situações existe uma carência por mais informações além daquelas apresentadas em imagens de TV. Essa necessidade de se buscar informações adicionais motivou a pesquisa por uma possibilidade de convergência entre TV e Internet, particularmente a "*World Wide Web*" (WWW).

A idéia, portanto é tornar a TV um ponto de partida para navegação, busca de mais informações e interação, criando assim uma TV interativa.

1.2 Hipótese

Formalizando a motivação do projeto, propõe-se a hipótese a seguir:

> É possível criar um programa de TV tal que tenha embutido em si um conjunto de referências que podem conduzir o telespectador a uma outra fonte de informação veiculada por outra mídia, interativa ou não, com conteúdo adicional àquele transmitido pelo próprio programa de TV. Cada elemento do conjunto de referências pode ser ativado independente dos demais, de acordo com a vontade do telespectador.

1.3 Objetivos

O trabalho relatado neste texto tem dois objetivos principais. O primeiro é a concepção de um modelo de hipervídeo, que esteja de acordo com a hipótese enunciada na seção anterior. O segundo objetivo principal é a implementação do modelo proposto, que servirá como prova de conceito para a hipótese.

Há ainda outros objetivos secundários. Entre eles está uma breve discussão a respeito de convergência digital - um tema em voga nos dias atuais e área onde este trabalho está inserido - e um resumo a respeito da família de padrões MPEG, que serviu como base tecnológica para este trabalho.

1.4 Estrutura do Texto

A estrutura do restante do texto está ilustrada na Figura 1.1 a seguir sendo formada pelos seguintes capítulos:

- no Capítulo 2 discutem-se algumas estratégias e padrões para a convergência digital entre TV e Internet;
- no Capítulo 3 apresenta-se um breve resumo a respeito da família de padrões MPEG para vídeo digital;
- o conceito e a composição de um hipervídeo são demonstrados no Capítulo 4;
- as estratégias de implementação do protótipo são explicadas no Capítulo 5;
- no Capítulo 6 são apresentados os resultados, conclusões e as propostas de trabalhos futuros.

A linha tracejada na Figura 1.1 a seguir sugere um atalho que conduz à parte principal deste texto. Além dos capítulos citados, na página 70 estão as definições de alguns termos técnicos e abreviaturas usados neste texto e no Apêndice A estão os principais trechos de código do protótipo desenvolvido.

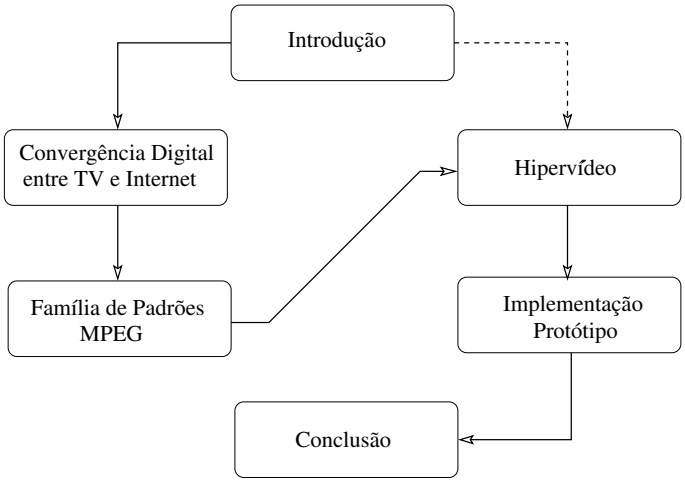


Figura 1.1: Estrutura do texto.

CAPÍTULO 2

Convergência Digital entre Televisão e Internet

"*Pluralitas non est ponenda sine necessitas.*[1]"
William de Ockham - A Navalha de Okcham.

Convergência digital é um conceito em evidência nos dias atuais. A quantidade de serviços requeridos pelos usuários, em particular serviços de comunicação, tornou necessária a justaposição de várias mídias em um único dispositivo. A motivação para esse conceito é transmitir a maior quantidade de informação no menor tempo possível.

Neste capítulo são apresentados alguns paradigmas envolvidos no conceito de convergência digital em geral e são discutidas as atuais propostas de implementação de convergência digital entre TV e Internet em particular.

2.1 Telecomunicações, Interatividade e Convergência

É notório o impulso que as telecomunicações tiveram a partir do final do século XIX, quando o rádio foi inventado em 1895. Desde então, saltos tecnológicos vêm ocorrendo e cada dia as pessoas

[1]"Pluralidade não é proposta sem necessidade". Em outras palavras, por que assumir que as coisas são complexas se uma teoria simples pode explicar todas as observações? De acordo com Ockham.

exigem comunicações mais rápidas e confiáveis. Cresceu também a necessidade por interatividade.

O conceito de interatividade ainda é vago e usado tanto para relação de pessoas com máquinas ou softwares, quanto para relação de pessoas entre si. Porém, apesar de impreciso, esse conceito serviu como alicerce para a consolidação das telecomunicações [26, 32, 33].

Quanto mais interativo for um dispositivo usado para comunicação melhor ele é considerado, ou seja, tal dispositivo atende melhor às especificidades da vontade do usuário. Tome, por exemplo, a chamada TV interativa que transmite jogos de futebol. Diz-se que é possível escolher qual o melhor ângulo para ver determinados lances, ou mesmo responder às pesquisas de opinião propostas pela emissora. Essa é a interatividade que a maioria das empresas dessa área promete.

Entretanto, pode-se observar esse cenário de outro modo. O que ocorre, de acordo com esse novo ponto de vista, é uma troca de informações. Tais informações poderiam estar embutidas no próprio programa sendo transmitido, tornando-o mais dinâmico, permitindo maior interatividade.

Aproveitando o exemplo já mencionado, imagine se fosse possível para o telespectador obter mais detalhes (informações) a respeito do que está sendo visto em cada momento. Será que essa possibilidade incrementaria a interação com o usuário?

A resposta à pergunta anterior seria afirmativa se for considerado que podem existir mais possibilidades a serem exploradas além do que está sendo visto. Mas, como seria o retorno? Afinal, interatividade exige troca. A réplica é possível, desde que seja aberto um canal paralelo de comunicação.

Nesse instante surge o conceito de convergência digital, i. e., juntando duas tecnologias bem conhecidas em um único dispositivo, simples de usar, é possível obter maior interatividade e conseqüentemente melhor comunicação. O fato de ser digital ajuda na interoperabilidade dessas tecnologias.

Alguém poderia questionar os conceitos anteriores afirmando que um computador onde se pode ver um vídeo e acessar uma página na WWW, por exemplo, também é uma instância de convergência digital. Talvez seja, mas esse conceito é mais sutil. É preciso que haja sincronismo. Transmitir ou receber mais de uma forma de comunicação é necessário, mas não suficiente, para determinar que um dispositivo é um exemplo de convergência digital. Para pertencer ao conceito de convergência digital é essencial que essas formas de comunicação apresentem conteúdos complementares e integrados [20].

2.2 Tendências em Convergência Digital

O Sr. Jack Fenn, um analista em uma empresa americana chamada Gartner Group, propôs a representação gráfica para um modelo de maturidade, adoção e aplicação comercial de tecnologias emergentes. Essa representação gráfica, chamada de "*Hype Cycle*", foi proposta em 1995 [49]. O objetivo dessa representação é tentar prever aproximadamente quando e o quanto investir em determinadas tecnologias.

Esse modelo tem cinco fases ilustradas na Figura 2.1, a saber:

1. Lançamento da tecnologia: quando um produto é lançado no mercado ou um evento gera interesse no produto.

2. Pico de expectativas inflacionadas: quando são geradas expectativas excessivas e não realistas a respeito da utilização do produto.

3. Poço de desilusão: quando se percebe que as expectativas em relação ao produto foram superestimadas.

4. Inclinação de esclarecimento: quando as expectativas são mais realistas e a tecnologia passa a ser bem compreendida.

5. Platô de produtividade: quando a tecnologia passa a ter seus benefícios amplamente demonstrados e reconhecidos, tornando-se mais estável seu desenvolvimento.

O Gartner Group publica anualmente uma série de relatórios que mostram o status de tecnologias emergentes em várias áreas. Em agosto de 2004, esse relatório mostrava que algumas tecnologias para consumo doméstico estavam caminhando para a fase do poço de desilusão (Fase 3) com previsão de 10 anos para atingir a fase platô de produtividade (Fase 5), e. g., HDTV, TV interativa, vídeo sob demanda. Outras, como TV digital e banda larga, já estão na fase de platô de produtividade ou muito próximas disso [22, 46].

De acordo com o gráfico na Figura 2.1, apesar do desenvolvimento em TV interativa as expectativas estão sendo deflacionadas. Isso indica que é uma boa hora de se investir nesse produto, uma vez que não existe ainda nenhuma abordagem definitiva, ou seja, técnicas para implementar a TV

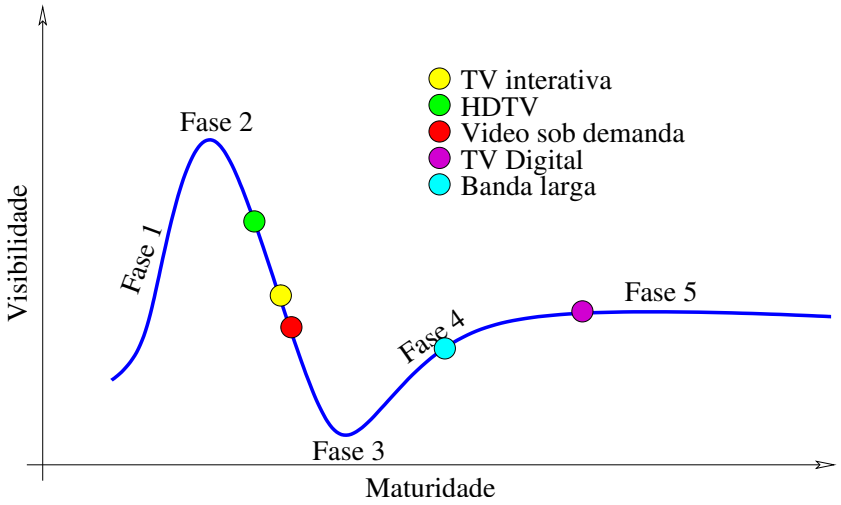


Figura 2.1: "*Hype Cycle*" de tecnologias emergentes para consumo doméstico.

interativa ainda podem ser propostas e melhoradas. Esse é mais um motivo para o desenvolvimento deste projeto.

2.3 Algumas Propostas para Convergência Digital entre TV e Internet

Convergência entre TV e Internet é uma possibilidade que existe desde que a WWW atingiu o meio comercial. Alguns padrões propostos para tentar alcançar essa convergência foram selecionados e são descritos nas seções a seguir.

2.3.1 Grupo de Especialistas em Codificação de Informação Multimídia e Hipermídia

O grupo de especialistas em codificação de informação multimídia e hipermídia ("*Multimedia and Hypermedia information coding Expert Group*"), ligado à ISO, é responsável pela especificação do padrão MHEG. O objetivo do padrão MHEG é o suporte à distribuição de aplicações multimídia interativas em um ambiente cliente-servidor. Um dos produtos do MHEG é uma linguagem para descrever apresentações multimídia [9, 15, 16].

2.3.2 Linguagem para Integração de Multimídia Sincronizada

A linguagem para integração de multimídia sincronizada ("*Synchronized Multimedia Integration Language*" - SMIL) foi criada pelo W3C ("*World Wide Web Consortium*") com o intuito de integrar um conjunto de objetos multimídia independentes em uma apresentação sincronizada [53]. A SMIL é baseada em XML[2] e, portanto, compatível com outras linguagens de marcação de textos. A idéia da SMIL é permitir que o autor da apresentação descreva seu comportamento temporal, descreva o "*layout*" dela e associe hiperlinks com objetos de mídia [43, 57].

Como exemplo, a Figura 2.2 mostra o código de uma apresentação em SMIL. Tal código especifica duas regiões (identificadas pelo comando `<region`) onde serão alocadas duas imagens, `vim32x32.gif` e `madewithsoja.gif`, que serão apresentadas ambas por seis segundos.

Nota-se que em um documento escrito em SMIL, a primeira parte do código especifica o "*layout*" da aplicação. Em seguida, associam-se os objetos e sua duração temporal e posições no "*layout*". É interessante observar a presença do comando `<par>` indicando que ambos os eventos ocorrem em paralelo.

Apesar dos atrativos, as apresentações construídas usando a SMIL como ferramenta precisam ser totalmente carregadas antes de serem apresentadas. Isso acarreta em um atraso significativo, tornando-a inviável para aplicações de TV interativa.

2.3.3 Plataforma Multimídia Doméstica

A Plataforma Multimídia Doméstica ("*Multimedia Home Platform*" - MHP) é um padrão proposto pela DVB ("*Digital Video Broadcast*"). Sua intenção é combinar televisão digital com a WWW [12, 13, 23, 24]. Os serviços básicos que a MHP provê são os guias eletrônicos de programação, teletexto econteúdo sincronizado com o vídeo [55].

A arquitetura básica da MHP é descrita na Figura 2.3 [11]. O sinal recebido é passado para o demodulador que, em seguida, é demultiplexado em três outros sinais: áudio, vídeo e dados. Cada um deles é decodificado e destinado a um módulo específico do sistema. No caso dos sinais de áudio

[2] "*EXtensible Markup Language*", um padrão aberto proposto pelo W3C para descrição de dados. Essa linguagem usa uma estrutura similar à HTML ("*HyperText Markup Language*"), i. e., com rótulos ("*tags*") [54]. Porém, os rótulos HTML descrevem como as informações são apresentadas, enquanto rótulos XML descrevem o que aqueles dados podem significar.

```
<smil>
 <head>
  <layout>
   <root-layout width="300" height="200"
        background-color="white" />
   <region id="vim_icon" left="75" top="50"
        width="32" height="32" />
   <region id="soja_icon" left="150" top="50"
        width="100" height="30" />
  </layout>
 </head>
 <body>
  <par>
   <img src="vim32x32.gif" alt="The vim icon"
        region="vim_icon" dur="6s" />
   <img src="madewithsoja.gif" alt="Made with SOJA"
        region="soja_icon" dur="6s" />
  </par>
 </body>
</smil>
```

Figura 2.2: Exemplo de código SMIL.

e vídeo eles são encaminhados para uma pequena CPU (Unidade Central de Processamento) que se encarrega de executá-los em hardware específico (e. g., caixas de som, telas) com a ajuda dos "*drivers*". No caso dos dados, eles são encaminhados para um software gerenciador de aplicações ou uma máquina virtual Java, se for o caso. Tais aplicações também interagem com a CPU para utilizar o hardware ou um canal de retorno para o provedor de serviços.

Deve-se perceber que o serviço de sincronização de conteúdo com vídeo se assemelha ao tipo de sistema que este trabalho relata. Contudo, há uma diferença fundamental. A MHP faz o sincronismo de informações complementares com um conjunto de cenas exibidas. No caso do hipervídeo proposto neste trabalho, a sincronização é feita com os objetos em cena, fornecendo uma maneira mais pontual de acessar informações complementares.

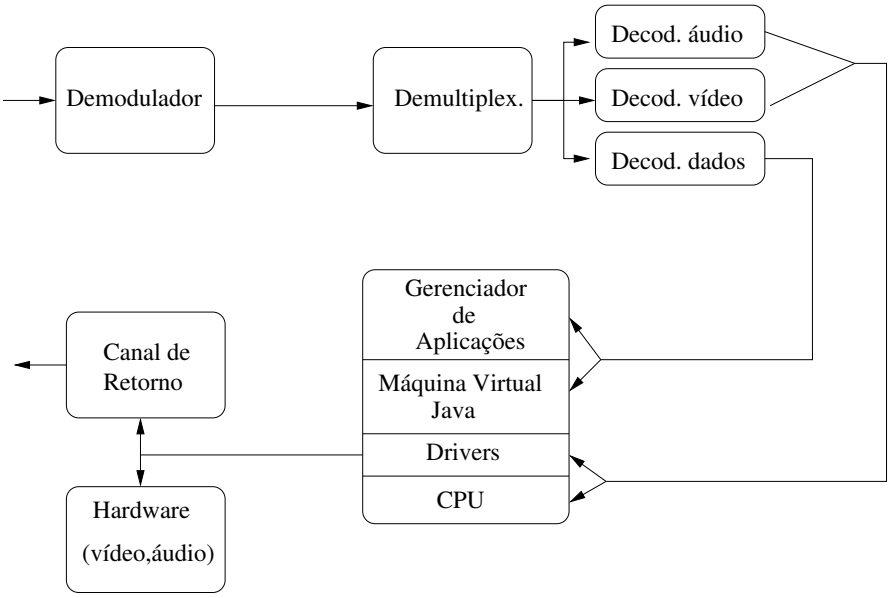


Figura 2.3: Arquitetura básica do MHP.

2.4 Sumário

Convergência digital foi o tema tratado neste capítulo. No início foi discutido o conceito de interatividade que dá suporte ao tema. Em seguida, o próprio conceito de convergência digital foi detalhado.

Depois algumas tendências em convergência digital, em particular entre TV e Internet, foram indicadas tendo como justificativa o chamado "*hype-cycle*", um modelo lançado pelo Gartner Group para prever a maturação, adoção e aplicação comercial de algumas tecnologias emergentes.

Finalmente, algumas das mais recentes propostas para convergência digital entre TV e Internet foram brevemente explicadas para delimitar a área de contribuição deste trabalho.

Portanto, a principal contribuição deste capítulo é a determinação do escopo dessa pesquisa, justificando a intenção de propor uma nova estratégia para TV interativa.

CAPÍTULO 3

A Família de Padrões MPEG

"*If you want to make an apple pie from scratch,*
you must first create the universe. "
Carl Sagan.

Este capítulo trata dos principais padrões abertos de compressão de vídeo usados atualmente: o MPEG ("*Motion Picture Experts Group*"). Na verdade, trata-se de uma família de padrões. Chama-se assim porque, à medida que novos requisitos eram propostos, novos padrões eram criados baseados nos anteriores. Essa abordagem culminou em um conjunto de padrões onde os mais recentes sempre mantêm a compatibilidade com os mais antigos, aprimorando-os.

Essa família de padrões foi concebida e é mantida pelo grupo de "*experts*" em imagens em movimento que empresta sua sigla para nomear os padrões. Esse grupo faz parte da ISO ("*International Standardization Organization*") uma organização responsável por vários padrões internacionais. Adotando a mesma abordagem da família MPEG, as linhas a seguir mostram as características de cada um dos padrões e o que cada um deles acrescentou em seu antecessor.

3.1 Algoritmos de Compressão

Antes de se apresentar o processo de codificação e decodificação de vídeo MPEG, convém mostrar os algoritmos de compressão envolvidos nesses processos.

Existem duas classes de algoritmos de compressão: os métodos de compressão sem perdas, nos quais toda a informação original é recuperada após a descompressão, e os métodos de compressão com perdas, nos quais parte da informação original é descartada na compressão. Essa parte da informação desperdiçada é, na verdade, redundante ou pouco significativa e por isso pode ser descartada.

Existe uma série de algoritmos de compressão sem e com perdas. Todavia, esse texto concentra-se especialmente nos algoritmos usados na família MPEG. A Seção 3.1.1 a seguir, mostra a codificação Huffman. Esse método de codificação pertence à classe de algoritmos de compressão sem perdas.

3.1.1 Codificação Huffman

Na codificação Huffman os dados de entrada são particionados em uma seqüência de símbolos para facilitar o processo de modelagem. A idéia é usar a probabilidade de ocorrência dos símbolos que aparecem em uma seqüência de entrada. Os códigos curtos são atribuídos aos blocos de entrada com altas probabilidades e códigos mais longos aos blocos com baixas probabilidades de ocorrência [42]. Quando usado em aplicações de compressão de imagem e vídeo, o alfabeto é restrito a 64.000 símbolos. O processo de codificação é descrito a seguir:

1. Ordenam-se os símbolos de acordo com as probabilidades de ocorrência.

2. Aplica-se um processo de contração aos dois símbolos com menor probabilidade de ocorrência.

3. Repete-se o passo anterior até que haja um único símbolo.

O processo de codificação Huffman pode ser visto como uma estrutura de dados do tipo árvore binária. Nas "folhas" da árvore estão os símbolos do alfabeto.

Para ilustrar o processo, considere o seguinte alfabeto $A = \{s_1, s_2, \ldots, s_N\}$ e as respectivas probabilidades de ocorrência $P(A) = \{p_1, p_2, \ldots, p_N\}$, s_k corresponde a $p_k, 1 \leq k \leq N$.

O primeiro passo é ordenar os símbolos de acordo com a probabilidade de ocorrência. Por simplicidade, no exemplo supõe-se que já estão ordenados, i. e., $p_1 \geq p_2 \geq \ldots \geq p_N$.

O segundo passo é aplicar um processo de contração de dois símbolos com menor probabilidade. Assume-se que no exemplo, os dois símbolos com menor probabilidade são s_{N-1} e s_N. O processo de contração resume-se em substituir esses dois símbolos por um símbolo hipotético, H_{N-1} cuja probabilidade é a soma das probabilidades $p_{N-1} + p_N$.

Pode-se afirmar que H_{N-1} é a raiz da sub-árvore cujas folhas são s_{N-1} e s_N. Agora, repete-se o segundo passo, considerando o alfabeto reduzido $A = \{s_1, s_2, \ldots, H_{N-1}\}$. A Figura 3.1 ilustra alguns passos do processo de contração.

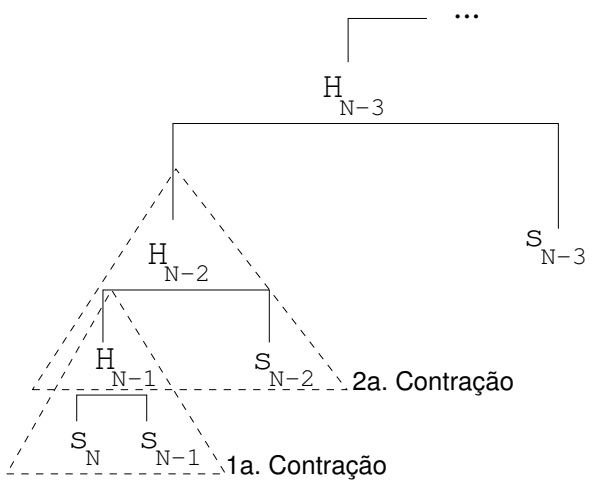


Figura 3.1: Parte da árvore de codificação Huffman de símbolos.

Ao final do processo de codificação tem-se uma palavra código para cada símbolo. Essa palavra código é composta por zeros e uns e indica que caminho percorrer na árvore para se obter o símbolo desejado. Os zeros indicam que deve seguir o caminho à esquerda. Os uns indicam que o caminho da direita deve ser o escolhido.

Tomando a Figura 3.1 como exemplo, para se obter o símbolo s_{N-1} a partir do nó H_{N-3}, a palavra código é 001. Para obter s_{N-2} a partir do mesmo nó a palavra código é menor: 01. Em suma, quanto maior a probabilidade de ocorrência de um símbolo, menor é a palavra código para acessá-lo. O tamanho médio das palavras código, que serve como medida de taxa de compressão, é $\sum t_i p_i$, t_i é o tamanho da palavra código para o símbolo s_i.

3.1.2 Decodificação Huffman

Existem dois métodos básicos de decodificação Huffman: decodificação baseada em tabela e serial. O primeiro tipo é rápido e a taxa de decodificação é constante, independente do tamanho da palavra código. Entretanto tem limitações de espaço uma vez que nesse método é necessária a criação de uma tabela de 2^T entradas (T é o tamanho da maior palavra código).

O segundo método, descrito a seguir, é mais adequado para aplicações com imagens. Ele possui uma taxa de entrada de bits fixa e uma taxa variável de saída de símbolos. Esse método consiste dos seguintes passos:

1. Lê-se o fluxo de bits de entrada bit por bit e percorre-se a árvore até que um nó folha seja alcançado.

2. Cada bit lido é descartado. Quando o nó folha for alcançado, o símbolo é exibido e volta-se ao nó raiz.

Considere o exemplo ilustrado na Figura 3.1 (supondo H_{N-3} como raiz) e a seguinte seqüência de bits: 0100100001000001. De acordo com a seqüência de bits proposta, a seqüência de símbolos de saída será: s_{N-2}, s_{N-1}, s_N.

3.1.3 Transformada Discreta do Co-seno

A seção anterior tratou da codificação Huffman, um método de compressão sem perdas. Nesta seção, a Transformada Discreta do Co-seno ou DCT[1] é abordada. A DCT é a transformada mais usada atualmente em algoritmos de compressão. Sua popularidade é justificada por obter uma compressão aproximada à Transformada de Karhunen-Loève (KLT). A KLT é uma técnica de transformada otimizada porque minimiza o erro mínimo quadrático[2], porém exige uma alta complexidade computacional [40]. As transformadas, nesse caso, são funções que transformam um sinal do domínio espacial para o domínio da freqüência.

[1] "*Discrete Cosine Transform*". Optou-se pelo uso dessa sigla para manter a compatibilidade com as publicações internacionais.

[2] O erro mínimo quadrático é uma técnica que evidencia os erros de aproximação.

A utilização da DCT é justificada a partir da observação que "*pixels*"[3] vizinhos são, na maioria das vezes, correlacionados. Portanto, a idéia é usar uma transformada que concentre a aleatoriedade desses "*pixels*" em poucos parâmetros não relacionados [50].

A computação básica utilizada na implementação da DCT estabelece a transformação de um bloco de imagem de $N \times N$ "*pixels*" do domínio espacial para o domínio da freqüência. No caso de sistemas computacionais, o melhor valor para N é 8 $(8 = 2^3)$.

Uma operação da DCT em duas dimensões para um bloco de 8×8 "*pixels*" gera um bloco composto por 8×8 coeficientes que representam um valor ponderado para cada um dos 64 padrões ortogonais. Esses padrões, quando "somados" formam a imagem original.

A maioria dos algoritmos DCT em duas dimensões pode ser classificada em duas categorias: a abordagem indireta (linha-coluna) ou a abordagem direta. Na abordagem indireta, pode-se aplicar a DCT em uma dimensão para linhas e depois aplicar para as colunas. Essa abordagem é possível porque a DCT é uma transformada ortogonal separável, i. e., pode-se aplicá-la para uma dimensão a cada instante. Na abordagem direta, o algoritmo é aplicado para cada elemento separadamente.

A abordagem indireta é menos eficiente porque antes de se aplicar o algoritmo em uma dimensão, é necessário aguardar o resultado da transformada na dimensão anterior. A abordagem direta é mais eficiente, mas requer hardware mais complexo [40].

A equação que efetua a DCT é descrita a seguir:

$$y_{kl} = \frac{c(k)c(l)}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} x_{ij} cos\left(\frac{(2i+1)k\pi}{16}\right) cos\left(\frac{(2j+1)l\pi}{16}\right) \tag{3.1}$$

em que (k, l), as coordenadas da freqüência no domínio da amostra, e (i, j), as coordenadas espaciais no domínio da transformada, podendo assumir valores inteiros de 0 e 7. Ainda,

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{se } k = 0 \\ 1 & \text{caso contrário.} \end{cases} \tag{3.2}$$

Na forma de matriz, a Equação 3.1 pode ser escrita da seguinte forma:

$$[Y]_{8\times8} = [T]_{8\times8}[X]_{8\times8}[T^t]_{8\times8} \tag{3.3}$$

[3] Abreviatura de "*Picture Elements*" em inglês, correspondente a cada ponto que forma uma imagem.

em que as linhas da matriz $[T]$ são os vetores base da DCT e $[T^t]$ é a matriz transposta de $[T]$.

O Processo de Quantização

Após a aplicação da DCT, o próximo passo é a quantização. O processo de quantização é irreversível, uma vez que descarta alguns pesos, i. e., zera alguns elementos da matriz. Esse processo é que faz com que a DCT seja um esquema de compressão com perdas.

O processo de quantização pode ser uniforme ou não-uniforme. A quantização uniforme é implementada como uma operação de divisão por um "passo de quantização" seguido por uma operação de arredondamento. A quantização não-uniforme é implementada a partir de um algoritmo de busca em uma tabela. Esse tipo de quantização é bem mais complexo e por isso onera muito o tempo de processamento. Portanto, o processo uniforme é o mais comum.

O processo de quantização uniforme de y_{kl} é expresso de acordo com a Equação 3.4 a seguir:

$$z_{kl} = round\left(\frac{y_{kl}}{q_{kl}}\right) = \left\lfloor \frac{y_{kl} \pm \lfloor \frac{q_{kl}}{2} \rfloor}{q_{kl}} \right\rfloor \tag{3.4}$$

Na equação anterior, q_{kl} denota o elemento kl da matriz de quantização Q e $\lfloor x \rfloor$ denota o maior inteiro menor ou igual a x. Convém ressaltar que se $y_{kl} \geq 0$ então os termos do numerador são somados e, caso contrário, são subtraídos.

A matriz Q depende das características visuais do sistema e de considerações relativas às taxas de compressão. Os padrões de compressão provêem valores para a matriz Q.

O processo inverso da quantização, representado neste texto por Q^{-1}, é descrito na Equação 3.5.

$$\hat{z}_{kl} = z_{kl} q_{kl} \tag{3.5}$$

A princípio, o processo de quantização é feito após a DCT. Porém, em [14] é proposto um método que embute o "passo de quantização" na rotina DCT, aumentando a eficiência.

3.1.4 Transformada Inversa Discreta do Co-seno

A transformada inversa discreta do co-seno (IDCT), como o próprio nome afirma, faz o inverso da DCT. Em outras palavras, a IDCT faz a transformação de coordenadas no domínio da freqüência para o domínio espacial. A equação é descrita a seguir:

$$x_{ij} = \sum_{k=0}^{7} \sum_{l=0}^{7} y_{kl} \frac{c(k)c(l)}{4} cos\left(\frac{(2i+1)k\pi}{16}\right) cos\left(\frac{(2j+1)l\pi}{16}\right) \tag{3.6}$$

3.2 Conceitos Gerais dos Padrões MPEG

Considera-se um vídeo como uma seqüência de quadros numerados onde cada quadro é uma imagem estática. Um dispositivo de apresentação de vídeo mostra um quadro após o outro a uma taxa próxima a 30 quadros por segundo.

Um vídeo no formato MPEG é não-entrelaçado (progressivo). Em vídeos entrelaçados, como aqueles gerados pelas câmeras de TV, cada quadro é composto por campos superiores (linhas pares) e inferiores (linhas ímpares). Os superiores são iluminados primeiro e, em seguida, os campos inferiores. Em vídeos progressivos, não há o conceito de campo. As linhas são iluminadas seqüencialmente. Sendo assim, para codificar um vídeo no padrão MPEG, é preciso que haja uma conversão prévia do formato entrelaçado para o progressivo.

A codificação de vídeo na família MPEG utiliza esquemas de compressão com perdas e sem perdas. No primeiro tipo, nem toda a informação presente na imagem original é preservada na compressão. Isso significa que imagens descomprimidas utilizando tal esquema são reconstruídas com imperfeições.

O esquema com perdas desconsidera detalhes imperceptíveis pelo usuário para tentar otimizar a compressão das imagens. Adicionalmente, tal esquema tenta explorar a redundância espacial, i. e., "*pixels*" correlatos no espaço. Por sua vez, a descompressão sem perdas restaura totalmente a informação antes comprimida.

Além da redundância espacial, a família MPEG também explora o conceito de redundância temporal. Essa segunda forma de redundância é percebida em uma seqüência de imagens, ou seja, entre uma imagem e outra há regiões presentes na mesma posição ou deslocadas da posição original.

Portanto, ao invés de codificar novamente a informação sobre tais regiões, basta tratar o vetor de movimento. Esse vetor contém as coordenadas da nova posição daquela região na cena.

Para facilitar o cálculo do vetor de movimento das regiões que compõem a cena, padronizou-se a divisão dos quadros em áreas de 16 por 16 "*pixels*", designadas como macroblocos. A escolha de macroblocos com essas dimensões é justificada pela facilidade em tratá-los computacionalmente. Ressalta-se que o cálculo do vetor de movimento é uma das tarefas que demandam um maior tempo computacional [4, 8].

Os quadros nos padrões MPEG são classificados basicamente em três tipos. Os intraquadros (quadros I) não referenciam nenhum outro quadro, ou seja, neles está toda a informação necessária para sua compressão. Por ter essa característica, esses quadros têm o menor grau de compressão entre os demais.

Os quadros preditivos (quadros P) são codificados usando predição de movimento a partir de quadros I ou outros quadros P. A predição de movimento é um processo de cálculo das diferenças entre quadros que envolve a computação do vetor de movimento. Os quadros P têm um grau de compressão melhor do que os quadros I.

Os quadros bidirecionalmente preditivos (quadros B) possuem o mais alto grau de compressão. Eles referenciam quadros I e P usando a técnica de predição de movimento. A Figura 3.2 exemplifica uma distribuição desses quadros em uma seqüência.

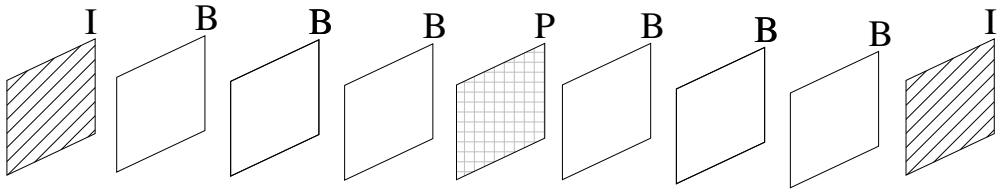


Figura 3.2: Seqüência de quadros no esquema de codificação MPEG.

Resumindo, a codificação de vídeo digital nos padrões MPEG envolve os seguintes passos [50]:

1. Redução da taxa de amostragem nos domínios espacial e temporal nos componentes de luminância[4], representada por Y, e crominância[5], representada por Cb e Cr. Tais componentes

[4]Quociente da intensidade luminosa de uma superfície pela área aparente dessa superfície.
[5]Informação sobre a cor. Cb é o sinal de diferença da cor azul. Cr é o sinal de diferença da cor vermelha.

formam o espaço de cores $YCbCr$.

2. Aplicação da DCT baseada em blocos nos quadros ("*intraframes*") e entre os quadros ("*interframes*").

3. Compensação de movimento baseada em blocos nos quadros preditivos (P e B) e interpolativos (I).

4. Codificação Huffman para compressão sem perdas dos vetores de movimento e dos coeficientes DCT quantizados.

Apesar dessa estrutura de quadros (I-B-P) facilitar a exibição convencional do vídeo, ela torna mais complexa outras funcionalidades, como acesso aleatório, avanço rápido e retrocesso, típicas em aparelhos de vídeo-cassete [34].

3.2.1 Processo de Codificação de Vídeo Digital no MPEG

O processo de codificação de vídeo de acordo com a família de padrões MPEG é ilustrado na Figura 3.3, a seguir.

A entrada é uma seqüência de bits. O pré-processamento não é especificado nos padrões, mas envolve, eventualmente, conversão de cores do formato RGB[6] para $YCbCr$, tradução do formato (entrelaçado para progressivo), pré-filtragem, entre outros.

Após o processamento inicial, há uma seleção do tipo de codificação em função do tipo de quadro (I, P ou B). Sendo um quadro I, não se faz necessária a estimação ou compensação de movimento já que esse quadro não faz referência a nenhum outro. Cada macrobloco é codificado usando-se DCT. Em seguida, os coeficientes são quantizados e depois codificados usando-se um código de tamanho variável (VLC), como o código de Huffman, por exemplo.

Após o passo de quantização, os blocos quantizados percorrem o caminho inverso, ou seja, é feita uma quantização inversa (Q^{-1}) e aplica-se a transformada inversa do co-seno. Essa cópia reprocessada da imagem é armazenada na memória de quadro para ser usada futuramente na codificação preditiva.

[6]Formato que codifica as cores básicas vermelho ("*Red*"), verde ("*Green*") e azul ("*Blue*").

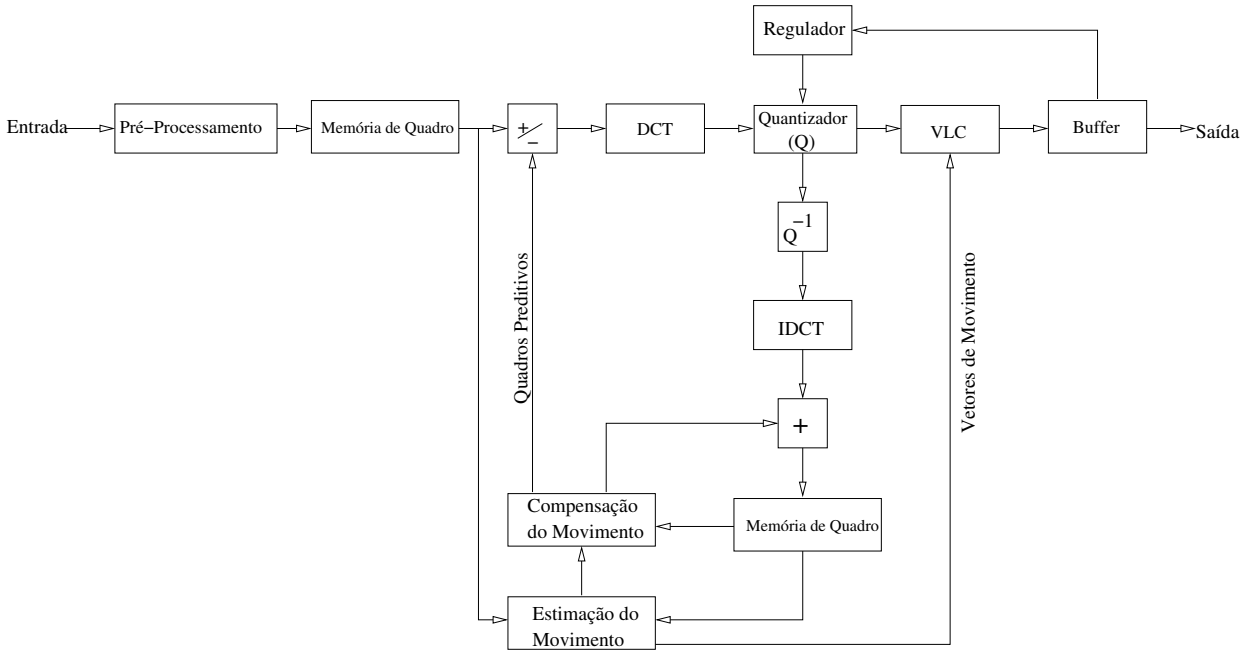


Figura 3.3: Processo de codificação MPEG.

Se a entrada for um quadro P ou B, então os macroblocos não são codificados diretamente. Ao invés disso, os erros de predição são codificados. Esse processo para quadros P, ilustrado na Figura 3.4 a seguir, é chamado codificação preditiva intraquadros.

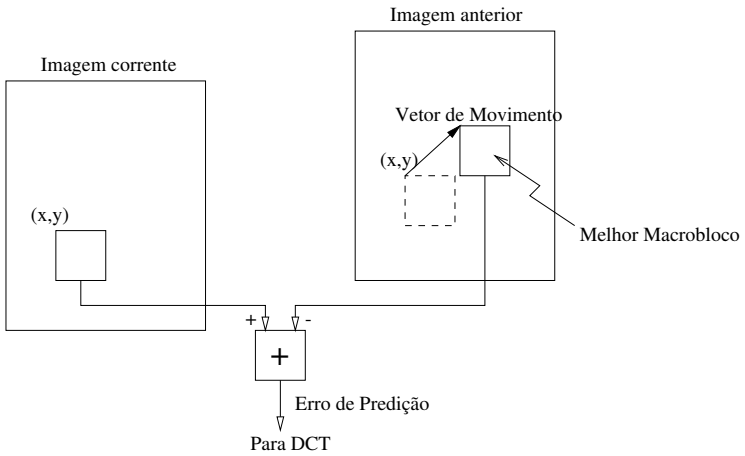


Figura 3.4: Processo de Estimação de Movimento para Quadros P.

Em quadros P, para cada macrobloco na imagem atual, um algoritmo calcula o vetor de movimento que indica as coordenadas do macrobloco que melhor casa suas características na área de busca na

imagem anterior. Os macroblocos são subtraídos e a diferença é codificada usando-se DCT.

Nos quadros B, o processo de estimação de movimento, ilustrado na Figura 3.5, é feito duas vezes: uma para imagens anteriores e outra para imagens posteriores. Isso gera dois vetores de movimento. Dessa forma, o codificador pode obter o macrobloco de predição de erro de um dos dois macroblocos gerados ou de uma média dos dois. O erro de predição é posteriormente codificado usando-se a DCT baseada em blocos. Tal processo é chamado codificação interpolativa intraquadros.

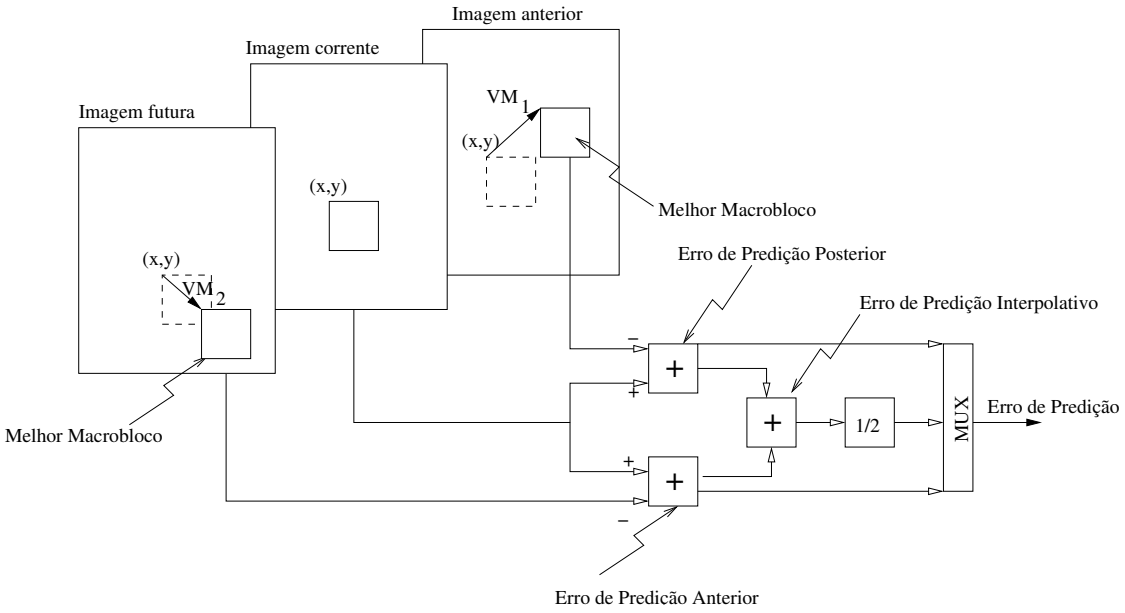


Figura 3.5: Processo de Estimação de Movimento para Quadros B.

Os coeficientes de erro de predição quantizados depois de aplicada a DCT, junto com os vetores de movimento, são multiplexados e codificados usando-se um código de tamanho variável (codificação Huffman).

Finalmente, é preciso fazer uma reordenação das imagens de entrada para que elas sejam enviadas de forma correta para o decodificador. A desordem é causada devido à predição bidirecional. Em outras palavras, os quadros B dependem dos quadros I e P, portanto eles só podem ser codificados depois dos quadros I e P. Ressalte-se que essa ordem dos quadros ainda não é a mesma ordem em que eles serão mostrados.

3.2.2 Processo de Decodificação de Vídeo Digital no MPEG

O processo de decodificação de vídeo nos padrões MPEG é similar ao processo que restaura os quadros P e B na codificação. Em suma, as imagens são decodificadas usando-se o algoritmo de Huffman e seu tipo identificado a partir da informação contida no cabeçalho. Para cada macrobloco, os coeficientes são dequantizados e traduzidos para o domínio espacial usando-se IDCT. A Figura 3.6 apresenta o processo de decodificação.

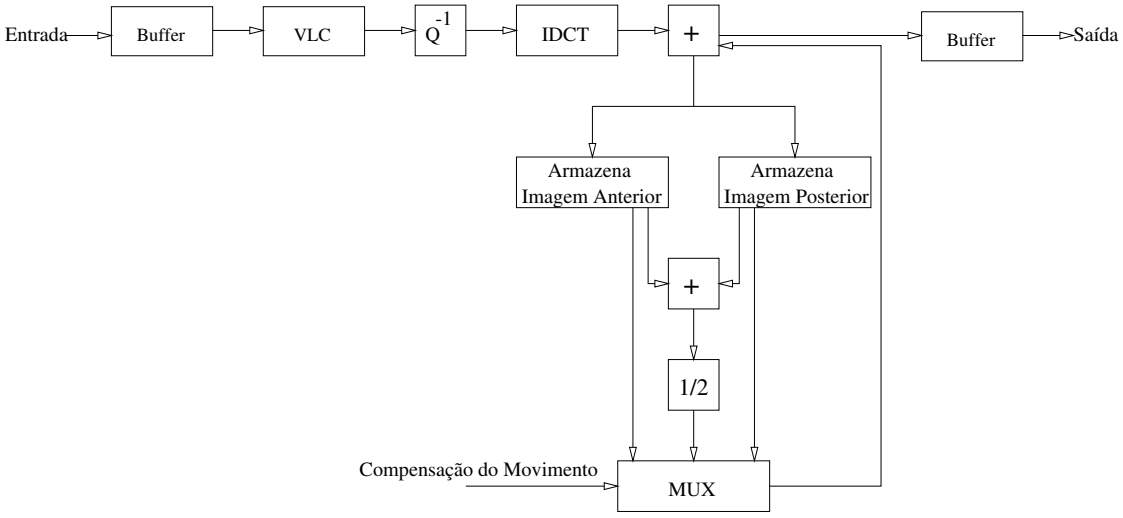


Figura 3.6: Processo de Decodificação de Vídeo.

De forma análoga à codificação, os quadros I, P e B determinam quais os passos que serão executados. Para os quadros I não há compensação de movimento. Sendo assim, utiliza-se a IDCT, salva-se como imagem anterior e armazena-se no "*buffer*" de saída.

Sendo um quadro P, para cada macrobloco utiliza-se a IDCT e faz-se a compensação de movimento. A compensação envolve adicionar a área indicada pelo vetor de movimento na imagem anterior na saída do IDCT. A imagem reconstruída é salva no armazenamento de imagem posterior.

Quando for um quadro B, computa-se a IDCT e executa-se uma compensação de movimento bidirecional. Essa compensação é feita usando-se dois vetores de movimento, acessando os valores de "*pixels*" das regiões correspondentes nas imagens anterior e posterior e computando-se a predição para o quadro. Soma-se a IDCT de saída com a predição.

3.2.3 Sintaxe do MPEG-1 para o Fluxo de Bits de Vídeo

Uma seqüência de vídeo no MPEG-1 é um fluxo ordenado de bits, com padrões especiais de bits marcando o início e o fim de uma seção lógica [25].

A sintaxe, i. e., a forma como os bits são interpretados, definida no MPEG-1 é representada em seis camadas, a saber: Camada de Seqüência; Camada Grupo de Imagens; Camada de Imagem; Camada de Pedaço; Camada de Macrobloco e Camada de Bloco. O relacionamento hierárquico entre as camadas é mostrado na Figura 3.7. Cada uma delas será descrita a seguir.

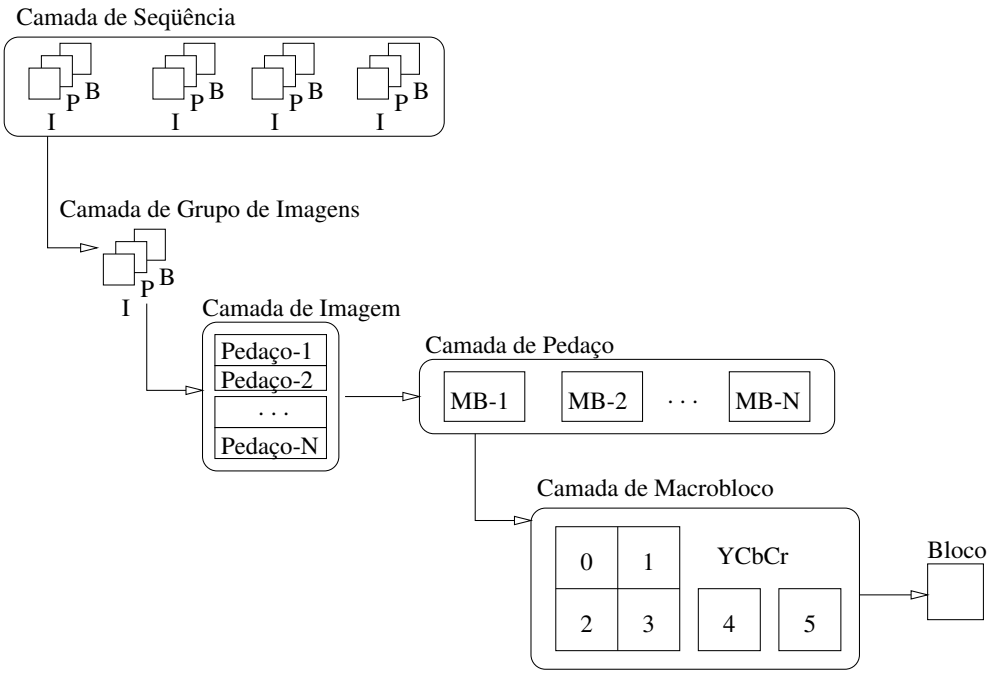


Figura 3.7: Sintaxe do MPEG-1.

Camada de Seqüência

A Camada de Seqüência é formada por um cabeçalho, seguido por um ou mais grupos de imagens, terminado por um código de fim de seqüência. As informações inclusas no cabeçalho são os tamanhos horizontal e vertical de cada imagem, a taxa de imagens por segundo, resolução do pixel, a taxa de bits em unidades de 400 bits por segundo e o tamanho mínimo necessário para o "*buffer*" do decodificador. O cabeçalho também pode incluir as matrizes de quantização para as imagens DCT.

Camada de Grupo de Imagens

Um Grupo de Imagens (GDI) é um conjunto de imagens em ordem de apresentação. Esse conjunto deve conter pelo menos um quadro I, deve começar com quadros I ou B e terminar com quadros I ou P. Se a primeira imagem for um quadro I ou B que não dependem do GDI anterior, então esse GDI pode ser codificado e apresentado independentemente de outro. Esse tipo particular de GDI é chamado GDI fechado.

O cabeçalho da camada GDI inclui informação de temporização, uma indicação para GDI fechado, entre outras informações do usuário.

Camada de Imagem

A Camada de Imagem define a informação de codificação para cada imagem. Em função da possibilidade de reordenamento dos quadros P e B, o cabeçalho provê uma referência (número) que permite definir a ordem de apresentação. Outras informações inclusas no cabeçalho são o tipo de quadro, sincronização, resolução e alcance dos vetores de movimento.

Camada de Pedaço

Cada imagem é dividida em pedaços que podem ser tão grandes quanto toda a imagem ou tão pequenos quando um macrobloco. Essa divisão pode ser útil em caso de perda de dados. Ao invés de descartar toda uma imagem por causa de corrupção em alguns dados, descarta-se apenas um pedaço.

Um cabeçalho dessa camada possui informação sobre sua posição na figura e um fator de quantização entre 1 e 31. Esse fator pode ser usado para dequantizar coeficientes DCT.

Camada de Macrobloco e Bloco

Um pedaço é dividido em macroblocos. O cabeçalho de um macrobloco inclui o tipo do macrobloco, informação da posição, códigos para os vetores de movimento horizontal e vertical, e quais blocos em um macrobloco são codificados e transmitidos.

3.3 Particularidades dos Padrões MPEG

À medida em que novos requisitos eram estabelecidos, um novo padrão era anexado à família MPEG. Esse novo padrão deveria manter a compatibilidade com os já existentes. Nesta seção são mostradas as particularidades de cada um dos padrões, i. e. o que cada padrão apresenta de diferente do seu antecessor.

O "primogênito" dessa família foi o MPEG-1 cujos conceitos já foram mencionados nas seções anteriores. Entretanto, com o MPEG-1 não era possível garantir qualidade de vídeo para aplicações do tipo "*broadcast*". O MPEG-2, descrito na próxima seção, foi seu sucessor.

3.3.1 MPEG-2

O MPEG-2, a segunda fase do grupo de trabalho da ISO, tem como objetivo definir um padrão genérico que dê suporte a taxas de fluxos de dados próximas a 5 Mbit/s para qualidade NTSC[7] ou PAL[8] e próximas a 10 Mbit/s para qualidade de estúdio [1]. Os requisitos originais do MPEG-2 são:

- Compatibilidade com o MPEG-1, i. e., um decodificador MPEG-2 deve decodificar um fluxo MPEG-1.
- Boa qualidade de imagem.
- Flexibilidade no formato dos dados de entrada.
- Capacidade para acesso aleatório.
- Capacidade para "*slow motion*", avanço rápido de cenas e execução reversa (de trás para frente).
- Escalonabilidade de fluxo de bits, i. e., capacidade para descartar porções do fluxo de dados e ainda assim obter uma razoável qualidade na imagem.
- Baixo atraso em comunicação bidirecional.

[7] Sigla para "*National TV Standards Committee*", um padrão de TV em cores desenvolvido nos EUA que transmite 30 quadros entrelaçados por segundo em 525 linhas de resolução. O sinal é um conjunto composto de vermelho, verde e azul e inclui áudio modulado em freqüência.

[8] Sigla para "*Phase Alternating Line*", um padrão de TV em cores desenvolvido na Alemanha, que transmite 25 quadros entrelaçados por segundo em 625 linhas de resolução. No Brasil usa-se PAL-M, cuja transmissão é de 30 quadros por segundo.

- Capacidade de recuperação de falhas ao nível dos bits.

Apesar de todos esses requisitos, o grupo MPEG observou que nem todas as aplicações usariam todas as possibilidades oferecidas pelo padrão. Além disso, seria difícil definir um único padrão que desse suporte a todos os requisitos. Outra questão levantada refere-se à necessidade de restringir a taxa de bits codificados em 10 Mbit/s, assim como saber se é possível obter taxas entre 80 e 100 Mbit/s usadas em aplicações de TV de alta definição.

Com base nessas observações, o grupo optou por dividir a codificação MPEG-2 em perfis e níveis. Em cada perfil/nível, o MPEG-2 provê uma sintaxe para o fluxo de bits codificados e para os requisitos de decodificação.

Um perfil é um subconjunto da sintaxe do fluxo de dados. Dentro de um perfil, um nível é definido como um conjunto de restrições imposto pelos parâmetros do fluxo de bits, tais como resolução da imagem ou taxa máxima de transmissão.

A sintaxe do MPEG-2 tem duas categorias. A sintaxe não escalonável é um conjunto que engloba a sintaxe de codificação do MPEG-1 com extensões adicionais para codificação de sinais entrelaçados. A sintaxe escalonável permite uma codificação em camadas do fluxo de vídeo. Cada uma das camadas possui um nível de qualidade diferente.

Imagens Entrelaçadas

O padrão MPEG-2 deve suportar imagens entrelaçadas e não entrelaçadas (progressivas). Em um quadro com imagens entrelaçadas existem dois campos, cada um contendo metade das linhas do quadro, conforme ilustrado na Figura 3.8.

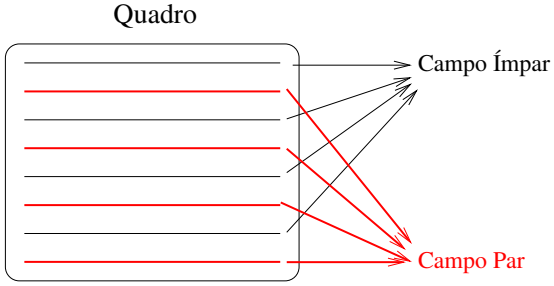


Figura 3.8: Disposição dos campos em um quadro.

As imagens entrelaçadas podem ser codificadas separadamente (imagens campo) ou codificados como uma única imagem (imagens quadro). A segunda forma de codificação já é especificada no padrão MPEG-1.

Tratando-se de codificação de imagens campo, se a primeira imagem de um quadro codificado é uma imagem I-campo, então a segunda imagem pode ser uma I-campo ou uma P-campo. Em outras palavras, o primeiro campo de uma imagem pode ser usado como um preditor para o segundo campo. Se uma imagem for uma P-campo ou B-campo, então a segunda imagem campo deve ter o mesmo tipo da primeira imagem campo.

Formatos de Subamostragem de Cores

As imagens no padrão MPEG-1 são codificadas no espaço de cores $YCbCr$, com macroblocos no formato 4:2:0 (4 blocos para Y, 2 blocos para Cb e zero blocos para Cr). O MPEG-2, por sua vez, suporta ainda os macroblocos nos formatos 4:4:2 e 4:4:4.

Modos de Predição

Uma seqüência de imagens no MPEG-2 pode ser uma coleção de quadros de imagens ou uma coleção de campos de imagens. Sendo assim, duas classes de predição são suportadas: predição de quadros ou predição de campos. A Figura 3.9 ilustra as diferenças.

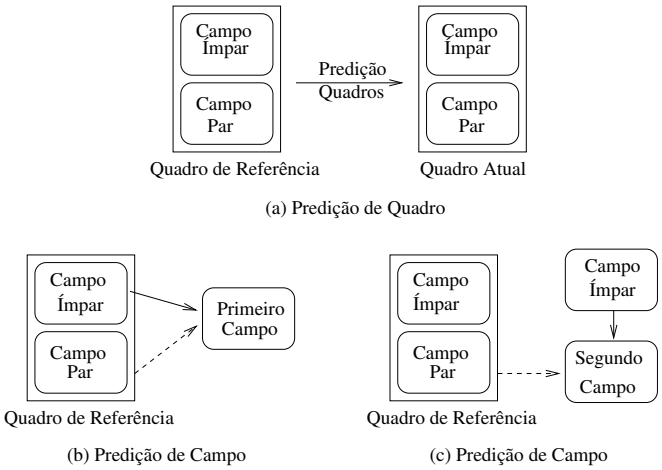


Figura 3.9: Modos de predição de quadros para imagens quadros.

Em imagens campo, as predições são feitas independentemente para cada campo a partir dos campos de referência. Isso é similar ao que é feito no MPEG-1, onde cada campo é considerado uma imagem independente.

Em predição de campo, cada quadro é tratado como dois campos separados. Um vetor de movimento pode apontar para um campo em outro quadro de referência ou para um campo no quadro atual. A Figura 3.9b ilustra um exemplo onde o primeiro campo pode ser preditivo a partir do campo ímpar ou do campo par.

Já no exemplo mostrado na Figura 3.9c, o segundo campo (campo par) pode ser predito a partir do campo par do quadro anterior ou do campo ímpar do quadro corrente.

Compensação de Movimento

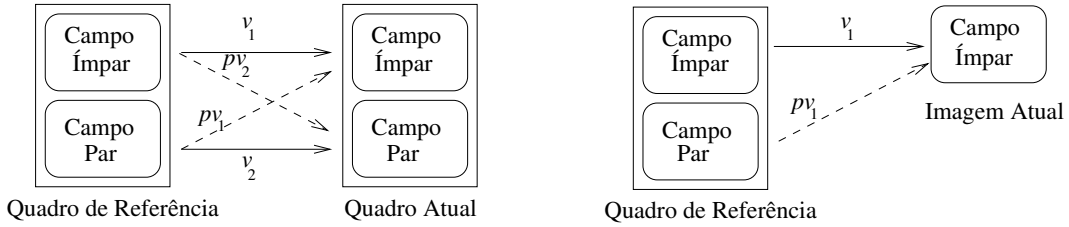
Em MPEG-2 todos os vetores de movimento são especificados em uma resolução de meio "*pixel*". Matematicamente, seja (u, v) o vetor de movimento dos componentes de luminância em um macrobloco. Em função da subamostragem dos componentes croma, os vetores de movimento são dados por $(\frac{u}{2}, \frac{v}{2})$. para o formato 4:2:0 e por $(\frac{u}{2}, v)$ para o formato 4:2:2. No formato 4:4:4, os componentes croma usam os vetores de movimento do componente de luminância na mesma escala.

A compensação de movimento no MPEG-2 utiliza dois modos. O modo de compensação de movimento 16×8 permite um macrobloco de 16×16 ser tratado como duas regiões - ímpar e par - ambas de 16×8. Em cada uma das regiões é aplicada a compensação de movimento independentemente.

O modo de compensação "*Dual Prime*" é aplicado apenas em imagens-P e em grupos de imagens que não têm imagens-B entre as imagens-P e os quadros de referência. Nas imagens quadro, em cada campo existem dois vetores de movimento associados. A Figura 3.10 mostra essa associação, onde os vetores de movimento v_1 e pv_1 são associados ao campo ímpar, enquanto os vetores v_2 e pv_2 são associados ao campo par do quadro atual.

Sejam $v_1(x)$, $pv_1(x)$, $v_2(x)$ e $pv_2(x)$ os componentes horizontais dos vetores de movimento v_1, pv_1, v_2 e pv_2. Os componentes verticais são expressos em função de y e os vetores de movimento pv_1 e pv_2 são derivados de v_1 e v_2 da seguinte forma:

$$pv_1(x) = \frac{v_1(x)}{2} + \delta(x) \tag{3.7}$$



(a) Predição de campo em imagens quadro

(b) Predição de campo em imagens campo

Figura 3.10: Modo de predição "*Dual Prime*"

$$pv_1(y) = \frac{v_1(y)}{2} - 1 + \delta(y) \tag{3.8}$$

$$pv_2(x) = \frac{3 \times v_2(x)}{2} + \delta(x) \tag{3.9}$$

$$pv_2(y) = \frac{3 \times v_2(y)}{2} + 1 + \delta(y) \tag{3.10}$$

onde $\delta(x)$ e $\delta(y)$ são vetores de movimento diferencial que podem assumir os valores -1, 0 ou 1.

DCT e Quantização

No padrão MPEG-1 são usadas duas matrizes de quantização para os coeficientes DCT: uma para os intrablocos e outra para os blocos não-intra. Nos formatos 4:2:2 e 4:4:4, o MPEG-2 suporta matrizes de quantização diferentes para os componentes de luminância e crominância. Sendo assim, tem-se duas matrizes para blocos de luminância (intra e não-intra) e outras duas para blocos de crominância (intra e não-intra). A quantização dos coeficientes AC é similar à do processo do MPEG-1, descrito na equação 3.11:

$$QDct[i,i] = \frac{8Dct[i,j]}{qQ[i,j]} \tag{3.11}$$

onde $QDct[i,j]$ representa os coeficientes AC quantizados, $Dct[i,j]$ é a saída da função DCT, $Q[i,j]$ é a matriz de quantização e q é um fator de quantização.

Fluxos de Bits Escalonáveis

O padrão MPEG-2 permite uma representação em camadas do fluxo de bits codificado. Os quatro modos básicos de escalonabilidade de fluxo de bits são:

- Particionamento de Dados: o fluxo de dados é particionado em duas camadas (partições). A primeira camada pode conter os cabeçalhos e vetores de movimento, enquanto a segunda pode conter o resto do fluxo de bits. Esse modo pode ser usado em aplicações que podem alocar dois canais para um único fluxo de dados.

- Escalonabilidade SNR[9]: útil em aplicações que suportam transmissão de vídeo em diferentes níveis de qualidade. Todos os níveis possuem a mesma resolução espacial, mas qualidades diferentes de vídeo. A camada inferior provê um vídeo de qualidade razoável. As camadas superiores melhoram a qualidade provendo mais dados para os coeficientes DCT na camada inferior.

- Escalonabilidade Espacial: o fluxo de bits é dividido em camadas de resolução espacial diferentes. A camada inferior provê resolução espacial básica. As demais camadas podem usar a camada inferior interpolada espacialmente para prover um fluxo de vídeo com resolução espacial total.

- Escalonabilidade Temporal: a camada inferior provê uma taxa temporal básica enquanto as camadas superiores são codificadas com predição temporal relativa à camada inferior. Todas as camadas têm os mesmos tamanhos de quadros e formatos de crominância, mas diferentes taxas de quadros.

3.3.2 MPEG-4

O maior objetivo dos padrões MPEG-1 e 2 foi tornar a transmissão e armazenamento de áudio e vídeo digitais eficientes. O objetivo do MPEG-4 é padronizar algoritmos para codificação audiovisual em aplicações multimídia, permitindo interatividade, alta compressão, escalonabilidade e suporte para conteúdo de áudio e vídeo, utilizando o paradigma de objetos.

[9] "*Signal to Noise Ratio*", traduzido como relação sinal-ruído.

O modelo até então usado nos padrões MPEG-1 e 2 era formado por quadros com áudio associado. O MPEG-4 define uma cena audiovisual como uma representação codificada de objetos audiovisuais que possuem certas relações no espaço e no tempo [28, 41].

Os objetos audiovisuais podem ser um objeto de vídeo em uma cena ou a imagem de fundo. Podem ser também um objeto de áudio, e. g., uma voz, um instrumento em uma banda ou até um latido ao fundo.

A utilização desse novo paradigma de representação da informação facilita a reutilização de dados, oferece flexibilidade, permite o uso inteligente da banda disponível entre outras melhorias.

A arquitetura do padrão MPEG-4 funciona da seguinte forma. No codificador, os objetos e suas relações espaciais e temporais geram fluxos de bits codificados. Esses fluxos são multiplexados com outros objetos armazenados e transmitidos [30].

No decodificador, o módulo compositor usa as relações espaço-temporais e, eventualmente, interações com o usuário para gerar a cena. Antes da transmissão dos objetos, o codificador e decodificador podem trocar informações de configuração. Isso pode ajudar o decodificador a determinar a classe de algoritmos, ferramentas e outros objetos necessários para o processo de decodificação dos objetos. A Figura 3.11 demonstra tal arquitetura.

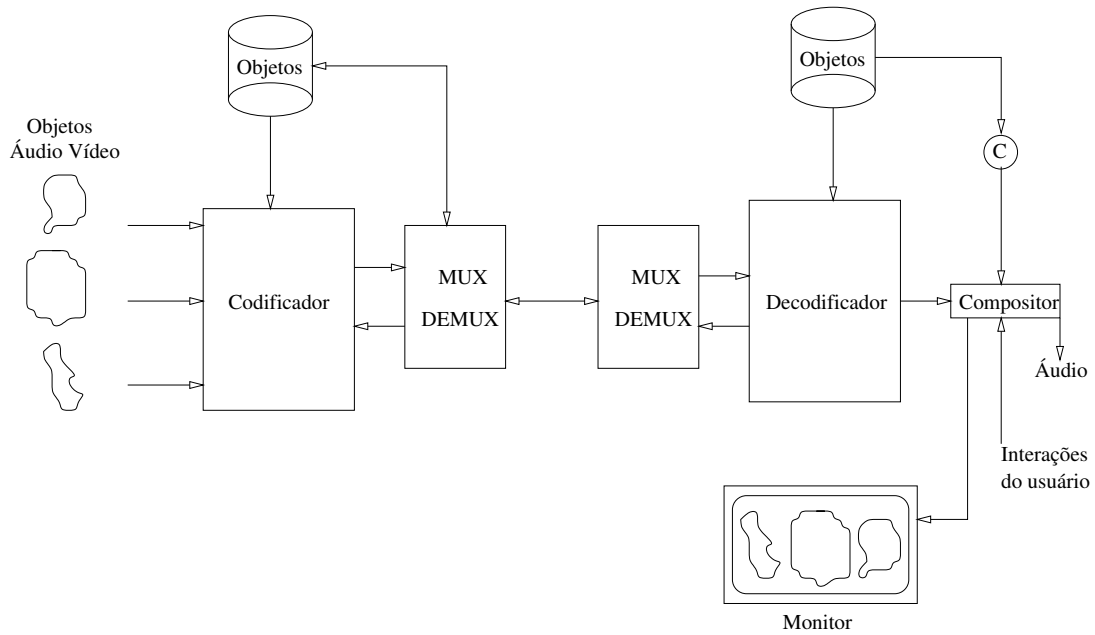


Figura 3.11: Arquitetura do padrão MPEG-4

Codificação de Vídeo no MPEG-4

A informação também tem uma representação hierarquizada no MPEG-4. Cada quadro de vídeo é segmentado em uma quantidade de regiões de imagem chamadas de planos de objetos vídeo (VOP[10]). O processo de segmentação não é especificado pelo padrão.

VOPs sucessivos que pertencem a um mesmo objeto físico em uma cena são chamados objeto vídeo (VO[11]). Os VOs são equivalentes aos GOPs nos padrões MPEG-1 e 2. As informações sobre forma, movimento e textura dos VOPs que pertencem a um mesmo VO são codificados em camadas de objeto vídeo (VOL[12]) separadas.

As informações relevantes necessárias para identificar cada uma das VOLs e a maneira como as VOLs são compostas são incluídas na própria VOL. Isso permite uma decodificação seletiva dos VOPs e provê escalonabilidade no decodificador.

Codificação da informação para cada VOP

Para cada VO, as informações sobre forma, movimento e textura dos VOPs são codificadas. A informação sobre a forma é denominada planos alfa. As técnicas adotadas pelo padrão vão prover codificação sem perdas dos planos alfa e codificação com perdas das informações de transparência e forma. Após codificar as informações sobre a forma do VOP, cada imagem VOP em um VO é particionada em macroblocos não sobrepostos.

As redundâncias temporais verificadas entre planos de vídeos diferentes em um mesmo objeto de vídeo são exploradas usando estimação e compensação de movimento baseados em blocos. A Figura 3.12 ilustra tal processo de codificação da compensação e estimação de movimento entre VOPs variando-se a localização, tamanho e forma.

A janela de referência é a borda da imagem de fundo original. O deslocamento é um parâmetro codificado para indicar a localização do VOP em relação às bordas da janela de referência. A janela VOP contém um número inteiro de macroblocos (cada macrobloco tem 16×16 "*pixels*").

Os macroblocos podem ser de dois tipos: no macrobloco padrão todos os "*pixels*" estão dentro da área ativa do VOP; nos macroblocos de contorno, onde alguns dos "*pixels*" estão fora da área ativa do

[10]Será usada a sigla VOP ("*Video Object Plane*") para manter a compatibilidade com publicações estrangeiras.
[11]Será usada a sigla VO ("*Video Object*") para manter a compatibilidade com publicações estrangeiras.
[12]Será usada a sigla VOL ("*Video Object Layer*") para manter a compatibilidade com publicações estrangeiras.

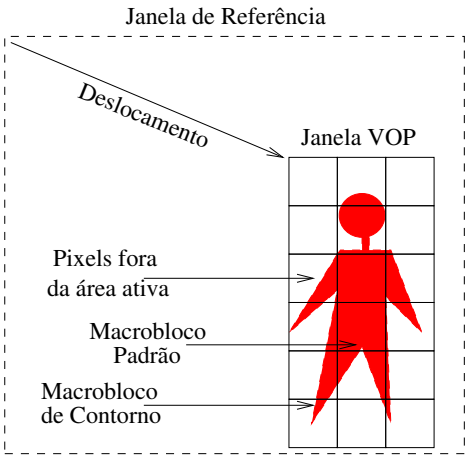


Figura 3.12: Codificação do VOP.

VOP. Antes da estimação e compensação de movimento do VOP atual em um quadro t, uma técnica simples de "estofamento" de imagem é usada para referenciar o VOP do quadro $t-1$. O método de "estofamento" consiste na extrapolação dos "*pixels*" de fora do VOP baseado nos pixels dentro do VOP. Depois da aplicação dessa técnica, a estimação e compensação de movimento são os mesmos usados para os macroblocos padrão. Contudo, durante o casamento dos blocos, somente os "*pixels*" que pertencem à área ativa do VOP são usados no processo de estimação do movimento.

Modos de Escalonabilidade

O padrão MPEG-4 permite acesso baseado em conteúdo em várias resoluções espaciais e temporais. Diferente do MPEG-2, que facilita os modos de escalonabilidade baseado em quadros, o MPEG-4 suporta modos de escalonabilidade baseado em objetos. Decodificadores incapazes de reconstruir os VOPs em resolução máxima podem decodificar subconjuntos de fluxos de bits para apresentar somente os VOPs selecionados.

A qualidade de apresentação dos objetos também pode variar, i. e., objetos mais importantes para a cena podem ser decodificados com uma qualidade melhor do que objetos menos importantes.

A Linguagem de Descrição de Sistemas MPEG-4

A linguagem de descrição de sistemas MPEG-4 ("MPEG-4 *system description language*" - MSDL) é uma linguagem que permite a integração de algumas ferramentas de vídeo. Essas ferramentas podem ser algoritmos, módulos de codificação de formas, módulo de compensação de movimento ou alguma outra técnica relacionada a essas.

Entre os componentes da MSDL estão as definições de interfaces entre as ferramentas de codificação, mecanismos para combinar essas diferentes ferramentas de codificação e construção de algoritmos, bem como mecanismos para acrescentar novas ferramentas.

A MSDL vai transmitir para o decodificador o fluxo de bits e a forma com a qual as ferramentas devem ser usadas para reconstruir o áudio e o vídeo. Em um nível mais avançado, a MSDL pode permitir a carga de ferramentas que não estão disponíveis no decodificador.

Os perfis suportados pela MSDL definem subconjuntos de ferramentas ou algoritmos específicos para cada aplicação, e. g., codificação com baixo atraso ou decodificação de baixa complexidade. Os níveis no MPEG-4 referem-se a uma especificação das restrições e critérios de desempenho necessários para satisfazer uma ou mais aplicações.

Como um decodificador com todas as facilidades do MPEG-4 seria impraticável, então se definem três níveis de programação que facilitam a flexibilidade e extensões.

O nível Flex_0 contém um conjunto padronizado de ferramentas de áudio e vídeo.

O nível Flex_1 possui um decodificador com um conjunto finito de ferramentas de áudio e vídeo e suas interfaces padronizadas que podem ser flexivelmente configuradas em algoritmos arbitrários. Os decodificadores flexíveis usam ferramentas de descompressão otimizadas localmente, mas podem descarregar novas classes de dados (objetos de áudio e vídeo).

O nível Flex_2 é o decodificador expansível que suporta um mecanismo padronizado que descreve algoritmos arbitrários feitos de ferramentas arbitrárias.

3.3.3 MPEG-7

Enquanto os padrões MPEG-1, 2 e 4 focalizam a representação codificada de conteúdo áudiovisual, o padrão MPEG-7 concentra-se na informação sobre o conteúdo. A idéia é prover um conjunto de ferramentas padronizadas para descrever um conteúdo multimídia [29, 7].

O padrão MPEG-7 é formalmente chamado de interface de descrição de conteúdo multimídia. O objetivo do padrão é prover uma solução simples, flexível e interoperável para problemas de indexação, busca e recuperação de recursos multimídia.

A base do padrão são os descritores (D). Um descritor define a sintaxe e a semântica da representação de uma característica particular em um conteúdo audiovisual. Por exemplo, a cor de uma imagem é uma característica. Os descritores para a característica cor são o histograma[13], o vetor RGB ou uma cadeia de caracteres que a representa. A Tabela 3.1 mostra alguns descritores usados no padrão MPEG-7.

Tipo	**Característica**	**Descritor**
Vídeo	Estruturas básicas	Layout
		Histograma
	Cor	Espaço de cores
		Cor dominante
		Histograma de cores
		Quantização de cores
	Textura	Distribuição espacial de intensidade de imagem
		Textura homogênea
	Forma	Caixa de contorno de objetos
		Forma baseada em região
		Forma baseada em contorno
		Descritor de forma 3-D
	Movimento	Movimento de câmera
		Trajetória de movimento do objeto
		Movimento parametrizado do objeto
		Velocidade, direção e aceleração
Áudio	Anotações de fala	Palavras e fonemas
	Timbre	Relação entre harmônicos pares e ímpares
		Coerência harmônica
	Melodia	Ritmo

Tabela 3.1: Descritores no padrão MPEG-7 [29]

Cada descritor é definido por partes normativas e não-normativas. As primeiras partes são compostas pela sintaxe do descritor, sua semântica e suas representações binárias. As últimas, opcionais, podem ser métodos de extração recomendados e métodos de buscas de similaridades.

A padronização envolve os seguintes itens:

[13] O histograma indica a distribuição dos "*pixels*" para cada nível de intensidade de cor.

- um conjunto de descritores que serão usados para descrever várias características multimídia;
- uma estrutura pré-definida de descritores e seus relacionamentos chamada de esquemas de descrição;
- uma linguagem para definir esquemas de descritores e esquemas de descrição, chamada de linguagem de definição de descrições (DDL[14]);
- representações de descrições codificadas para habilitar acesso rápido e armazenamento eficiente.

O esquema de descrição (ED) especifica a estrutura e a semântica das relações entre os componentes que podem ser descritores ou outros esquemas de descrição.

Os conceitos usados para descrever conteúdo audiovisual em um grupo de esquemas de descrição são:

- Estrutura sintática: estrutura física e lógica do conteúdo audiovisual.
- Estrutura semântica: estrutura baseada no significado do conteúdo audiovisual.
- Ligação sintático-semântica: associação entre elementos sintáticos e semânticos.

O ED audiovisual genérico, especificado em [37], representa a integração de uma série de esquemas de descrição. Tal esquema consiste dos seguintes elementos:

- Uma coleção de esquemas de descrição de estruturas sintáticas. Por exemplo, características físicas como regiões, cores e texturas, entre outras.
- Uma coleção de esquemas de descrição de estruturas semânticas, e. g., "milésimo gol de Pelé" ou "Propaganda da Sereia".
- Referências estático-semânticas que relacionam elementos sintáticos e semânticos.
- Sumários de Esquemas de Descrição usados para navegação em diferentes níveis de granularidade.

[14]Usa-se a sigla DDL para identificar "*Description Definition Language*"

- Esquemas de Descrição MetaInfo usados para descrever informações geradas pelo autor ou editor.
- Esquemas de Descrição MídiaInfo que contêm descritores relacionados ao meio de armazenamento, e. g., formato de arquivo, sistema, duração e formato de compressão, entre outros.
- Modelo de ED que provê meios de descrever os métodos de classificação para dados audiovisuais ou correspondências entre o conteúdo audivisual corrente e conteúdos em modelos diferentes.

Linguagem de Definição de Descrições

Para especificar e modificar os esquemas de descrição é necessária a utilização de uma linguagem que expresse as relações espaciais, temporais, estruturais e conceituais entre os elementos de um ED. Tal linguagem deve também prover um modelo que suporte ligações e referências entre uma ou mais descrições e os dados que elas descrevem.

A soma das características que essa linguagem deve possuir influenciou o grupo que trabalha no MPEG-7 a baseá-la na sintaxe da XML. Portanto, a Linguagem de Definição de Descrições (DDL) se baseia no padrão XML Schema [51, 52].

3.3.4 MPEG-21

O padrão MPEG-21 tem como objetivo a definição de uma estrutura que permita a utilização transparente de recursos multimídia através de uma grande variedade de redes e dispositivos utilizados por diferentes comunidades [5]. Em suma, o foco desse padrão é interoperabilidade.

O MPEG-21 baseia-se no conceito de item digital, usado pelo próprio padrão como sinônimo de conteúdo multimídia, para conseguir tal interoperabilidade. O item digital é o objeto que permite a interação entre os usuários. Essa interação ocorre sobre a estrutura proposta pelo MPEG-21.

Para que a estrutura forneça os meios de interação aos usuários, foram definidos no MPEG-21 sete elementos principais descritos a seguir.

1. Declaração de Item Digital: esquema interoperável de abstração flexível e uniforme para declarar itens digitais.

2. Identificação e Descrição de Item Digital: estrutura para identificação e descrição de qualquer entidade independente da sua natureza, tipo ou granularidade (tamanho e tempo para processar tal item).

3. Utilização e Manipulação de Conteúdo: interfaces e protocolos que permitem a criação, manipulação, busca, acesso, armazenamento, uso e reuso do conteúdo através de uma cadeia de distribuição e consumo de conteúdo.

4. Proteção e Gerenciamento de Propriedade Intelectual: meios para habilitar a persistência e gerenciamento de conteúdo em uma variedade de dispositivos e redes.

5. Redes e Terminais: acesso transparente e interoperável através de diferentes redes e terminais.

6. Representação de conteúdo: representação dos recursos multimídia.

7. Relatório de eventos: métricas e interfaces que habilitam os usuários a entender precisamente os eventos que foram relatados.

O modelo de Declaração de Item Digital (DID) possui os elementos a seguir, entre outros.

1. "*Container*": estrutura que permite o agrupamento de itens ou outros "*containers*". Esses agrupamentos podem formar pacotes lógicos.

2. Item: agrupamento de subconjuntos ou componentes que são ligados a descritores relevantes. Os descritores contém informações sobre o item, como uma representação de um trabalho. Os itens contêm opções que permitem sua configuração ou personalização. A relação entre itens e itens digitais é que os primeiros são representações declarativas dos últimos.

3. Componente: ligação entre um recurso e todos os seus descritores relevantes. Esses descritores são informações relacionadas a toda ou parte da instância do recurso.

4. Âncora: corresponde à localização específica de um recurso e faz a ligação entre descritores e fragmentos.

5. Descritor: associa informação com um elemento. Essa informação pode ser um componente ou uma sentença textual.

6. Recurso: objeto (áudio, vídeo, imagem, texto) individualmente identificável.

7. Fragmento: designa um ponto específico ou intervalo em um recurso.

A Figura 3.13 ilustra a relação entre alguns desses elementos.

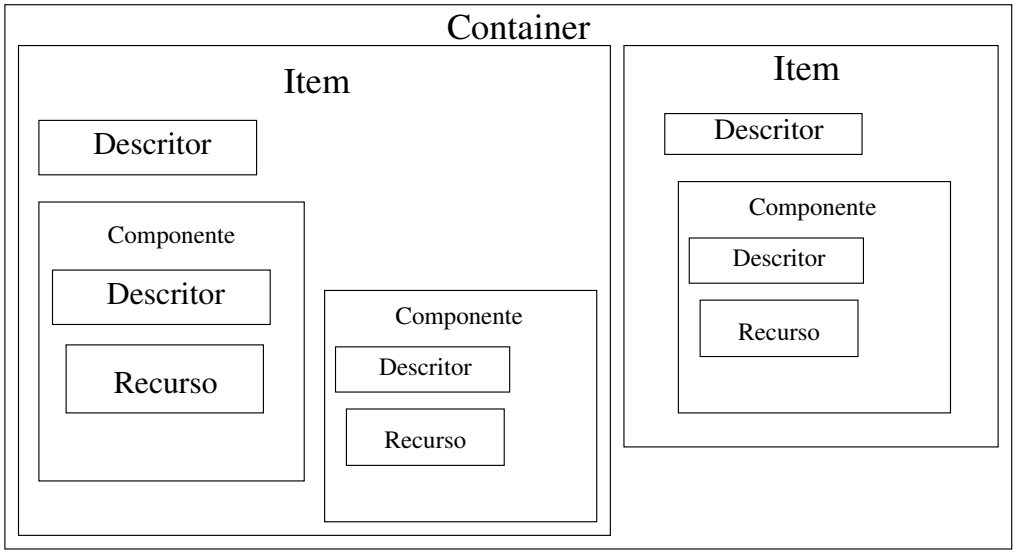


Figura 3.13: Relação entre alguns elementos do DID.

Por sua vez, a especificação da Identificação de Item Digital (DII) inclui como identificar unicamente os seguintes itens.

- Os itens digitais e suas partes (incluindo os recursos).
- O endereço IP relacionado com os itens digitais.
- Os esquemas de descrição.

Além disso, especifica como usar identificadores para ligar os itens digitais às informações relacionadas, tais como metadados descritivos.

3.4 Sumário

No decorrer deste capítulo procurou-se mostrar como a família de padrões MPEG evoluiu desde sua concepção, com o MPEG-1, até seu atual estágio, o MPEG-21.

No início do capítulo foram apresentadas as bases necessárias para se construir os primeiros padrões. Os algoritmos descritos na Seção 3.1 são essenciais para se compreender como se atinge a compressão requerida na família MPEG. A compressão sem perdas mantém toda a informação original preservada, enquanto que a compressão com perdas descarta informações que não são perceptíveis pelos seres humanos.

Na Seção 3.2 procurou-se descrever em detalhes os processos de codificação e decodificação de vídeo digital no padrão MPEG-1. Observe-se que esse processo se mantém em todos os padrões da família. Afinal, um dos pré-requisitos dos padrões é manter a compatibilidade com os demais.

Finalmente na Seção 3.3 mostraram-se as características particulares de cada padrão, ou seja, o que cada padrão novo acrescenta ao seu antecessor. A Figura 3.14 resume a relação entre os padrões.

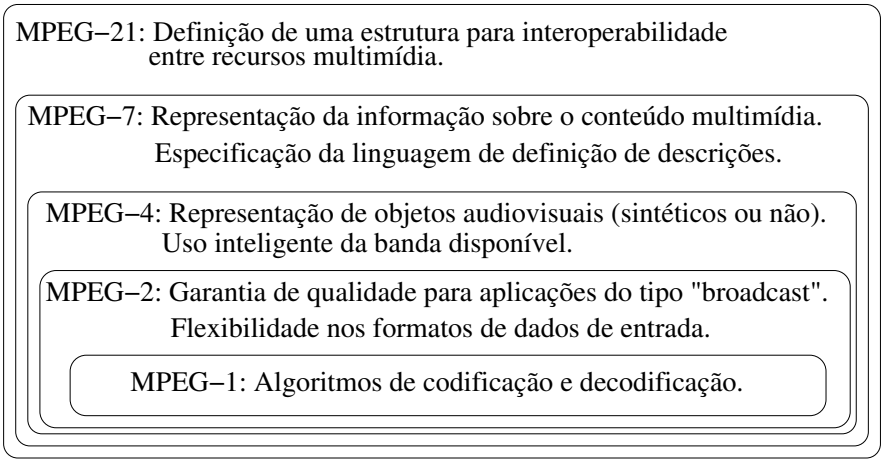


Figura 3.14: Relação entre os padrões MPEG.

CAPÍTULO 4

Hipervídeo

"*In theory, there is no difference between theory and practice.*
In practice, there is."
Yogi Berra.

Neste capítulo, são tratados alguns tópicos relativos à idéia de hipervídeo. Primeiro, se esclarece a relação entre os conceitos de hipertexto, hipermídia e outros para construir uma versão particular do conceito de hipervídeo. Em seguida, discute-se a idéia central desta tese: uma proposta de implementação do conceito de hipervídeo. Para isso, expõem-se as dificuldades da implementação da proposta e a definição da estrutura das informações hipermídia a serem embutidas. Finalmente, são apresentadas idéias gerais dos processos de codificação, decodificação e exibição de um hipervídeo.

4.1 Origens do Hipervídeo

"*Hypér*" é uma palavra de origem grega que significa "além", entre outros significados com a mesma conotação. Em um artigo de 1945, Vannevar Bush propôs o dispositivo Memex ("*Memory Extension*") [6]. Tal dispositivo foi concebido para guardar conhecimentos. Porém não de forma

linear, é o caso dos livros, e sim associativa como a mente humana. A Figura 4.1 ilustra ambas as formas.

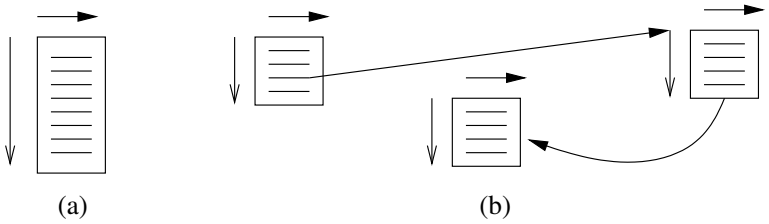


Figura 4.1: Formas de aquisição de conhecimentos: (a) linear, da esquerda para a direita, de cima para baixo; (b) não linear, da esquerda para a direita, de cima para baixo, mas visitando vários conteúdos diferentes.

Anos depois, em um artigo publicado em 1965 [39], Theodor Holm Nelson propôs o termo "*hypertext*". De acordo com Nelson, hipertexto significa um conjunto de textos e imagens interconectado de maneira tão complexa que não seria conveniente representar em papel. Além disso, o hipertexto conteria resumos e mapas do seu conteúdo e das inter-relações entre os tópicos. A palavra "hiper", então, é usada no sentido de "além do que está escrito de maneira linear". Deve-se considerar também a relação (referências) entre os tópicos.

Em 1968, Douglas Engelbart demonstrou o que seria a primeira implementação do que viria a ser hipertexto; o sistema chamava-se "*oN-Line System*" (NLS). Além de um sistema hipertexto, o NLS trouxe consigo algumas invenções muito úteis na computação atual e no campo dos hipertextos em particular. Entre tais invenções estão os sistemas de ajuda on-line, o ambiente de janelas e o, quase indispensável, "*mouse*" [31].

Em 1989, Tim Berners-Lee propôs um projeto de hipertexto global [3], aproveitando a infra-estrutura da Internet - que ainda estava para romper as barreiras das universidades e centros de pesquisa. Esse projeto deu origem ao que se conhece hoje por "*World Wide Web*" (WWW).

Com o potencial de comunicação que a WWW possui, a evolução do conceito de hipertexto para hipermídia foi quase imediata. Ao invés de se conectar idéias expressas em imagens estáticas e textos, por que não aumentar as possibilidades conectando também imagens dinâmicas (vídeo), sons e quaisquer outros meios de comunicação (mídias)?

Convém ressaltar a diferença entre multimídia e hipermídia. O primeiro conceito trata da utili-

zação de diversas mídias (meios de comunicação) em formato digital, sincronizadas entre si, para apresentação de conteúdo de forma linear [19].

O segundo conceito é uma extensão do primeiro. Além de ter todas as características de multimídia, hipermídia permite a apresentação de conteúdo de forma não-linear e dependente do interesse do usuário ou, mais formalmente, do mapa de conhecimentos que o usuário desenha ao ter contato com as informações apresentadas.

O conceito de hipervídeo, por sua vez, é uma particularização do conceito de hipermídia[44]. Tanto em hipertexto, como em hipermídia, o ponto de partida para navegação entre os conteúdos é o texto. No hipervídeo, como o próprio conceito sugere, o ponto de partida é o vídeo.

Portanto, há mais desafios para se implementar hipervídeo do que havia na implementação de hipertextos. Entre tais desafios estão a natureza fortemente linear dos vídeos (quadro a quadro), as relações espaço-temporal entre cenas e onde embutir as referências a outros conteúdos.

Salienta-se a necessidade de se concentrar no conceito de "contexto em hipervídeo", o diferencial deste trabalho. O contexto em hipervídeo é a relação entre um determinado objeto[1] mostrado na cena e as informações acerca desse objeto, não necessariamente apresentadas na cena. É importante vincular alguns objetos aos seus respectivos contextos, assim como nos hipertextos, onde as palavras-chaves atuam como referências a outros conteúdos complementares.

4.2 Inclusão de Informações Hipermídia em Vídeos

No âmbito deste trabalho consideram-se informações hipermídia como referências a outros conteúdos, implementados em outros meios de comunicação (inclusive vídeo), associadas a certos objetos presentes na mídia de origem.

Nos hipertextos, as informações hipermídia são interpretadas pelo navegador ("*browser*") que as associa a determinadas palavras, frases, figuras ou outros objetos visuais criando assim os "*hyperlinks*". Desse modo, quando o usuário clica sobre um "*hyperlink*", o navegador visita um determinado endereço e carrega uma outra página com outro conteúdo. Em vídeos tais procedimentos não são tão simples assim.

[1] Tais objetos são integrantes da imagem mostrada, podendo ser personagens, figuras ou quaisquer elementos pertencentes à cena apresentada.

Como argumento inicial ressalta-se que, na verdade, o vídeo é uma seqüência de imagens apresentada quadro a quadro a uma taxa de 24 quadros por segundo, no mínimo. Isso o torna uma mídia muito mais dinâmica do que o texto. O segundo argumento vem do fato de que, em vídeos, os objetos aos quais se associam os "*hyperlinks*" geralmente estão em posições diferentes em cada quadro, variando-as em função do tempo.

Diante dos argumentos citados, foi importante decidir onde as informações hipermídia devem ser inseridas, tendo-se então considerado três possibilidades. São elas:

i. criar um quadro extra, chamado quadro-H, que contivesse um mapa de informações hipermídia;

ii. inserir referências no cabeçalho da camada de macrobloco;

iii. inserir informações hipermídia no cabeçalho da camada de imagem.

A possibilidade (i) foi descartada porque isso levaria a um aumento do número de quadros. Apesar dos quadros-H serem passíveis de grande compressão (por ser basicamente texto), poucas informações hipermídia seriam aproveitadas, acarretando em desperdício de poder computacional.

A possibilidade (ii) também foi desconsiderada por sobrecarregar a decodificação dos macroblocos. Tal decodificação deve ser a mais eficiente possível, pois é grande a quantidade de macroblocos a tratar.

Por outro lado, a possibilidade (iii) se apresentou como a mais viável dentre as três consideradas. As justificativas são:

- no cabeçalho da camada de imagem há espaço não utilizado, onde informações hipermídia podem ser inseridas;
- pode-se mapear (especificar coordenadas) áreas diferentes em cada quadro, associando-se referências diferentes a cada área;
- com a associatividade entre áreas e referências, pode-se obter uma melhor sincronização entre contextos e objetos mostrados na imagem.

Portanto, analisando-se os argumentos, decidiu-se que a melhor alternativa seria incluir as informações hipermídia no cabeçalho da camada de imagem. O passo seguinte é definir, na prática, como serão tais informações.

4.2.1 Definição da Estrutura das Informações Hipermídia

Observando a maneira como os "*hyperlinks*" são definidos nos hipertextos atuais - em particular aqueles implementados em páginas HTML - percebe-se que o formato geral é:

```
<âncora referência> objeto </âncora>
```

O que está entre os símbolos "<"e ">"é chamado "*tag*". Portanto, para sinalizar o início de um "*hyperlink*" existe uma "*tag*" com uma `âncora` e a `referência` (endereço) para onde o "*hyperlink*" conduzirá. Marcando o final do "*hyperlink*" existe uma "*tag*" informando o final da âncora (`/âncora`). O `objeto`, que pode ser uma palavra, frase ou figura será assinalado (sublinhado ou com uma cor diferente) como a palavra chave que conduzirá o usuário a outro conteúdo.

No caso de um vídeo, não se pode garantir que o objeto ao qual se quer associar uma referência permaneça na mesma posição durante toda a exibição. Sendo assim, ao invés de se associar "*hyperlinks*" ao objeto, associam-se os "*hyperlinks*" às coordenadas do objeto em cada quadro. Dessa forma, ao clicar em uma determinada região em um quadro, é possível determinar se as coordenadas do ponto onde ocorreu o clique pertencem ou não à parte interna do contorno do objeto.

Obviamente, dependendo do contorno do objeto, a quantidade de coordenadas seria grande para mapeá-lo. Assim, para facilitar a verificação se o clique foi dentro ou fora da região de contorno do objeto, optou-se por definir um retângulo onde o objeto estaria inserido. Para determinar um retângulo, bastam apenas as coordenadas de dois pontos.

Logo, resolveu-se definir informações hipermídia como uma tripla contendo:

- coordenadas de início do retângulo;
- endereço de um conteúdo na WWW;
- coordenadas de fim do retângulo.

Nas seções 4.3 e 4.4 a seguir, são descritos os passos para codificação, decodificação e exibição desse tipo de hipervídeo.

4.3 Processo de Codificação

A arquitetura proposta para a parte do sistema que codifica a seqüência de quadros para formar um hipervídeo é ilustrada na Figura 4.2 a seguir[2].

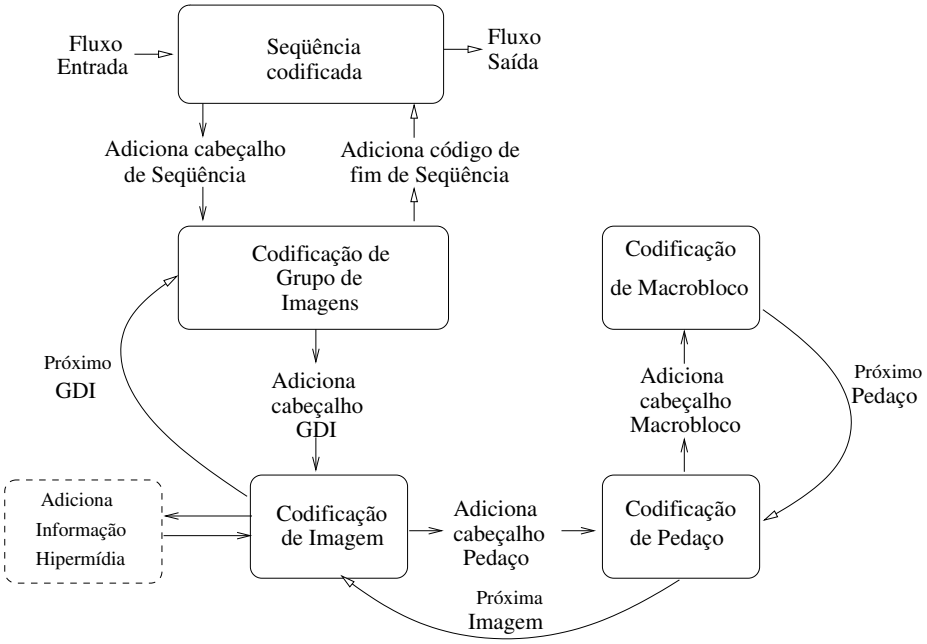


Figura 4.2: Arquitetura para o programa de codificação de um hipervídeo.

No início, há um fluxo de entrada de bits com informações a respeito de cada "*pixel*" de cada quadro. Antes da codificação, um cabeçalho de seqüência é incluído.

A partir de então, os grupos de imagens são codificados. A cada GDI são incluídos os cabeçalhos dos respectivos GDIs e uma série de imagens são codificadas. Para cada imagem, são codificados seus pedaços e são incluídos seus cabeçalhos. Para cada pedaço os macroblocos são codificados e seus cabeçalhos incluídos.

O diferencial desse processo em relação à codificação MPEG-1 está na codificação da imagem. Ao gerar o cabeçalho na imagem, todas as informações hipermídia relativas ao quadro sendo codificado são incluídas no cabeçalho. Isto significa que para cada objeto que possui uma referência associada, são incluídas suas coordenadas e seu respectivo "*hyperlink*".

[2]Detalhes a respeito das informações incluídas nos cabeçalhos estão na Seção 3.2.3, página 23.

4.4 Processo de Decodificação e Exibição

O processo de decodificação envolve também a exibição do hipervídeo. Em função da aglutinação dessas tarefas, esse processo torna-se um pouco mais complexo do que a codificação, conforme ilustra a Figura 4.3.

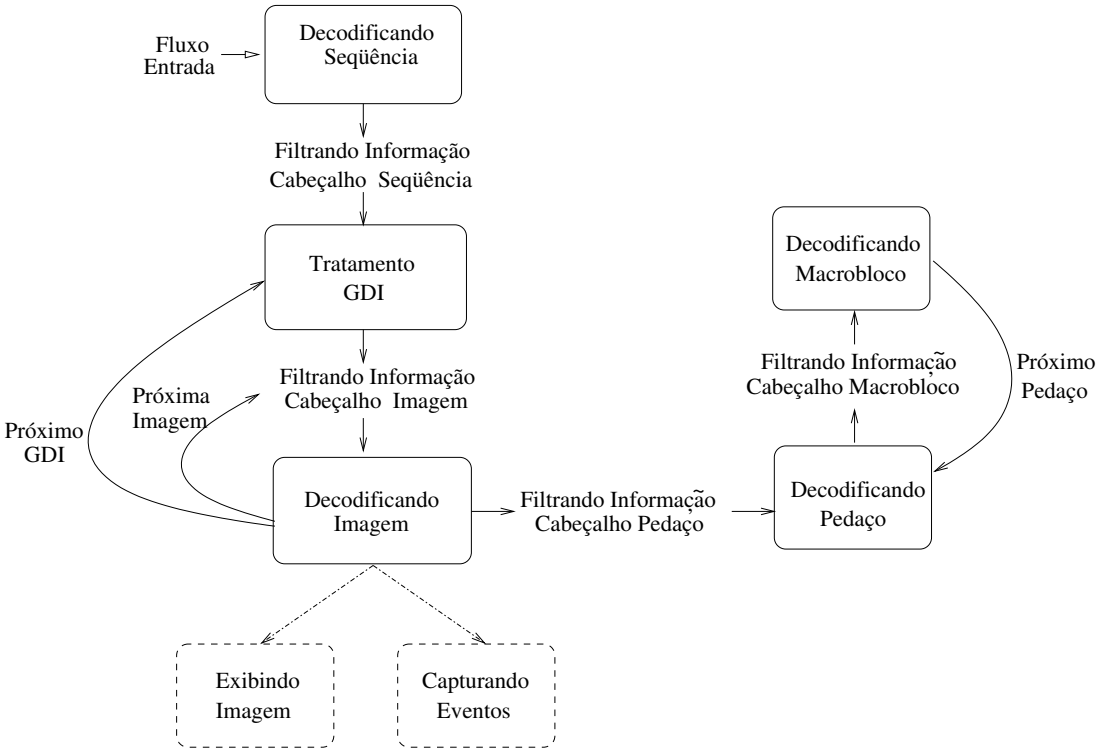


Figura 4.3: Arquitetura para o programa de decodificação de um hipervídeo.

O fluxo de bits de entrada está codificado de acordo com o padrão MPEG-1. Assim, uma seqüência de quadros é decodificada. O primeiro passo para a decodificação é filtrar as informações que estão no cabeçalho de seqüência. A partir de então vários grupos de imagens são obtidas. Para cada um desses grupos, o respectivo cabeçalho é filtrado e as informações do grupo específico são obtidas.

Cada imagem pertencente ao grupo tem seu respectivo cabeçalho filtrado e a imagem em si é decodificada em pedaços. Cada pedaço que forma a imagem também possui seu respectivo cabeçalho com informações próprias, bem como os macroblocos. Depois que todas as informações pertinentes à imagem são recolhidas, então se prossegue com a exibição.

O que foi descrito nos dois últimos parágrafos está relacionado ao processo de decodificação comum a um vídeo implementado no padrão MPEG. Concomitante a esse está o processo de exibição que se resume em alterar cada "*pixel*" que forma a imagem.

No caso da proposta de hipervídeo implementada neste trabalho, há ainda um terceiro processo ocorrendo "simultaneamente". Trata-se da captura de eventos. Esse terceiro processo é importante para acompanhar a interatividade do usuário com o vídeo. O objetivo, portanto, é capturar cada um dos eventos instanciados pelo usuário quando se relaciona com as imagens apresentadas[3].

O processo de captura de eventos envolve também o tratamento desses eventos. Existem, basicamente, dois comportamentos implementados nesse processo. Um deles é desconsiderar o evento gerado pelo usuário quando o ponto da imagem clicado não está associado a nenhum "*hyperlink*". O outro comportamento inclui disparar o processo que trará informações complementares sobre o objeto apresentado dentro da área delimitada pelas informações hipermídia, onde ocorreu o clique do usuário.

4.5 Sumário

Neste capítulo foram apresentados o conceito de hipervídeo e os desafios para implementá-lo. Ressalta-se que esse conceito ainda não é tão sólido quanto os demais com o prefixo "*hiper*". Portanto, a implementação que se propõe neste texto ainda não é a definitiva, mas uma prova de conceito.

No que se refere à implementação da prova de conceito, este capítulo descreveu a estratégia escolhida para a inclusão de informações hipermídia em um vídeo, bem como a estrutura dessas informações. Foram ainda mostrados os processos de codificação e decodificação de hipervídeos de acordo com o conceito proposto.

Em suma, este capítulo apresentou a principal contribuição da pesquisa desenvolvida nesta tese: o conceito de hipervídeo. Tal conceito é a base para a TV interativa proposta neste trabalho e traz algumas inovações em relação a outras propostas já discutidas no Capítulo 2. O principal diferencial é a vinculação das informações hipermídia aos objetos em cena, criando o chamado "contexto em hipervídeo". Isso permite uma maior interatividade do telespectador com o conteúdo assistido.

[3] O relacionamento do usuário com as imagens se dá a partir de cliques com o "*mouse*" em alguns pontos da imagem apresentada.

CAPÍTULO 5

Implementação do Protótipo

"*Don't be too proud of this technological terror you have constructed.*
The ability to destroy a planet is insignificant next to the power of the Force."
Darth Vader em "*StarWars IV: A New Hope*".

Neste capítulo é descrita a implementação do conceito de hipervídeo proposto. O sistema implementado divide-se em duas partes: a primeira é relativa à codificação, enquanto a segunda refere-se à decodificação de um hipervídeo.

Ambos os procedimentos foram ilustrados no capítulo anterior e, neste, detalhes relativos aos códigos dos programas serão apresentados. Ao final deste capítulo são exibidas algumas telas que mostram a execução do protótipo. Para iniciar, apresenta-se uma breve discussão a respeito das ferramentas utilizadas.

5.1 Software Livre

Na época atual, há uma grande discussão a respeito de software. Entre os temas envolvidos nessa discussão, um dos que ganha destaque é a dúvida entre a utilização de software livre e proprietário.

De acordo com a GNU [18], software livre é aquele em que o acesso, execução, melhoramento e redistribuição do programa são permitidos, sem a necessidade de solicitações de licença. Entretanto, para que tal liberdade seja mantida, é necessário que os autores sejam citados. Por outro lado, há o software proprietário onde o acesso, melhoramento e redistribuição do programa não são permitidos a não ser por autorização expressa do detentor (pessoa física ou jurídica) da licença.

A discussão a respeito de qual dos dois tipos de software é mais recomendado não pertence ao escopo deste trabalho. Entretanto, a preferência por software livre para a implementação do protótipo é justificada por questões econômicas[1] e acadêmicas, uma vez que o conteúdo produzido por este trabalho deve estar amplamente disponível a qualquer pessoa.

5.1.1 Ferramentas Utilizadas

Algumas das ferramentas de software utilizadas para a produção do protótipo que implementa as idéias deste trabalho merecem destaque. São elas:

- Sistema operacional Linux, versão do "*kernel*" 2.4.20-8, distribuição Red Hat 9.
- Sistema de interface gráfica X Window [36, 56].
- Compilador GCC ("*Gnu Compiler Colection*", linguagem C) [17], versão 3.4.3.
- Sistema de Documentação Doxygen [47], versão 1.3.9.1.

Além dessas ferramentas, foram utilizados como base para a implementação do protótipo os códigos-fonte de um codificador, um decodificador e um visualizador, disponíveis na WWW a partir da página do Berkeley Multimedia Resource Center [2] gratuitamente. Esses softwares foram desenvolvidos pelo "*MPEG Software Simulation Group*" [38].

Nas seções a seguir, são descritos os trechos principais dos programas que implementam a codificação e decodificação de um hipervídeo, bem como as estruturas de dados utilizadas.

[1] Lembrando que software livre não significa necessariamente software grátis. Contudo, a maioria dos softwares utilizados estava disponível gratuitamente.

5.2 Estruturas de Dados

Na decodificação dos hipervídeos são necessárias algumas estruturas de dados para, por exemplo, armazenar as informações hipermídia pertencentes àquele quadro. Para esse caso, em particular, foi criada uma lista ligada dinâmica[2] de informações hipermídia. Essa lista possui a estrutura descrita na Figura 5.1.

```
typedef struct
{
 unsigned int x;
 unsigned int y;
} coordenadas;

typedef struct
{
 coordenadas inicial;
 char *link;
 coordenadas final;
} HypervideoData;

typedef struct Lista
{
  HypervideoData *hvd;
  struct Lista *prox;
 } ListaHVD;
```

Figura 5.1: Estrutura de dados `ListaHVD`.

Sendo assim, cada nó da lista `ListaHVD` possui uma referência para a estrutura `HypervideoData` e para o próximo nó da lista, a ser criado quando necessário. Por sua vez, a estrutura `HypervideoData` possui as coordenadas inicial e final do retângulo, que contém o objeto a ser referenciado e uma referência ao endereço vinculado. Finalmente, as coordenadas são representadas pela estrutura `coordenadas`, que contém dois inteiros.

[2]Uma lista ligada é uma estrutura de dados composta por zero ou mais nós. Cada nó é um registro contendo um conjunto de informações qualquer. Por ser dinâmica, essa lista pode aumentar ou diminuir em número de nós ocupando mais ou menos espaço de memória, à medida que for necessário.

Há ainda, no programa visualizador, uma estrutura importante que armazena diversas informações a respeito de cada imagem que compõe o hipervídeo, inclusive uma lista de informações hipermídia pertencentes apenas àquela imagem. A Figura 5.2 mostra a definição da estrutura `Pict`.

```
typedef struct pict {
 unsigned int temp_ref;           // Referencia temporal.
 unsigned int code_type;          // Tipo de quadro.
 unsigned int vbv_delay;          // Atraso do buffer
 BOOLEAN full_pel_forw_vector;    // Usado para decod. de vetores
 unsigned int forw_r_size;        // Usado para decod. de vetores
 unsigned int forw_f;             // Usado para decod. de vetores
 BOOLEAN full_pel_back_vector;    // Usado para decod. de vetores
 unsigned int back_r_size;        // Usado para decod. de vetores
 unsigned int back_f;             // Usado para decod. de vetores
 char *extra_info;                // Bit para informacao extra
 char *ext_data;                  // Dados de extensao
 char *user_data;                 // Dados de usuario
 ListaHVD *ListaHypervideoData; // Informacoes hipermidia
} Pict;
```

Figura 5.2: Estrutura de dados `Pict`.

5.3 Implementação do Codificador

Antes de entrar em detalhes a respeito da implementação do codificador, convém descrever o que o programa recebe como dados de entrada e produz como saída.

5.3.1 Dados de Entrada e Saída

Esse procedimento de codificação de um hipervídeo recebe como entrada um arquivo de parâmetros, um arquivo contendo os "*hyperlinks*". O arquivo de parâmetros, em texto puro, contém um conjunto de informações que permite a codificação de arquivos com os hipervídeos. Entre tais informações estão:

- nomes dos arquivos que contêm informações sobre luminância e crominância de cada quadro, nos formato YUV;
- número de quadros total do filme;
- número de quadros por grupo de imagens;
- informação sobre "*aspect ratio*";
- outras informações para codificação no formato MPEG.

Quanto ao arquivo de "*hyperlinks*", ele contém um conjunto de informações hipermídia, que serão inseridas nos quadros que compõem o hipervídeo. Na saída de dados, o programa produz um hipervídeo pronto para exibição pelo programa visualizador.

5.3.2 "*Modus Operandi*" do Codificador

Na fase inicial, o programa codificador abre o arquivo de parâmetros e obtém todos os dados necessários para a codificação, inclusive a matriz de quantização. Em seguida, o arquivo de saída, que conterá o hipervídeo, é preparado para a gravação e, ao final dessa fase, o arquivo de "*hyperlinks*" também é aberto.

A segunda fase contempla a gravação de cabeçalhos e dados, de acordo com o padrão MPEG. Dentro do conjunto de funções que desempenham essa tarefa, ressalta-se a função `putpict`, responsável pela gravação de cada quadro no arquivo que contém um hipervídeo. Um trecho da função `putpict` está descrito na Figura 5.3.

A função `putpict` recebe como parâmetro um conjunto de bits que compõem o quadro que será gravado no arquivo. Nas primeiras linhas dessa função, é definido o controle de taxa de quadros por segundo. Em seguida, é gravado o cabeçalho da imagem corrente, através da função `putpicthdr`. Depois, o arquivo de "*hyperlinks*" é aberto e, se seu conteúdo não for vazio, as linhas desse arquivo são lidas e colocadas em cada um dos quadros. Nesse instante, a função `putpicthdrext`, apresentada na Figura 5.4, é invocada para gravar as informações hipermídia no quadro corrente.

A função `putpicthdrext` é bastante simples e recebe como parâmetros os dados a serem gravados no quadro corrente. No início os bits são alinhados, i. e., os espaços no cabeçalho são

```
void putpict(unsigned char *frame)
{
 ...
 char *res;
 ...
 rc_init_pict(frame);        // Definicao do controle de taxa.
 putpicthdr();               // Grava cabecalho da imagem corr.
 res = le(arquivoHiperlinks);// Le o arquivo de Hiperlinks
 if (res != NULL)
    putpicthdrext(res);      // Grava informacoes hipermidia.
 else
    putpicthdrext("NULO ");
 ...
}
```

Figura 5.3: Trecho da função `putpict`.

```
void putpicthdrext(char *userdata)
{
  alignbits(); //alinha os bits
  putbits(ANDRE_START_CODE,32); /* Grava um codigo de 32 bits
                                 * que determina o INICIO das
                                 * informacoes hipermidia.*/
  while (*userdata)
      putbits(*userdata++,8);  /* Gravacao, bit a bit, das
                                * informacoes hipermidia
                                */

   putbits(ANDRE_END_CODE,32); /* Grava um codigo de 32 bits
                                * que determina o FINAL das
                                * informacoes hipermidia.*/
}
```

Figura 5.4: Função `putpicthdrext`.

preenchidos com zeros até a posição exata onde devem estar as informações hipermídia. Em seguida, é gravado um código de 32 bits que determina o início do campo reservado às informações hipermídia.

A partir de então, qualquer quantidade de informações pode ser armazenada no cabeçalho. Ao final, um código de 32 bits para determinar o final do campo reservado às informações hipermídia é gravado. O restante do programa funciona de acordo com o padrão original.

5.4 Implementação do Visualizador

O visualizador, por sua vez, é um programa um pouco mais complexo, pois lida com a decodificação e exibição do hipervídeo, bem como o tratamento de eventos, atuando como um sistema multitarefa [27, 35, 45].

5.4.1 Decodificação do Hipervídeo

Quanto à decodificação do fluxo de vídeo, o programa segue o algoritmo convencional até o ponto em que o cabeçalho de cada imagem é interpretado. Observe-se o trecho de código da função `mpegVidRsrc` na Figura 5.5, responsável pela decodificação do fluxo de vídeo e exibição quadro a quadro.

No trecho em questão, o início do cabeçalho da camada de imagem é tratado. Se o `status` de uma imagem for `SKIP_PICTURE`, indicando que o cabeçalho não é de uma imagem e sim de um grupo de imagens (`GOP_START_CODE`) ou fim de seqüência (`SEQ_END_CODE`), então essas informações são filtradas até que o procedimento se posicione no início da camada de imagem. Após essa sincronização, ou seja, quando chega o ponto de se interpretar o trecho do cabeçalho onde estão as informações hipermídia, a função `ParsePictureExtension` é chamada.

Observe-se, na Figura 5.6, um trecho da função `ParsePictureExtension`. No trecho mostrado, o código de início de informações hipermídia é descartado. Em seguida, as informações são obtidas através da função `get_hypermedia_data` e colocadas na lista de informações hipermídia apropriada. A partir de então, cada imagem possui uma lista ligada dinâmica de informações hipermídia. O restante da decodificação segue conforme o padrão.

```
VidStream *mpegVidRsrc(TimeStamp time_stamp,
                       VidStream *vid_stream,
                       int first, XInfo *xinfo)
{...
// Interpreta o inicio do cabecalho da imagem
status = ParsePicture(vid_stream, time_stamp);
if (status == SKIP_PICTURE)
{
  next_start_code(vid_stream);
  while (!next_bits(32, PICTURE_START_CODE, vid_stream))
  {
     if (next_bits(32, GOP_START_CODE, vid_stream))
         break;
     else if (next_bits(32, SEQ_END_CODE, vid_stream))
         break;
     flush_bits(24);
     next_start_code(vid_stream);
  }
  goto done;
} else if (status != PARSE_OK)
          goto error;

// Interpreta o trecho que contem as informacoes hipermidia
ParsePictureExtension(vid_stream);

// Continua com a interpretacao do cabecalho da camada de pedaco
if (ParseSlice(vid_stream) != PARSE_OK)
       goto error;
break;
...
}
```

Figura 5.5: Trecho da função `mpegVidRsrc`.

5.4.2 Tratamento de Eventos

Em relação à exibição do vídeo, convém lembrar que, por ser um protótipo baseado no sistema operacional Linux, a interface gráfica do programa visualizador foi implementada utilizando-se o sistema X Window. Esse sistema provê mecanismos para criar objetos gráficos (e.g. botões, janelas) e capturar eventos.

```
static int ParsePictureExtension(VidStream *vid_stream)
{
  char *string;
  ...
  // Descarta o codigo de inicio de info. hiperm.
  if (next_bits(32,ANDRE_START_CODE, vid_stream))
  {
    flush_bits32;
  }
  else return PARSE_OK;

  string = (char *) get_hypermedia_data(vid_stream);

  vid_stream->picture.ListaHypervideoData =
                          (ListaHVD *) filtraString(string);
  ...

  // Descarta o codigo de fim de info. hiperm.
  if (next_bits(32,ANDRE_END_CODE, vid_stream))
  {
    flush_bits32;
  }

  return PARSE_OK;
}
```

Figura 5.6: Trecho da função `ParsePictureExtension`.

No caso do protótipo para visualização dos hipervídeos, a função `ControlBar` é responsável pelo tratamento de eventos. O trecho de código da respectiva função é mostrado na Figura 5.7. Note-se que a função `XNextEvent`, que faz parte da biblioteca X Window, obtém o próximo evento ligado ao "*display*" (área do programa responsável pela exibição do hipervídeo). Todas as informações ligadas ao evento, inclusive as coordenadas (no caso de um clique do "*mouse*"), serão armazenadas na estrutura `event` que foi passada como parâmetro para a função `XNextEvent`.

Uma vez que as coordenadas do clique do "*mouse*" são conhecidas, o próximo passo é chamar a função `browser`. Além das coordenadas, o endereço da lista de informações hipermídia referentes àquela imagem também é transmitido como parâmetro. Cabe à função verificar se as coordenadas do

```
void ControlBar(VidStream **vid_stream,
                XInfo *xinfo,
                int numMovies)
{
  ...
  XNextEvent(display, &event);
  ...
       coord.x=(unsigned int)event.xbutton.x;
       coord.y=(unsigned int)event.xbutton.y;
  ...
       if (browser(&coord,
                  (**vid_stream).picture.ListaHypervideoData))
            printf("Coordenadas dentro\n");
       else
            printf("Coordenadas fora\n");
  ...
}
```

Figura 5.7: Trecho da função `ControlBar`.

clique estão embutidas em algum retângulo que é delimitado a partir das informações hipermídia e invocar o navegador, se esse for o caso. Já a função `browser` foi codificada na Figura 5.8.

No início, se a lista de informações hipermídia for vazia, a função retorna imediatamente com um código informando que as coordenadas do clique estão fora de qualquer objeto que possua um "*hyperlink*" associado. O mesmo acontece se, após procurar em todos os nós da lista, não for encontrado qualquer conjunto de coordenadas que contenha as coordenadas do clique do "*mouse*".

Caso nenhuma das opções do parágrafo anterior seja confirmada, um novo processo (chamado filho) é disparado através da primitiva `fork`[3]. O processo anterior (chamado pai) retorna imediatamente à função `ControlBar`, enquanto que o processo filho invoca o programa navegador, já passando como parâmetro o endereço do site (URL) onde estão as informações complementares.

[3] `fork` é uma primitiva do sistema operacional responsável pela criação de um processo filho com o mesmo código do processo pai, porém com identificadores de processo diferentes. Após a chamada, cada processo pode seguir caminhos de execução diferentes [27, 35, 45].

```
int browser(coordenadas *coordClick, ListaHVD *lista)
{
 int pid;
 ListaHVD *aux;
 if (lista == NULL) return 0;
 aux = lista;
 while (aux != NULL)
 if(COORDENADASINCLUSAS(aux->hvd->inicial.x,aux->hvd->inicial.y,
                        coordClick->x, coordClick->y,
                        aux->hvd->final.x, aux->hvd->final.y))
 {
   ...
   break;
 }
 else
    aux = aux->prox;

 if (aux == NULL) return 0;

 pid = fork();
  if (pid<0)  exit(1); // se ocorreu erro, aborte programa.

 if (pid!=0)
     return 1;// se for o processo pai, retorne imediatamente.

 // Se for o processo filho, continue desta linha em diante.
 retorno = disparaBrowser(aux->hvd->link /*URL*/ );
 if (!retorno)
  printf("OK");
 else
  exit(1);
}
```

Figura 5.8: Trecho da função `browser`.

5.5 Prova de Conceito

Uma prova de conceito é uma realização, geralmente incompleta, de uma idéia para mostrar que ela é factível, ou seja, é uma evidência que essa idéia é viável. Sendo assim, para demonstrar a

viabilidade da idéia proposta neste trabalho, são apresentadas algumas telas que ilustram a execução do protótipo[4].

A primeira tela, apresentada na Figura 5.9, mostra todos os arquivos que compõem um vídeo convencional no formato YUV, além do arquivo de links `arquivo.link`, e o arquivo de parâmetros de vídeo `Grom.par`.

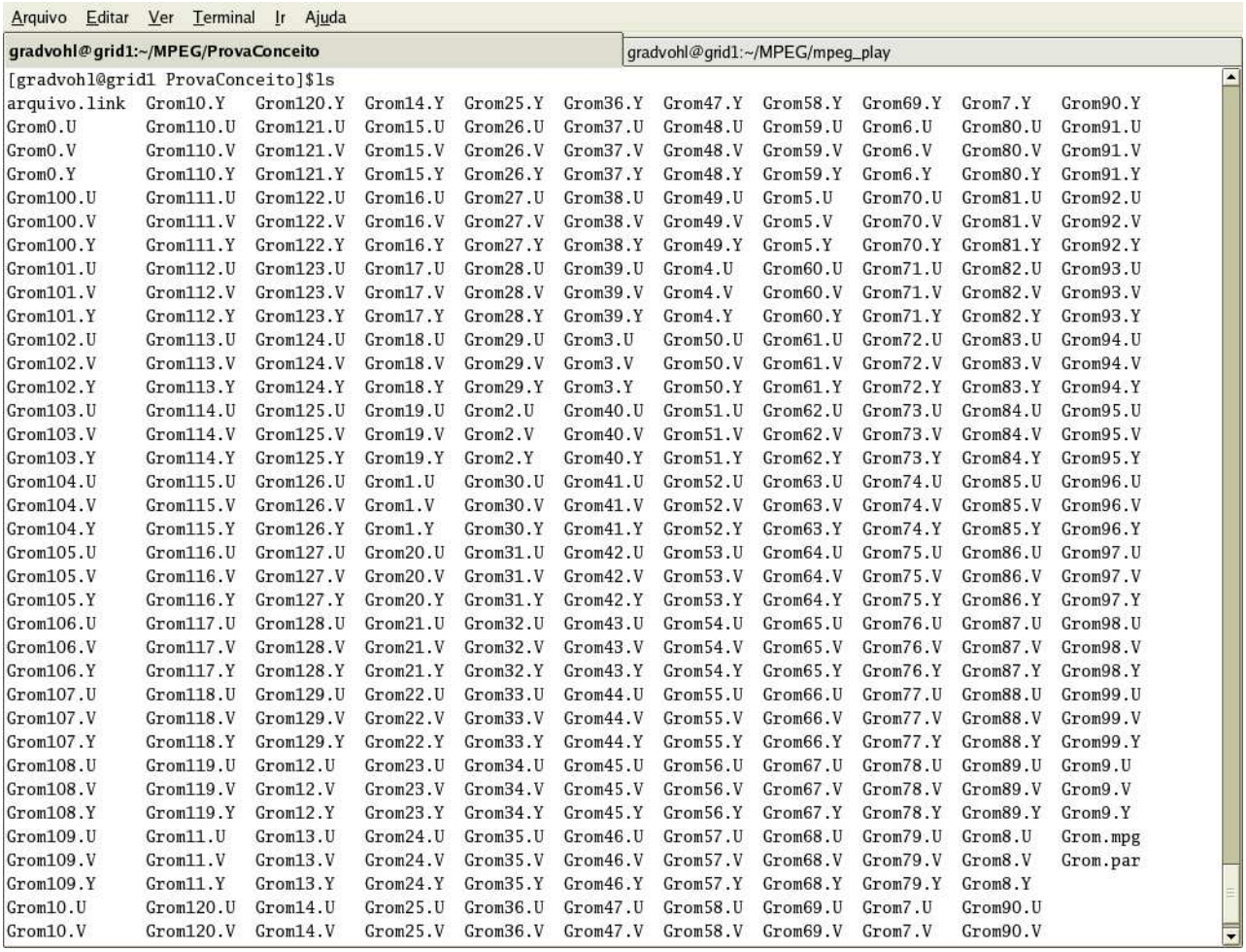


Figura 5.9: Quadros do vídeo exemplo, codificados no formato `YUV`.

A próxima tela, mostrada na Figura 5.10, ilustra o processo de codificação do hipervídeo. Vê-se primeiramente a linha de comando que instancia o processo. A seguir, são passados como informações para o programa de codificação um arquivo de parâmetros de vídeo, o nome do arquivo que conterá o hipervídeo e o arquivo de "*links*". Depois de cada quadro codificado, são também

[4]Essas telas foram capturadas com a ajuda do software Image Magick[10].

indicadas as informações hipermídia inseridas. No caso particular do exemplo da Figura 5.10 o "*link*" `http://home.in.tum.de/paula/mpeg/index.html` será incluído nas coordenadas `(100,10)` e `(250,190)`.

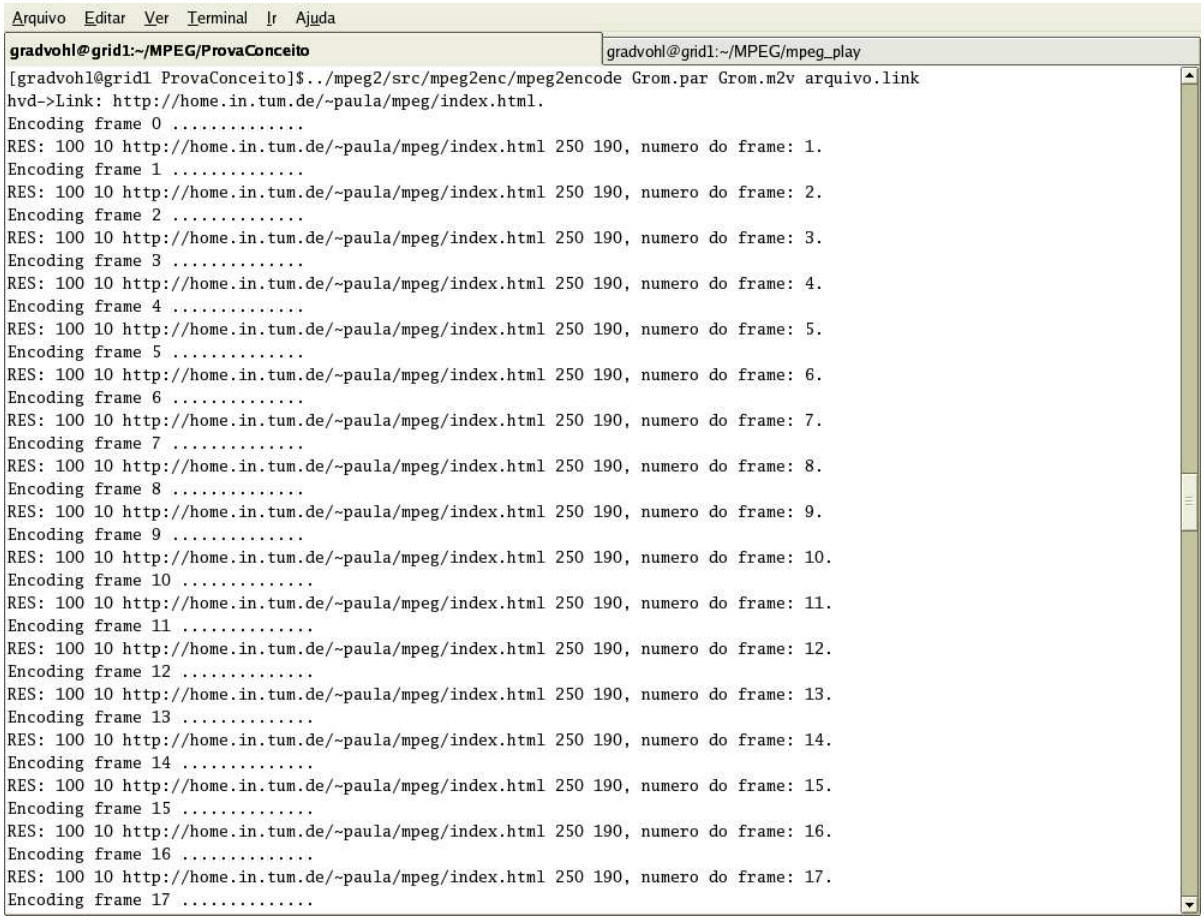


Figura 5.10: Codificação do hipervídeo.

A tela ilustrada na Figura 5.11 mostra os primeiros momentos da execução do protótipo. Em primeiro plano está o protótipo exibindo o hipervídeo, enquanto no segundo plano estão as informações hipermídia filtradas do próprio hipervídeo que vem sendo exibido. Ao final da interpretação de um grupo de imagens, a quantidade de imagens daquele grupo também é mostrada. No caso desse exemplo, dez imagens formam um grupo de imagens convencional.

Finalmente, na tela da Figura 5.12, está registrado o momento logo após o clique do "*mouse*" em um determinado ponto do hipervídeo. Nesse momento em particular foi disparado um clique nas co-

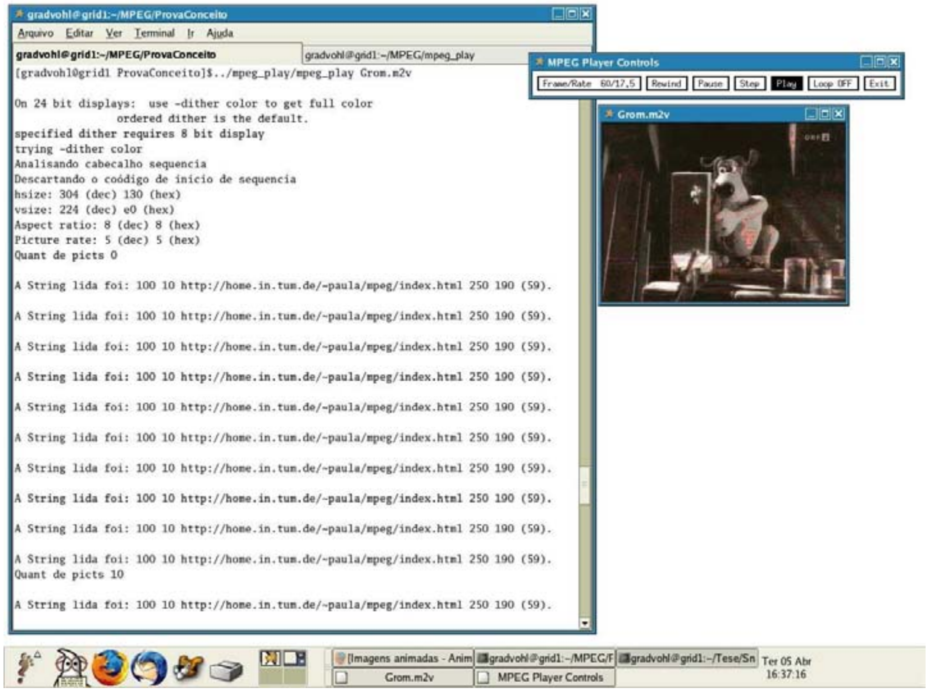


Figura 5.11: Execução do hipervídeo.

ordenadas `(160,149)`. Essa ação invocou implicitamente um comando[5] que instanciou o navegador no endereço indicado pelas informações hipermídia (`http://home.in.tum.de/paula/mpeg/index.html`).

Para melhor ilustrar o processo, a Figura 5.13 mostra um outro hipervídeo em execução. Nesse caso, o "*hiperlink*" (`http://pt.wikipedia.org/wiki/Foguete`) está associado ao foguete apresentado. Ao clicar em um ponto dentro da área delimitada pelas informações hipermídia associadas à imagem do foguete, uma janela com o endereço de uma página com informações para a construção de um foguete é exibida. Vê-se, em segundo plano, o ponto exato em que o evento ocorreu (coordenadas `(168, 62)`) e o comando que instanciou o navegador para exibir a página indicada.

[5]O comando `/usr/local/firefox1.0/mozilla-xremote-client "openURL(URL,new-window)"` faz parte do navegador Firefox. O objetivo desse comando é abrir uma nova janela do navegador (`new-window`) na `URL` especificada[48].

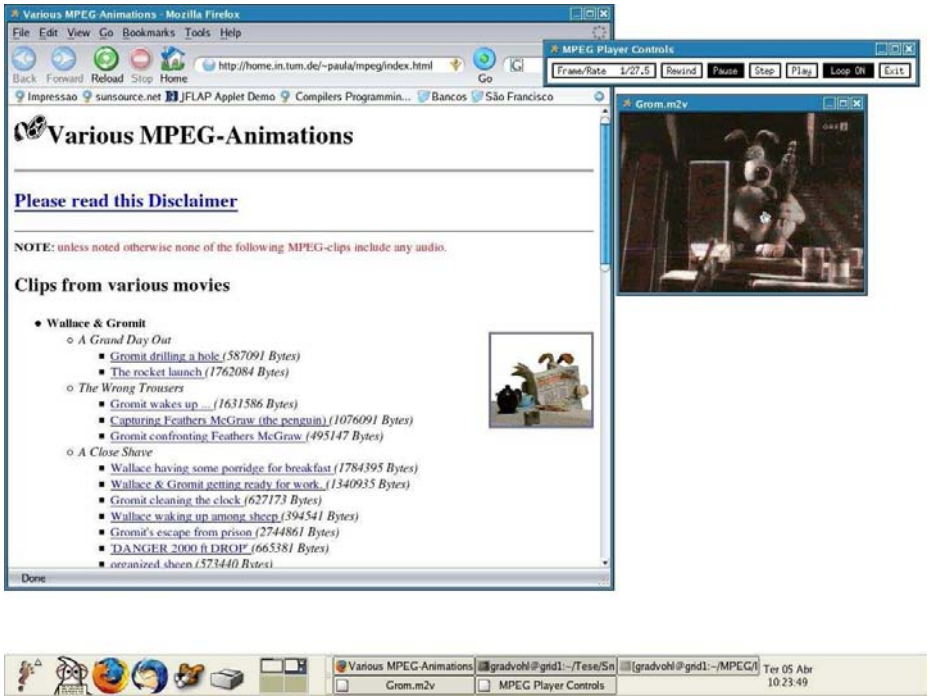


Figura 5.12: Ação do clique do usuário sobre o hipervídeo.

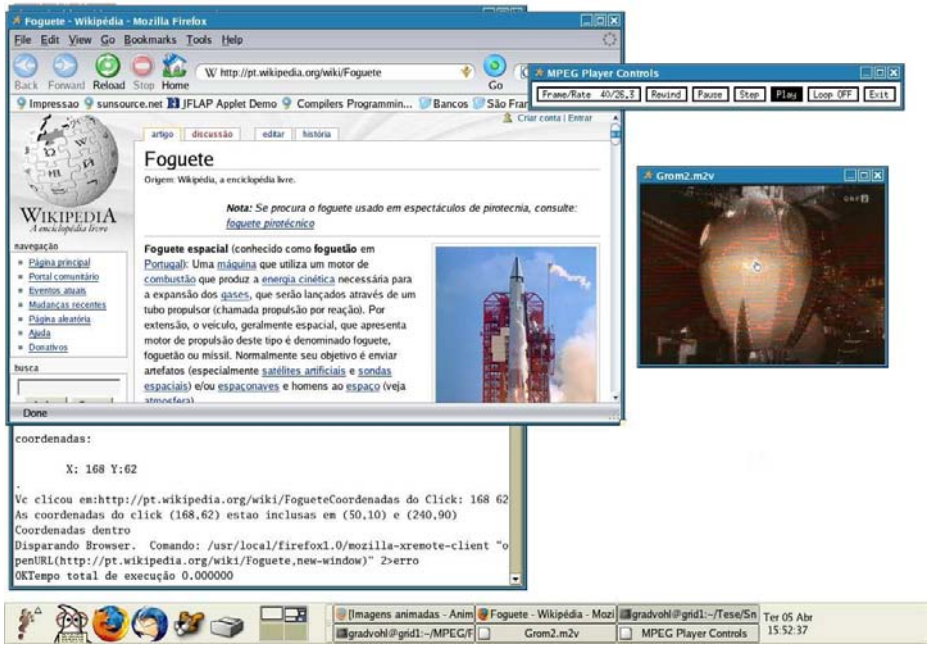


Figura 5.13: Ação do clique do usuário sobre o hipervídeo.

5.6 Sumário

Os focos deste capítulo foram a implementação do protótipo e a demonstração da sua viabilidade. No começo, uma breve discussão a respeito de software livre e proprietário foi iniciada para justificar a escolha do primeiro tipo. Em seguida, foram apresentados trechos de códigos que pretendem elucidar algumas partes do protótipo, uma das contribuições deste trabalho.

Entre os trechos de código mostrados, os primeiros são relativos a algumas das estruturas de dados utilizadas no protótipo. Depois, a implementação do codificador de hipervídeos foi apresentada, indicando os pontos onde as informações hipermídia são inseridas. Em seguida, os detalhes do programa visualizador foram mostrados. O modo como a decodificação ocorre e a forma com que o evento de clique do "*mouse*" é tratado foram detalhados. Os códigos relativos às funções citadas neste capítulo são transcritos na íntegra no Apêndice A, no final do texto.

Finalmente, algumas telas que descrevem a codificação e exibição de um hipervídeo, bem como o acesso às informações hipermídia, foram mostradas. As telas apresentadas são a prova de conceito deste trabalho e confirmam a viabilidade das idéias propostas.

CAPÍTULO 6

Conclusão

"*Parece um teorema sem ter demonstração.*
E parece que sempre termina
Mas não tem fim."
Teorema - Legião Urbana.

No capítulo final deste texto são resumidos os resultados obtidos, discutidas algumas propostas de desenvolvimentos futuros e apresentadas as conclusões obtidas neste trabalho. Tais desenvolvimentos visam complementar e aperfeiçoar o que foi desenvolvido até agora.

6.1 Visão Geral

O foco deste trabalho concentrou-se em dois aspectos: a elaboração e subseqüente implementação de um conceito de hipervídeo. Assim, considerando esse foco, no Capítulo 1 foi proposta uma hipótese que formaliza a motivação para este trabalho. Ainda nesse capítulo foi mostrada a estrutura do texto e os objetivos do trabalho.

O Capítulo 2 tratou do tema convergência digital, esclarecendo-o e mostrando que existe uma

forte tendência de investimentos nessa área. Em particular, mostrou-se através do chamado "*hype cycle*" que as pesquisas em TV Interativa - campo ao qual pertence o conceito de hipervídeo - têm um longo e lucrativo caminho a percorrer. Também nesse capítulo mostraram-se algumas propostas para convergência digital entre TV e Internet que podem tornar-se padrão em um futuro próximo.

A família de padrões MPEG foi abordada no Capítulo 3. Conhecer os detalhes da codificação de vídeos digitais é condição *sine qua non* para propor um conceito de hipervídeo. Sendo assim, nesse capítulo foram abordados alguns algoritmos de compressão, com ou sem perdas, conceitos gerais da família de padrões MPEG e algumas peculiaridades dos mesmos, especificamente as versões 2, 4, 7 e a mais recente 21.

O Capítulo 4 contém uma das principais contribuições deste trabalho: a definição de um conceito de hipervídeo. Em particular, o capítulo tratou da origem do termo "hiper", desde quando a palavra foi cunhada até sua utilização nos dias de hoje, quando se junta a outras palavras para gerar novas idéias como "hipertexto", "hipermídia"e até mesmo "hipervídeo", o foco deste trabalho. Além disso, o Capítulo 4 também definiu um modelo para se implementar o conceito próprio de hipervídeo, detalhando a estrutura das informações hipermídia e como deveriam ser os processos de codificação, decodificação e exibição de um hipervídeo.

A implementação do conceito de hipervídeo foi detalhada no Capítulo 5. Nele, foi discutida e justificada a escolha de ferramentas de software livre para implementação de um programa protótipo. Tal protótipo atua como uma prova de conceito, mostrando a viabilidade da implementação do conceito de hipervídeo proposto neste trabalho. Ainda nesse capítulo foram detalhados alguns aspectos práticos da codificação do protótipo. Começando pelas estruturas de dados projetadas para armazenar as informações hipermídia e continuando com detalhes do programa para codificar, decodificar e exibir um hipervídeo. No final do capítulo, a prova de conceito foi demonstrada em detalhes, apresentando algumas telas que exibem a execução do programa.

6.1.1 Constatação da Hipótese

Na Seção 1.2, página 2, a hipótese que norteou este trabalho foi enunciada. Basicamente, é questionada a possibilidade da criação de um programa de TV que contenha "*hyperlinks*" para conteúdo adicional àquele exibido.

Pode-se afirmar, portanto, que a hipótese é comprovada a partir de um modelo de inserção de "*hyperlinks*" e exibição de hipervídeo, proposto no Capítulo 4 e a partir da execução da prova de conceito do modelo proposto, no Capítulo 5.

6.1.2 Informações Quantitativas do Protótipo

O código do protótipo desenvolvido é extenso e trabalhoso. A fim de quantificar alguns parâmetros são mostrados alguns dados para se perceber a complexidade do protótipo. São eles:

- O programa responsável pela codificação foi gerado por 19 arquivos-fonte e 4 arquivos cabeçalho[1], totalizando 8.955 linhas de código (incluindo comentários).
- O tamanho do programa codificador é de aproximadamente 89 kilobytes e, quando está em execução, ocupa cerca de 1,5 megabytes de memória principal.
- O tempo que é consumido para codificar 128 quadros, gerando um hipervídeo de 4 segundos, é de aproximadamente 12 segundos[2].
- O arquivo que contém um hipervídeo com 128 quadros ocupa cerca de 600 kilobytes, contra 573 kilobytes do arquivo no formato MPEG usado para gerar o hipervídeo.
- O programa responsável pela exibição foi gerado por 26 arquivos-fonte e 9 arquivos cabeçalho, totalizando 21.706 linhas de código (incluindo comentários).
- O tamanho do programa que exibe o hipervídeo é de aproximadamente 142 kilobytes e, quando está em execução, ocupa cerca de 2 megabytes de memória principal.
- O tempo entre o clique do "*mouse*" e a exibição das informações complementares é de aproximadamente dois segundos.

6.2 Síntese dos Resultados

Os resultados obtidos neste trabalho foram:

[1] Os arquivos de cabeçalho contêm definições de constantes, protótipos de funções e macros.
[2] Considerando a execução em um processador `Athlon XP 1800+`.

- um novo conceito de hipervídeo que pode, brevemente, servir como base para uma nova forma de comunicação, mais interativa e com maior capacidade de entretenimento e informação;
- um protótipo de um software capaz de implementar o conceito de hipervídeo em equipamentos com poucos recursos e, conseqüentemente, de baixo custo;
- um resumo das tecnologias atuais que implementam técnicas da chamada convergência digital e podem evoluir aglutinando, entre outros, o conceito de hipervídeo e ainda servir como referência para um futuro padrão brasileiro de TV interativa;
- um resumo da família de padrões MPEG que serve como referência para estudos posteriores.

6.3 Trabalhos Futuros

Algumas questões não foram tratadas em detalhes por estarem fora do escopo deste texto, merecendo um aprofundamento em trabalhos futuros, como os sugeridos a seguir:

- aprimoramento do módulo de codificação dos "*hyperlinks*" para suportar codificação ao vivo;
- melhoramento do módulo de exibição do protótipo para tornar mais evidente a posição dos "*hyperlinks*", sem comprometer a qualidade da imagem;
- implementação e teste da prova de conceito em "*set-top boxes*" e televisores reais;
- implementação e teste do sistema em redes difusoras;
- estudo de usabilidade do protótipo;
- solução para o problema da sobreposição de objetos;
- desenvolvimento de um sistema integrado de visualização de hipervídeo e exibição de informações complementares;
- um protótipo de ferramenta que, depois de aprimorada, pode ser útil em técnicas de educação à distância.

6.4 Conclusões Obtidas

Duas principais conclusões, constatadas a partir da observação do protótipo e da sua execução, podem ser obtidas deste trabalho. São elas:

1. o conceito de hipervídeo, da forma como foi proposta neste trabalho, permite a associação de objetos em determinados contextos a informações complementares, criando assim uma nova estratégia para a TV interativa e uma nova abordagem para o conceito de hipervídeo.

2. é possível implementar uma solução para TV interativa onde os programas de TV atuam como pontos de partida para busca e navegação por informações complementares ao conteúdo sendo transmitido;

Portanto, as idéias propostas neste trabalho vislumbram a criação de um novo tipo de TV interativa. O padrão definitivo para esse tipo de TV ainda não está concluído, o que significa que as discussões ainda não estão encerradas e que ainda há muito a fazer.

Olhando um pouco para o futuro, espera-se que este trabalho tenha suscitado idéias que semeiem novas estratégias de educação à distância, entretenimento e propaganda, entre outras. No caso particular da educação à distância, o conceito de hipervídeo pode ser uma ferramenta interessante para despertar a curiosidade para aprofundar-se em determinados assuntos que foram despertados ao assistir à televisão.

Definições e Abreviaturas

"*It is with words as with sunbeams,*
the more they are condensed, the deeper they burn."
Robert Southey.

As seguintes definições de termos e abreviaturas são usadas neste texto:

"*Browser*" Também chamado de navegador, um "*browser*" é um programa específico para visualizar hipertextos, em particular aqueles disponíveis na WWW.

Camadas de Objeto Vídeo As informações sobre forma, movimento e textura dos Planos de Objetos Vídeo são codificados em camadas de objeto vídeo.

Codificação Huffman Método de compressão sem perdas aplicado em uma das fases da família de padrões MPEG. Mais informações na página 12. Ver também Código de Tamanho Variável.

Código de Tamanho Variável Código produzido por um esquema de compressão sem perdas cuja entrada é um bloco de símbolos de tamanho fixo e produz como saída um código com tamanho variável, baseado nas probabilidades dos símbolos de entrada. Ver Codificação Huffman e mais detalhes na página 12.

DCT Ver "*Discrete Cosine Transform*".

Decodificação Huffman Método de descompressão sem perdas aplicado em uma das fases da família de padrões MPEG. Mais informações na página 14.

Declaração de Item Digital Esquema interoperável, flexível e uniforme para declarar items digitais.

Descritor Elemento que define a sintaxe e a semântica de uma característica audiovisual no MPEG-4.

DID Ver Declaração de Item Digital.

"*Digital Video Broadcast*" Um padrão internacional para a TV digital. Este padrão transmite em um canal entre 6 e 8 MHz e usa o padrão MPEG-2.

"*Discrete Cosine Transform*" Método de compressão com perdas, fundamentado na observação de que "*pixels*" vizinhos são correlacionados. Mais detalhes na página 14.

DVB Ver "*Digital Video Broadcast*".

Endereço IP Endereço utilizado na camada de rede na arquitetura de protocolos TCP/IP. Tal endereço especifica unicamente uma máquina conectada à rede que implementa a arquitetura TCP/IP. Ver também IP.

Esquema de Descrição Especifica a estrutura e a semântica das relações entre os componentes, no padrão MPEG-4.

Erro Mínimo Quadrático Técnica estatística que evidencia os erros de aproximação.

"*eXtensible Markup Language*" É um padrão especificado pelo W3C para estruturação de documentos e dados na WWW. Essa linguagem possui a mesma estrutura da linguagem HTML, ou seja, utiliza-se de rótulos ("*tags*") e atributos para delimitar "pedaços" de dados e deixa que a interpretação desses dados seja feita pela aplicação que os lê.

Formato entrelaçado Formato de vídeo onde cada quadro é composto por linhas pares (campos superiores) e linhas ímpares (campos inferiores).

Formato não-entrelaçado Formato de vídeo onde as linhas são iluminadas seqüencialmente.

Formato progressivo Ver Formato não-entrelaçado.

GNU uma sigla recursiva que significa "*GNU is Not Unix*". Esta é a sigla de um projeto que desenvolve e mantém um ambiente completo de software, incluindo sistema operacional, utilitários, entre outros. Este projeto é apoiado pela "*Free Software Foundation*".

Hiperlink Uma ligação ou referência entre um objeto (geralmente um hipertexto) e outro.

Hipermídia Generalização do conceito de hipertexto. Nesse caso, o conteúdo dos documentos interligados pode estar implementado em diferentes mídias (áudio, texto, vídeo ou uma combinação desses diferentes meios de comunicação). Detalhes na página 41.

Hipertexto Uma abordagem para gerenciamento de informação tal que os dados são organizados em nós, conectados por "*links*" (referências).

Hipervídeo Particularização do conceito de hipertexto onde o ponto de partida para navegação entre os diferentes documentos conectados é um vídeo. Mais informações na página 41.

Histograma Indica a distribuição dos "*pixels*" para cada nível de intensidade de cor.

HTML Ver "*HyperText Markup Language*".

"*Hype Cycle*" Representação gráfica para um modelo de maturidade, adoção e aplicação comercial de tecnologias emergentes. Mais detalhes na página 6.

HyperText Markup Language É uma linguagem de marcação de hipertextos. Ela utiliza rótulos para especificar referências para outros documentos e como informações serão apresentadas em um programa específico para navegação na WWW ("*browser*").

IDCT Ver "*Inverse Discrete Cosine Transform*".

Informações hipermídia No âmbito deste trabalho, informações hipermídia são um conjunto de dados que contêm uma referência a um endereço na WWW, onde estão informações complementares sobre um objeto exibido em cena e as coordenadas de um retângulo que contém esse objeto. Detalhes na página 45.

Intraquadros Ver Quadros I.

"*Internet Protocol*" Protocolo da camada de rede que facilita roteamento e controle de congestionamento. Ver também Endereço IP.

"*Inverse Discrete Cosine Transform*" Função inversa da função DCT. Ou seja, transforma as coordenadas do domínio da freqüência para o domínio espacial. Detalhes na página 17.

"*International Standardization Organization*" Organização que estabelece padrões internacionais.

IP Ver "*Internet Protocol*".

ISO Ver "*International Standardization Organization*".

Macrobloco Cada quadro em uma seqüência de quadros que compõem um vídeo é dividido em áreas de 16×16 "*pixels*". Essas áreas são chamadas de macroblocos.

MHEG Ver "*Multimedia and Hypermedia information coding Expert Group*".

MHP Ver "*Multimedia Home Platform*".

Mídia Oriunda do grego, esta palavra designa algo que ocupa uma posição entre dois extremos. No caso deste texto, refere-se a um meio de comunicação.

Modo de Predição "*Dual Prime*" Método de compensação de movimento que explora de forma eficiente a redundância temporal entre campos.

MPEG Ver "*Moving Pictures Expert Group*".

"*Moving Pictures Expert Group*" grupo de especialistas em imagens em movimento. É um grupo de trabalho da "*International Standardization Organization*" que desenvolve padrões para imagens em movimento.

"*Multimedia and Hypermedia information coding Expert Group*" grupo que propõe a padronização de estratégias para integração de informações multimídia. Também é o nome de um padrão para integração de informações multimídia. Mais informações na página 7.

"*Multimedia Home Platform*" Padrão proposto pela "*Digital Video Broadcast*" para combinar TV digital com a WWW. Detalhes na página 8.

Navegador Ver "*browser*".

Objeto Vídeo É um conjunto de planos de vídeos sucessivos que referenciam um objeto em cena.

Planos de Objetos de Vídeo Regiões segmentadas em quadros que formam um vídeo.

Pixel Abreviatura de "*Picture Element*". Corresponde a cada ponto que compõe uma imagem.

Prova de Conceito Demonstração de um conceito para demonstrar sua viabilidade.

Quadros B quadros bidirecionalmente preditivos ou quadros bidirecionais que buscam as informações para sua compressão em quadros I ou P. Esse tipo de quadro possui o mais alto grau de compressão entre os tipos de quadros.

Quadros Bidirecionais Ver Quadros B.

Quadros Bidirecionalmente Preditivos Ver Quadros B.

Quadros I ou intraquadros são quadros que contêm toda a informação pertinente a sua compressão. Em outras palavras, eles não precisam referenciar outros quadros para obter informações de compressão.

Quadros Interpolativos Ver Quadros I.

Quadros P ou quadros preditivos são comprimidos a partir de quadros I ou de outros quadros P. Isso significa que uma parte das informações para compressão desses quadros vem de quadros I ou de outros quadros P. A página 18 apresenta mais detalhes.

Quadros Preditivos Ver Quadros P.

Quantização Processo de descarte de informações menos relevantes com vistas ao aumento do grau de compressão. Mais informações na página 16.

Redundância Espacial Regiões próximas tendem a ter as mesmas características, i. e., são redundantes.

Redundância Temporal Quadros seguidos têm uma grande probabilidade de apresentar regiões redundantes.

RGB Sigla para "*Red*", "*Green*" e "*Blue*". É um formato que obtém uma cor com base nas três cores básicas: vermelho, verde e azul.

"*Set-top box*" Dispositivo que recebe, demultiplexa e trata um sinal digital. O "*set-top box*" é muito utilizado na TV digital para demultiplexar um sinal recebido em vários outros (sinal de áudio, vídeo e, eventualmente, dados) e tratá-los de acordo com seu tipo [21].

SMIL Ver "*Synchronized Multimedia Integration Language*".

SNR Sigla para "*Signal to Noise Ratio*": razão entre a potência ou volume (amplitude) de um sinal e a quantidade de interferência indesejada (ruído) que veio misturada ao sinal. Em linhas gerais essa razão mede a clareza de um sinal em um canal.

"*Synchronized Multimedia Integration Language*" É uma linguagem para especificação de apresentações audiovisuais. Essa especificação inclui a integração de diferentes tipos de mídia. Detalhes na página 8.

Transformadas Funções que transformam um sinal de um domínio para outro e, com isso, podem obter um certo grau de compressão com perdas.

Transformada Discreta do Co-seno Ver DCT.

Transformada Inversa Discreta do Co-seno Ver IDCT.

Uniform Resource Locator Localizador Uniforme de Recursos. Endereço de um recurso na Internet. Esse recurso é um arquivo que contém som, imagem, texto ou uma mistura de todos esses elementos.

URL Ver "*Uniform Resource Locator*".

Variable Length Code Código de Tamanho Variável. Ver VLC.

VLC "*Variable Length Code*". Ver Código de Tamanho Variável.

VO "*Video Object*". Ver Objeto Vídeo.

VOP "*Video Object Plane*". Ver Planos de Objeto Vídeo.

VOL "*Video Object Layer*". Ver Camada de Objeto Vídeo.

XML Ver "*eXtensible Markup Language*".

YCrCb Ver YUV.

YUV Modelo de cores tipicamente usado para codificação de vídeo. O Y é o sinal de luminosidade. O U e o V são os sinais de diferença de cores. U é o sinal da cor vermelha menos o sinal Y ($U = R - Y$). V é o sinal da cor azul menos o sinal Y ($V = B - Y$). Esse modelo também é conhecido como $YCrCb$. $Cr = R - Y$ e $Cb = B - Y$.

W3C Ver "*World Wide Web Consortium*".

"*World Wide Web*" Serviço implementado na Internet que referencia documentos que estão armazenados no mesmo computador ou remotamente, em uma outra máquina, possivelmente em outra rede.

"*World Wide Web Consortium*" Consórcio formado por indústrias, fundado em 1994 para desenvolvimento de padrões comuns para a WWW.

WWW Ver "*World Wide Web*".

Referências Bibliográficas

[1] S. M. Akramullah, I. Ahmad, and M. L. Liou. Performance of Software-Based MPEG-2 Video Encoder on Parallel and Distributed Systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(4):687–695, Agosto de 1997.

[2] Berkley Multimedia Research Center. Berkley MPEG Tools. *Disponível no endereço http://bmrc.berkeley.edu/frame/research/mpeg*, Agosto de 2001.

[3] T. Berners-Lee. Information Management: a proposal. *Disponível no endereço http://www.w3.org/History/1989/proposal.html*, Maio 1990.

[4] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards*. Kluwer Academic Publishers, Amsterdã, Holanda, 1995.

[5] J. Bormans and K. Hill. MPEG-21 Overview. *Disponível no endereço http://mpeg.telecomitalialab.com/standards/mpeg-21/mpeg-21.htm*, Maio de 2002.

[6] V. Bush. As we may think. *Atlantic Monthly*, Julho 1945. *Disponível no endereço http://www.ps.uni-sb.de/ duchier/pub/vbush/vbush.shtml*.

[7] S. Chang, T. Sikora, and A. Puri. Overview of MPEG-7 Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):688–694, Junho de 2001.

[8] J. Chen and K. J. R. Liu. Efficient Architecture and Design of an Embedded Video Coding Engine. *IEEE Transactions on Multimedia*, 3(3):285–297, Setembro de 2001.

[9] H. Cossmann, C. Griwodz, G. Grassel, M. Puhlhofer, M. Schreiber, R. Steinmetz, H. Wittig, and L. Wolf. Interoperable ITV systems based on MHEG. In *Multimedia Computing and Networking, Proceedings of the SPIE - The International Society for Optical Engineering*, páginas 60–68, 1995.

[10] J. Cristy. Imagemagick 6.1.8. *Disponível no endereço http://www.imagemagick.org*, Janeiro de 2005.

[11] A. Daniels. The Multimedia Home Platform: Creating a Multimedia World in the STB. *Disponível no endereço http://www.commsdesign.com/story/OEG20020918S0011*, Setembro de 2002.

[12] Digital Video Broadcast. *Multimedia Home Platform*, Outubro de 2001.

[13] Digital Video Broadcast. *Multimedia Home Platform Specification 1.0.3*, Junho de 2003.

[14] A. Docef, F. Kossentini, K. Nguuyen-Phi, and I. R. Ismaeil. The Quantized DCT and Its Application to DCT-Based Video Coding. *IEEE Transactions on Image Processing*, 11(3):177–187, Março de 2002.

[15] S. Done. MHEG - A Multimedia Presentation Standard. *Atlantic Monthly*, Junho 1996. *Disponível no endereço http://www.doc.ic.ac.uk/nd/surprise_96/journal/ vol2/srd2/article2.htm*.

[16] M. Echiffre, C. Marchisio, P. Marchisio, P. Panicciari, and S. Del Rossi. MHEG-5: aims, concepts, and implementation issues. *IEEE Multimedia*, 5(1):84–91, Março 1998.

[17] Free Software Foundation. GCC home page. *Disponível no endereço http://gcc.gnu.org*, Dezembro de 2004.

[18] Free Software Foundation. Phylosophy of the GNU Project. *Disponível no endereço http://www.gnu.org/philosophy*, Novembro de 2004.

[19] B. Furht. Multimedia systems: an overview. *IEEE Multimedia*, 1(1):47–59, 1994.

[20] B. Furht et al. Design issues for interactive television systems. *Computer*, 28(5):25–39, Maio de 1995.

[21] M. Gaillard. Massive Digital Deployment: winning the set-top war. *Communications Technology Magazine*, páginas 84–86, Junho de 2001.

[22] Gartner Research. Emerging technology hype cycle. Technical report, Gartner Group, Julho 2003.

[23] A. Gil, J. Pazos, R. Díaz, M. Fernández, and M. Ramos. Internet-TV Convergence in DVB-MHP. In *2002 IEEE International Conference on Multimedia and Expo*, volume 2, páginas 285–288, Agosto de 2002.

[24] A. Gil, J. Pazos, C. López, J. López, R. Rubio, R. Díaz, and M. Ramos. Surfing the Web on TV: the MHP Approach. In *4th EURASIP-IEEE Symposium on Video/Image Processing and Multimedia Communications*, páginas 447–451, Zadar, Croácia, Junho de 2002.

[25] K. L. Gong and L. A. Rowe. Parallel MPEG-1 Video Encoding. In *Picture Coding Symposium*, páginas 415–425, Setembro de 1994.

[26] R. Goularte, E. dos Santos Moreira, and M. da Graça C. Pimentel. Structuring interactive tv documents. In *DocEng '03: Proceedings of the 2003 ACM symposium on Document engineering*, páginas 42–51. ACM Press, 2003.

[27] A. L. S. Gradvohl. *Multiprogramação*. *Disponível no endereço http://www.cenapad.unicamp.br/servicos/treinamentos/multiprog.shtml*, 2004.

[28] Y. He, I. Ahmad, and M. L. Liou. Real-Time Interactive MPEG-4 System Encoder Using a Cluster of Workstations. *IEEE Transactions on Multimedia*, 1(2):217–233, Junho de 1999.

[29] J. Hunter. MPEG-7 Behind the Scenes. *D-Lib Magazine*, 5(9), 1999.

[30] H. Kalva, L. Tang, J. Huard, G. Tselikis, J. Zamora, L. Cheok, and A. Eleftheriadis. Implementing Multiplexing, Streaming and Serving Interaction for MPEG-4. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1299–1311, Dezembro de 1999.

[31] B. Kennelly. The History of Hypertext. *Disponível no endereço http://www.historyofhypertext.co.uk*, Outubro de 2004.

[32] S. Kiousis. Interactivity: a concept explanation. *New Media & Society*, 4(3):355–383, 2002.

[33] P. Laine. Explicitness and interactivity. In *ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies*, páginas 421–426. Trinity College Dublin, 2003.

[34] C. Lin, J. Zhou, J. Youn, and M. Sun. MPEG Video Streaming with VCR Functionality. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):415–425, Março de 2001.

[35] D. Marshall. *Programming in C: UNIX System Calls and Subroutines using C. Disponível no endereço http://www.cs.cf.ac.uk/Dave/C/CE.html*, 1999.

[36] D. Marshall. *X Window/Motif Programming. Disponível no endereço http://www.cs.cf.ac.uk/Dave/X_lecture/X_book_caller/X_book_caller.html*, 1999.

[37] MPEG-7 Description Scheme Group. *MPEG-7 Description Schemes (V0.5)*. ISO/IEC SC29/WG11 N2844, Vancouver, Julho de 1999.

[38] MPEG Software Simulation Group. MSSG. *Disponível no endereço http://www.mpeg.org/MPEG/MSSG*, Maio de 2000.

[39] T. H. Nelson. Complex information processing: a file structure for the complex, the changing and the indeterminate. In *Anais da 20ª conferência Nacional da ACM*, páginas 84–100, Cleveland, Ohio, EUA, Agosto 1965. ACM Press.

[40] R. Neogi and A. Saha. Embedded Parallel Divide-and-Conquer Video Decompression Algorithm and Architecture for HDTV Applications. *IEEE Transactions on Consumer Electronics*, 41(1):160–170, Fevereiro de 1995.

[41] K. Panusopone, X. Chen, R. Eifrig, and A. Luthra. Coding Tools in MPEG-4 for Interlaced Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(5):755–766, Agosto de 2000.

[42] N. Phamdo. Lossless Data Compression. *Disponível no endereço http://www.data-compression.com/lossless.html*, 2001.

[43] K. Pihkala. *Extensions to the SMIL Multimedia Language*. Tese de doutorado, Helsinki University of Technology, Novembro 2003.

[44] N. Sawhney, D. Balcom, and I. Smith. Authoring and navigating video in space and time. *IEEE Multimedia*, (4):30–39, Outubro-Dezembro de 1997.

[45] Sun Microsystems. *Multithreaded Programming Guide*, Maio 2002.

[46] J. Twist. Peering beyond the technology hype. *Disponível no endereço http://news.bbc.co.uk/2/hi/technology/3577746.stm*, Agosto 2004.

[47] D. van Heesch. Doxygen. *Disponível no endereço http://www.doxygen.org*, Dezembro de 2004.

[48] D. Wang. Mozilla's command line options. *Disponível no endereço http://www.mozilla.org/docs/command-line-args.html*, Junho de 2004.

[49] P. Wilson-Smith. Fund managers take fright over myners. *Financial News*, Março 2001.

[50] J. Wiseman. An Introduction to MPEG Video Compression. *Disponível no endereço http://www.siggraph.org/education/materials/HyperGraph/video/mpeg*, Junho de 2002.

[51] World Wide Web Consortium. *XML Schema Part 1: Structures*, Maio de 1999. *Disponível no endereço http://www.w3.org/1999/05/06-xmlschema-1*.

[52] World Wide Web Consortium. *XML Schema Part 2: Datatypes*, Maio de 1999. *Disponível no endereço http://www.w3.org/1999/05/06-xmlschema-2*.

[53] World Wide Web Consortium. *Synchronized Multimedia Integration Language (SMIL) 2.0 Specification*, Agosto de 2001. *Disponível no endereço http://www.w3.org/TR/2001/REC-smil20-20010807*.

[54] World Wide Web Consortium. *Extensible Markup Language*, Agosto de 2004. *Disponível no endereço http://www.w3.org/XML*.

[55] T. Worthington. Internet-TV Convergence with the Multimedia Home Platform. *Disponível no endereço http://www.tomw.net.au/2001/itv.html*, Setembro de 2001.

[56] XFree86 Project. XFree86: Home to the X Window System. *Disponível no endereço http://www.xfree86.org*, Dezembro de 2004.

[57] C. Yang and Y. Yang. SMILAuthor: an authoring system for SMIL-based multimedia presentations. In *Multimedia Tools and Applications*, number 21, páginas 243–260, Holanda, 2003. Kluwer Academic Publishers.

APÊNDICE A

Principais Trechos de Códigos do Protótipo

"*Verba Volant. Scripta Manent.*[1]"

Provérbio Latino.

O código inteiro do protótipo implementado neste trabalho é muito extenso, tornando inviável sua reprodução por completo nesse texto. Todavia, as funções citadas no Capítulo 5 serão descritas a seguir, na íntegra.

As oito próximas figuras (A.1 a A.8) descrevem a função `putpic`, responsável por calcular e incluir o cabeçalho da camada de imagem.

```
void putpict(unsigned char *frame)
{
  int i, j, k, comp, cc, mb_type, prev_mquant,cbp, MBAinc=0;
  int PMV[2][2][2];
  char *res;
```

Figura A.1: Início da função `putpict`.

[1] "Discussões são esquecidas. O código permanece."

```
rc_init_pict(frame); /* set up rate control */
/* picture header and picture coding extension */
putpicthdr();

rc_init_pict(frame);        // Definicao do controle de taxa.
putpicthdr();               // Grava cabecalho da imagem corr.
res = le(arquivoHiperlinks);// Le o arquivo de Hiperlinks
if (res != NULL)
   putpicthdrext(res);      // Grava informacoes hipermidia.
 else
    putpicthdrext("NULO ");
 if (!mpeg1)
   putpictcodext();

 /* initialize quantization parameter */
 prev_mquant = rc_start_mb();

 k = 0;

 for (j=0; j<mb_height2; j++)
 {
   /* macroblock row loop */

   for (i=0; i<mb_width; i++)
   {
     /* macroblock loop */
     if (i==0)
     {
       /* slice header (6.2.4) */
       alignbits();

      if (mpeg1 || vertical_size<=2800)
        putbits(SLICE_MIN_START+j,32); /* slice_start_code */
      else
      {
       /* slice_start_code */
        putbits(SLICE_MIN_START+(j&127),32);
        putbits(j>>7,3); /* slice_vertical_position_extension*/
      }
```

Figura A.2: Continuação da função `putpict`.

```
      /* quantiser_scale_code */
    putbits(q_scale_type ? map_non_linear_mquant[prev_mquant]
                         : prev_mquant >> 1, 5);
    putbits(0,1); /* extra_bit_slice */

    /* reset predictors */
    for (cc=0; cc<3; cc++)
      dc_dct_pred[cc] = 0;

    PMV[0][0][0]=PMV[0][0][1]=PMV[1][0][0]=PMV[1][0][1]=0;
    PMV[0][1][0]=PMV[0][1][1]=PMV[1][1][0]=PMV[1][1][1]=0;

    MBAinc = i + 1; /*first MBAinc denotes absolute position*/
  }
  mb_type = mbinfo[k].mb_type;

  /* determine mquant (rate control) */
  mbinfo[k].mquant = rc_calc_mquant(k);

  /* quantize macroblock */
  if (mb_type & MB_INTRA)
  {
    for (comp=0; comp<block_count; comp++)
      quant_intra(blocks[k*block_count+comp],
                  blocks[k*block_count+comp],
                  dc_prec,intra_q,mbinfo[k].mquant);
    mbinfo[k].cbp = cbp = (1<<block_count) - 1;
  }
  else
  {
   cbp = 0;
   for (comp=0;comp<block_count;comp++)
     cbp =
      (cbp<<1) |  quant_non_intra(blocks[k*block_count+comp],
                                  blocks[k*block_count+comp],
                                  inter_q,mbinfo[k].mquant);
      mbinfo[k].cbp = cbp;
    if (cbp)
      mb_type|= MB_PATTERN;
  }
```

Figura A.3: Continuação da função `putpict`.

```
/* output mquant if it has changed */
if (cbp && prev_mquant!=mbinfo[k].mquant)
  mb_type|= MB_QUANT;

/* check if macroblock can be skipped */
if (i!=0 && i!=mb_width-1 && !cbp)
{
  /* no DCT coefficients and neither first nor last
      macroblock of slice */
 if (pict_type==P_TYPE && !(mb_type&MB_FORWARD))
 {
  /* P picture, no motion vectors -> skip */

  /* reset predictors */

  for (cc=0; cc<3; cc++)
    dc_dct_pred[cc] = 0;

  PMV[0][0][0]=PMV[0][0][1]=PMV[1][0][0]=PMV[1][0][1]=0;
  PMV[0][1][0]=PMV[0][1][1]=PMV[1][1][0]=PMV[1][1][1]=0;

  mbinfo[k].mb_type = mb_type;
  mbinfo[k].skipped = 1;
  MBAinc++;
  k++;
}
```

Figura A.4: Continuação da função `putpict`.

```
if (pict_type==B_TYPE && pict_struct==FRAME_PICTURE
&& mbinfo[k].motion_type==MC_FRAME
&& ((mbinfo[k-1].mb_type^mb_type)&(MB_FORWARD|MB_BACKWARD))==0
&& (!(mb_type&MB_FORWARD) ||
      (PMV[0][0][0]==mbinfo[k].MV[0][0][0] &&
      PMV[0][0][1]==mbinfo[k].MV[0][0][1]))
&& (!(mb_type&MB_BACKWARD) ||
          (PMV[0][1][0]==mbinfo[k].MV[0][1][0] &&
           PMV[0][1][1]==mbinfo[k].MV[0][1][1])))
  {
   /* conditions for skipping in B frame pictures:
    * - must be frame predicted
    * - must be the same prediction type
    *   (forward/backward/interp.)as previous macroblock
    * - relevant vectors (forward/backward/both) have to be
    *   the same as in previous macroblock
    */

    mbinfo[k].mb_type = mb_type;
    mbinfo[k].skipped = 1;
    MBAinc++;
    k++;
    continue;
  }
```

Figura A.5: Continuação da função `putpict`.

```
if (pict_type==B_TYPE && pict_struct!=FRAME_PICTURE
&& mbinfo[k].motion_type==MC_FIELD
&& ((mbinfo[k-1].mb_type^mb_type)&(MB_FORWARD|MB_BACKWARD))==0
&& (!(mb_type&MB_FORWARD) ||
 (PMV[0][0][0]==mbinfo[k].MV[0][0][0] &&
 PMV[0][0][1]==mbinfo[k].MV[0][0][1] &&
 mbinfo[k].mv_field_sel[0][0]==(pict_struct==BOTTOM_FIELD)))
&& (!(mb_type&MB_BACKWARD) ||
 (PMV[0][1][0]==mbinfo[k].MV[0][1][0] &&
  PMV[0][1][1]==mbinfo[k].MV[0][1][1] &&
  mbinfo[k].mv_field_sel[0][1]==(pict_struct==BOTTOM_FIELD))))
   {
     /* conditions for skipping in B field pictures:
      * - must be field predicted
      * - must be the same prediction type
      *   (forward/backward/interp.) as previous macroblock
      * - relevant vectors (forward/backward/both) have to be
      *   the same as in previous macroblock
      * - relevant motion_vertical_field_selects have to be of
      *   same parity as current field
      */
     mbinfo[k].mb_type = mb_type;
     mbinfo[k].skipped = 1;
      MBAinc++;
      k++;
      continue;
   }
 }
 /* macroblock cannot be skipped */
 mbinfo[k].skipped = 0;

  /* there's no VLC for 'No MC, Not Coded':
   * we have to transmit (0,0) motion vectors    */
  if (pict_type==P_TYPE && !cbp && !(mb_type&MB_FORWARD))
    mb_type|= MB_FORWARD;

  putaddrinc(MBAinc); /* macroblock_address_increment */
  MBAinc = 1;

 putmbtype(pict_type,mb_type); /* macroblock type */
```

Figura A.6: Continuação da função `putpict`.

```
if (mb_type & (MB_FORWARD|MB_BACKWARD) && !frame_pred_dct)
  putbits(mbinfo[k].motion_type,2);

if (pict_struct==FRAME_PICTURE && cbp && !frame_pred_dct)
     putbits(mbinfo[k].dct_type,1);

if (mb_type & MB_QUANT)
{
putbits(q_scale_type?map_non_linear_mquant[mbinfo[k].mquant]
                        : mbinfo[k].mquant>>1,5);
prev_mquant = mbinfo[k].mquant;
}

if (mb_type & MB_FORWARD)
{
  /* forward motion vectors, update predictors */
  putmvs(mbinfo[k].MV,PMV,mbinfo[k].mv_field_sel,
          mbinfo[k].dmvector,0,mbinfo[k].motion_type,
          forw_hor_f_code,forw_vert_f_code);
}

if (mb_type & MB_BACKWARD)
{
  /* backward motion vectors, update predictors */
  putmvs(mbinfo[k].MV,PMV,mbinfo[k].mv_field_sel,
          mbinfo[k].dmvector,1, mbinfo[k].motion_type,
          back_hor_f_code,back_vert_f_code);
}

if (mb_type & MB_PATTERN)
{
     putcbp((cbp >> (block_count-6)) & 63);
     if (chroma_format!=CHROMA420)
       putbits(cbp,block_count-6);
}
```

Figura A.7: Continuação da função `putpict`.

```
      for (comp=0; comp<block_count; comp++)
      {
        /* block loop */
        if (cbp & (1<<(block_count-1-comp)))
        {
          if (mb_type & MB_INTRA)
          {
            cc = (comp<4) ? 0 : (comp&1)+1;
            putintrablk(blocks[k*block_count+comp],cc);
          }
          else
            putnonintrablk(blocks[k*block_count+comp]);
        }
      }

      /* reset predictors */
      if (!(mb_type & MB_INTRA))
        for (cc=0; cc<3; cc++)
          dc_dct_pred[cc] = 0;

      if (mb_type & MB_INTRA ||
          (pict_type==P_TYPE && !(mb_type & MB_FORWARD)))
      {
        PMV[0][0][0]=PMV[0][0][1]=PMV[1][0][0]=PMV[1][0][1]=0;
        PMV[0][1][0]=PMV[0][1][1]=PMV[1][1][0]=PMV[1][1][1]=0;
      }

      mbinfo[k].mb_type = mb_type;
      k++;
    }
  }

  rc_update_pict();
  vbv_end_of_picture();
}
```

Figura A.8: Final da função `putpict`.

Os trechos de código mostrados nas figuras a seguir (A.9 a A.14) descrevem a função `mpegVidRsrc`.

```
VidStream *mpegVidRsrc(TimeStamp time_stamp,
                       VidStream *vid_stream,
                       int first,
                       XInfo *xinfo)
{
  unsigned int data;
  int i, status;

  /* If vid_stream is null, create new VidStream structure. */
  if (vid_stream == NULL) {
    return NULL;
  }

  /*
   * If called for the first time, find start code, make sure
   * it is a sequence start code.
   */

  if (first) {
    vid_stream->sys_layer=-1;
    vid_stream->num_left=0;
    vid_stream->leftover_bytes=0;
    vid_stream->seekValue=0;
    vid_stream->Parse_done=FALSE;
    next_start_code(vid_stream);  /* sets curBits */
    show_bits32(data);
    if (data != SEQ_START_CODE) {
      fprintf(stderr, "This is not an MPEG video stream. (%x)\n"
                       ,data);
      DestroyVidStream(vid_stream,xinfo);
      return NULL;
    }
  } else {
    vid_stream->curBits = *vid_stream->buffer;
  }
```

Figura A.9: Início da função `mpegVidRsrc`.

```
  /* Get next 32 bits (size of start codes). */
  show_bits32(data);

  /*
   * Process according to start code (or parse macroblock if not
   * a start code at all).
   */
  switch (data) {
  case SEQ_END_CODE: puts("Fim sequencia");
  case 0x000001b9:   /*  handle ISO_11172_END_CODE too */

    /* Display last frame. */
    if (vid_stream->future != NULL) {
      vid_stream->current = vid_stream->future;
#ifndef NOCONTROLS
      ExecuteDisplay(vid_stream, 1, xinfo);
#else
      ExecuteDisplay(vid_stream, xinfo);
#endif
    }
    /*Sequence done. Do the right thing. For right now, exit.*/
    if (!quietFlag) {
      fprintf(stderr, "\nDone!\n");
    }

#ifdef ANALYSIS
    PrintAllStats(vid_stream);
#endif
    PrintTimeInfo(vid_stream);

    vid_stream->film_has_ended=TRUE;
#ifdef NOCONTROLS
    if (loopFlag) {
      clear_data_stream(vid_stream);
    } else DestroyVidStream(vid_stream, xinfo);
#endif /* !NOCONTROLS */
    goto done;
    break;
```

Figura A.10: Continuação da função `mpegVidRsrc`.

```
  case SEQ_START_CODE:
    /* Sequence start code. Parse sequence header. */
    puts ("Analisando cabecalho sequencia");
    if (ParseSeqHead(vid_stream,xinfo) != PARSE_OK)
      goto error;

    /*
     * Return after sequence start code so that application
     * above can use info in header.
     */
    if (vid_stream->seekValue > 0) {
      SeekStream(vid_stream);
    }
    goto done;

  case GOP_START_CODE:
    /* Group of Pictures start code. Parse gop header. */
    if (ParseGOP(vid_stream) != PARSE_OK)
      goto error;
    goto done;

  case PICTURE_START_CODE:

 /* Picture start code.
  * Parse picture header and first slice header.
  */
    status = ParsePicture(vid_stream, time_stamp);

    if (status == SKIP_PICTURE) {
      next_start_code(vid_stream);
      while (!next_bits(32, PICTURE_START_CODE, vid_stream)) {
        if (next_bits(32, GOP_START_CODE, vid_stream))
          break;
        else if (next_bits(32, SEQ_END_CODE, vid_stream))
          break;
        flush_bits(24);
        next_start_code(vid_stream);
      }
      goto done;
    } else if (status != PARSE_OK)
      goto error;
```

Figura A.11: Continuação da função `mpegVidRsrc`.

```
  ParsePictureExtension(vid_stream);
  if (ParseSlice(vid_stream) != PARSE_OK)
    goto error;
  break;

case SEQUENCE_ERROR_CODE:
  flush_bits32;
  next_start_code(vid_stream);
  goto done;

default:

  /* Check for slice start code. */
  if ((data >= SLICE_MIN_START_CODE)
   && (data <= SLICE_MAX_START_CODE))
 {
    /* Slice start code. Parse slice header. */
    if (ParseSlice(vid_stream) != PARSE_OK)
      goto error;
  }
  break;
}

/* Parse next MB_QUANTUM macroblocks. */
for (i = 0; i < MB_QUANTUM; i++) {
  /* Check to see if actually a startcode
   * and not a macroblock.
   */
  if (!next_bits(23, 0x00000000, vid_stream)) {

    /* Not start code. Parse Macroblock. */
    if (ParseMacroBlock(vid_stream) != PARSE_OK)
      goto error;
```

Figura A.12: Continuação da função `mpegVidRsrc`.

```
#ifdef ANALYSIS
      if (showmb_flag) {
        DoDitherImage(vid_stream);
#ifndef NOCONTROLS
        ExecuteDisplay(vid_stream, 1, xinfo);
#else
        ExecuteDisplay(vid_stream, xinfo);
#endif /* !NOCONTROLS */
      }
#endif /* ANALYSIS */

    } else {

    /* Not macroblock, actually start code. Get start code. */
    next_start_code(vid_stream);
    show_bits32(data);

      /*
       * If start code is outside range of slice start codes,
       * frame is complete, display frame.
       */
      if (((data < SLICE_MIN_START_CODE)
           || (data > SLICE_MAX_START_CODE))
          && (data != SEQUENCE_ERROR_CODE)) {
#ifdef ANALYSIS
         EndTime();
         stat_a[0].totsize += bitCountRead() - pictureSizeCount;
         PrintOneStat();
       };

 CollectStats();
#endif

 DoPictureDisplay(vid_stream, xinfo);
      }
      goto done;
    }
  }
```

Figura A.13: Continuação da função `mpegVidRsrc`.

```
/* Check if we just finished a picture
 * on the MB_QUANTUMth macroblock
 */
  if (next_bits(23, 0x00000000, vid_stream)) {
    next_start_code(vid_stream);
    show_bits32(data);
    if ((data < SLICE_MIN_START_CODE)
      || (data > SLICE_MAX_START_CODE))
   {
#ifdef ANALYSIS
 EndTime();
 stat_a[0].totsize += bitCountRead() - pictureSizeCount;

 if (showEachFlag) {
   PrintOneStat();
 };

 CollectStats();
#endif

 DoPictureDisplay(vid_stream, xinfo);
    }
  }

  /* Return pointer to video stream structure. */

  goto done;

error:
  fprintf(stderr, "Error!!!!\n");
  next_start_code(vid_stream);
  goto done;

done:
  return vid_stream;
}
```

Figura A.14: Final da função `mpegVidRsrc`.

O trecho de código ilustrado na Figura A.15 a seguir descreve a função `ParsePictureExtension`.

```
static int ParsePictureExtension(VidStream *vid_stream)
{
  char *string;

  /* Flush header start code. */
  if (next_bits(32,ANDRE_START_CODE, vid_stream))
  {
    flush_bits32;
  }
  else return PARSE_OK;

  string = (char *) get_hypermedia_data(vid_stream);

  vid_stream->picture.ListaHypervideoData =
                           (ListaHVD *) filtraString(string);

  printf("\nA String lida foi: %s (%d)",string,strlen(string));
  puts(".");

  /*incluido para retirar ANDRE_END_CODE 0x1BDL*/
  if (next_bits(32,ANDRE_END_CODE, vid_stream))
  {
    flush_bits32;
  }

  return PARSE_OK;
}
```

Figura A.15: Função `ParsePictureExtension`.

Os trechos de código apresenados nas figuras a seguir (A.16 a A.20) descrevem a função `ControlBar`.

```
void ControlBar(VidStream **vid_stream,
                XInfo *xinfo,
                int numMovies)
{
  GC gc;
  int gcflag, winNum;
  Window oldbutton, newbutton;
  XEvent event;
  static int LastState = CTRL_UNDEFINED;
  Display *display=xinfo[0].display;

  gcflag = 0;

  coordenadas coord;//linha inclusa;

  /* Check to see if ControlState was modified outside     */
  /* this function,  and update control displays if it was.*/
  if (LastState != ControlState) {
    if (!gcflag) {
      gc = CreateCtrlGC(display);
      gcflag = 1;
    }
    if ((oldbutton = GetStateButton(LastState)) != (Window)NULL)
     {
      XClearWindow(display, oldbutton);
      DrawButton(oldbutton, gc, display, vid_stream, numMovies);
      if (LastState == CTRL_EOF) {
        XClearWindow(display, playwin);
        DrawButton(playwin, gc, display, vid_stream, numMovies);
      }
    }
    DrawButton(GetStateButton(LastState = ControlState), gc,
        display, vid_stream, numMovies);
  }
```

Figura A.16: Início da função `ControlBar`.

```
  /* Process events, if any */
  if (XPending(display) < 1) { /* No events */
    if (gcflag) {
      XFreeGC(display, gc);
    }
    LastState = ControlState;
    return;
  }
  if (!gcflag) {
    gc=CreateCtrlGC(display);
    gcflag=1;
  }
  do {
    XNextEvent(display, &event);
/* Define if needed; Some older X's don't have this */
#ifdef HAVE_XFILTEREVENT
    if (XFilterEvent(&event, ctrlwindow)) continue;
#endif
    switch(event.type) {
      case ButtonPress:
        /* Toggle Buttons */
        if (event.xbutton.window == loopwin) {
          if (loopFlag) {   XClearWindow(display,loopwin);
                                loopFlag = 0;
          }
          else {
           loopFlag = 1;
           if (ControlState == CTRL_EOF
                  && ControlMotion == CTRLMOTION_ON)
          {   ControlState = CTRL_REWIND;
              DrawRewind(gc,display);
              XClearWindow(display, playwin);
              DrawPlay(gc,display);
              XClearWindow(display, pausewin);
              DrawPause(gc,display);
           }
          }
          DrawLoop(gc,display);
          break;
        }
```

Figura A.17: Continuação da função `ControlBar`.

```
 /* Click in display window */
else if (
      WindowSearch(xinfo,event.xbutton.window,numMovies) != -1)
      {
       puts("coordenadas:");
       printf("\n\tX: %d Y:%d\n",
                   event.xbutton.x,event.xbutton.y);
       puts(".");
       coord.x=(unsigned int) event.xbutton.x;
       coord.y=(unsigned int) event.xbutton.y;
     /** A funcao browser dispara um thread com o browser na
         url passada como parametro.*/
     if (browser(&coord,
         (**vid_stream).picture.ListaHypervideoData))
       printf("Coordenadas dentro\n");
     else
       printf("Coordenadas fora\n");

      if (ControlShow) {  ControlShow = CTRLBAR_OFF;
         } else {   ControlShow = CTRLBAR_ON;      }

         ShowHideControls(&xinfo[0]);
         break;
       }
       /* ControlState buttons --- */
       /* Get currently selected button */
       oldbutton = GetStateButton(ControlState);
       /* Update state */
       if (event.xbutton.window == pausewin)  {
         if ((ControlState == CTRL_EOF
            || ControlState == CTRL_FFWD)
            && ControlMotion == CTRLMOTION_ON) {
           ControlMotion = CTRLMOTION_OFF;
           XClearWindow(display, playwin);
           DrawPlay(gc,display);
         }
         else if (ControlState == CTRL_PLAY) {
           ControlMotion = CTRLMOTION_OFF;
           ControlState = CTRL_PAUSE;
         }
```

Figura A.18: Continuação da função `ControlBar`.

```
    else if (ControlState == CTRL_PAUSE) {
      ControlMotion = CTRLMOTION_ON;
      ControlState = CTRL_PLAY;
    }
  }

  else if (event.xbutton.window == stepwin) {
    if (ControlState == CTRL_PAUSE
     || ControlState == CTRL_PLAY)
      ControlState = CTRL_STEP;
    else if (ControlState == CTRL_EOF && loopFlag)
      ControlState = CTRL_REWIND;
    ControlMotion = CTRLMOTION_OFF;
  }
  else if (event.xbutton.window == playwin) {
    ControlMotion = CTRLMOTION_ON;
    if (ControlState == CTRL_EOF) {
      if (loopFlag) {
        ControlState = CTRL_REWIND;
       }
      DrawButton(playwin,gc,display,vid_stream,numMovies);
    }
    else if (ControlState == CTRL_PAUSE) {
       ControlState = CTRL_PLAY;
    }
  }
  else if (event.xbutton.window == rewindwin) {
    ControlState = CTRL_REWIND;
   }
  else if (event.xbutton.window == exitwin) {
    ControlState = CTRL_EXIT;
  }
  /* Get newly selected button */
  newbutton = GetStateButton(ControlState);
  if (LastState == ControlState) break;
  /* Adjust stopwatch */
  if (LastState == CTRL_PLAY)
    StopWatch(STOPWATCH_STOP);  /* Stop playing */
  else if (ControlState == CTRL_PLAY)
    StopWatch(STOPWATCH_START); /* Start playing */
```

Figura A.19: Continuação da função `ControlBar`.

```
     /* Update button display */
      if (oldbutton != (Window)NULL) {
        XClearWindow(display,oldbutton);
        DrawButton(oldbutton,gc,display,vid_stream,numMovies);
      }
      DrawButton(newbutton,gc,display,vid_stream,numMovies);
        break;
      case ClientMessage:
        if (event.xclient.message_type != protocol_atom ||
            event.xclient.data.l[0] != delete_atom      ||
            event.xclient.format != 32)
               break; /* Not WM_DELETE_WINDOW */
        /* Otherwise drop through to DestroyNotify */
      case DestroyNotify:
        ControlState=CTRL_EXIT;
        break;
      case Expose:
        if (event.xexpose.count > 0)
          break; /* Wait for last expose event */
        if ((winNum=
             WindowSearch(
                          xinfo, event.xexpose.window,
                          numMovies)) != -1)
        {
          if ( vid_stream[winNum]->current != NULL) {
          ExecuteDisplay(vid_stream[winNum],0,&xinfo[winNum]);
        }
      }
      else {
        DrawButton(event.xexpose.window,gc,
                         display,vid_stream,numMovies);
        }
        break;
      default:  break;
    }
  } while (XPending(display) > 0);
  if (gcflag)
    XFreeGC(display, gc);
  LastState = ControlState;
}
```

Figura A.20: Final da função `ControlBar`.

Os trechos de código mostrados nas figuras A.21 e A.22 a seguir descrevem a função `browser`.

```
int browser(coordenadas *coordClick, ListaHVD *lista)
{ clock_t t1,t2;
  register int retorno;  int pid;  ListaHVD *aux;

  if (lista == NULL) return 0;

  /*Verificar se as coordenadas estao inscritas em algum
    hiperlink da lista*/
   aux = lista;

  printf("Coordenadas do Click: %d %d\n",
                (int)coordClick->x, (int)coordClick->y);

  while (aux != NULL)
   if(COORDENADASINCLUSAS(aux->hvd->inicial.x,
     aux->hvd->inicial.y,
     coordClick->x, coordClick->y,
     aux->hvd->final.x, aux->hvd->final.y))
   {
     printf("As coordenadas do click (%d,%d) estao inclusas em
           (%d,%d) e (%d,%d)\n",coordClick->x,coordClick->y,
            aux->hvd->inicial.x,aux->hvd->inicial.y,
            aux->hvd->final.x,aux->hvd->final.y);
     break;
   }
   else
      aux = aux->prox;

  if (aux == NULL) return 0;
```

Figura A.21: Início da função `browser`.

```
  pid = fork();
  if (pid<0)
  {
    fprintf(stderr,"impossivel fazer o FORK");
    exit(1);
  }

  if (pid!=0) return 1; /* se for o processo pai,
                         * retorne imediatamente.
                         */
  retorno = disparaBrowser(aux->hvd->link /*URL*/ );

  if (!retorno)
  {
    printf("OK");
  }
  else
  {
    fprintf(stderr,"Impossiel disparar Browser! Retorno: %d\n",
                retorno);
    exit(1);
  }

  exit(0);
}
```

Figura A.22: Final da função `browser`.