Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

ENCRIPTAÇÃO E COMPRESSÃO DE VÍDEOS DIGITAIS BASEADAS EM PERMUTAÇÃO

Caio César Sabino Silva

TRABALHO DE GRADUAÇÃO

Recife
28/06/2012

Universidade Federal de Pernambuco
Centro de Informática

Caio César Sabino Silva

ENCRIPTAÇÃO E COMPRESSÃO DE VÍDEOS DIGITAIS BASEADAS EM PERMUTAÇÃO

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Tsang Ing Ren*

Recife
28/06/2012

Dedico este trabalho a meus pais.

AGRADECIMENTOS

Agradeço primeiramente à minha família, sobretudo a meus pais, *José Antônio Sobrinho* e *Valdeci Sabino Silva*, por todo amor, apoio e compreensão que me foram dados e me motivaram a superar as dificuldades durante a graduação.

Durante os últimos quatro anos e meio de graduação, algumas pessoas estiveram ao meu lado fazendo projetos de disciplina, seminários e participando de outras atividades acadêmicas. Expresso minha profunda gratidão a todos amigos e colegas de curso que me acompanharam nessa jornada, especialmente a *Amora Albuquerque*, *Anália Lima*, *Ícaro Valgueiro*, *Ivson Diniz*, *Laís Sousa* e *Ruan Carvalho*. Agradeço também à amiga *Lorena Araújo*, que embora não esteja mais no curso, sempre esteve bastante próxima durante toda a graduação.

Expresso minha consideração por todas as pessoas que colaboraram direta ou indiretamente para meu crescimento profissional na área de Ciência da Computação. Em especial a *Mozart Araújo* por todo aprendizado no estágio na Sucinta.

A motivação desse trabalho surgiu durante um projeto em dupla na disciplina de Processamento de Imagens. Agradeço a *Laís Sousa* pela colaboração nas ideias que inspiraram esse trabalho de graduação.

Reconheço também a contribuição de professores e funcionários do *Centro de Informática* para minha formação acadêmica. Em especial, agradeço ao professor *Tsang Ing Ren*, por sua orientação neste trabalho.

Rather than love, than money, than fame, give me truth.

—HENRY DAVID THOREAU (Walden)

RESUMO

No contexto de segurança de aplicações multimídia, a encriptação de vídeos digitais surgiu com o propósito de garantir o sigilo das informações contidas nesse tipo de mídia. Um esquema criptográfico existente na literatura se baseia na utilização de transformações de permutação e aplica a encriptação antes do passo de codificação de vídeo de maneira a preservar ou aumentar a correlação espacial dos quadros do vídeo. A técnica possui um desempenho muito bom, em termos de compressão e qualidade de vídeo, com sequências de vídeo de pouco movimento. Entretanto, a possível melhora na correlação espacial dos quadros depende da correlação temporal de quadros consecutivos. Por isso, em vídeos com considerável movimento, a performance da técnica cai significativamente. O objetivo deste trabalho é propor modificações na computação de permutação e um mecanismo de explorar a correlação temporal do vídeo mais adequadamente com algoritmos de estimativa de movimento para obter um desempenho melhor, em termos de tempo de processamento, taxa de compressão e qualidade de vídeo, sobretudo em vídeos com grande quantidade de movimento.

Palavras-chave: encriptação de vídeo, codificação de vídeo, segurança de dados multimídia, correlação espacial, estimativa de movimento, compensação de movimento

ABSTRACT

In the security context of multimedia applications, digital video encryption intends to assure the confidentiality of the information contained in this type of media. An existing cryptographic system is based on permutation transformations and applies the encryption prior to the coding stages in order to preserve or improve the spatial correlation of the video frames. The technique has a very good performance, in compression rate and video quality, with low motion video sequences. However, the possible improvement on the spatial correlation depends on the temporal correlation between consecutive frames. Hence, in videos with considerable motion, the technique's performance decreases significantly. The objective of this work is to propose modifications on the permutation computation and a mechanism to explore the video temporal correlation more properly by using motion estimation algorithms to obtain a better performance, in terms of processing time, compression rate and video quality, especially in high motion video sequences.

Keywords: video encryption, video coding, multimedia data security, spatial correlation, motion estimation, motion compensation

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

CAPÍTULO 1

INTRODUÇÃO

Este capítulo introduz o contexto da pesquisa desenvolvida neste trabalho de graduação. Além disso, uma visão geral do escopo delimitado por este projeto será mostrada, bem como o objetivo específico deste trabalho.

1.1 CONTEXTO

Criptografia é definida como o estudo de "técnicas matemáticas relacionadas a aspectos de segurança de informações, tais como confidencialidade, integridade de dados, autenticação de entidade e autenticação de origem de dados" [1]. Sendo assim, é um conhecimento usado para garantir o sigilo de informações que devem ser reconhecíveis apenas por indivíduos autorizados. Isso é feito através da aplicação de transformações nos dados para uma forma irreconhecível em que somente usuários autorizados possam extrair a informação original. O processo que torna os dados ilegíveis é chamado de *Encriptação*, enquanto o processo reverso de obter o dado original é *Decriptação*.

No contexto de aplicações multimídias comerciais, devido à abertura de redes públicas, a garantia de segurança dos dados se tornou uma questão importante [2], principalmente no que se refere ao sigilo das informações dos seus usuários. Em razão disso, surgiu a necessidade de aplicação de técnicas de criptografia nessa área.

Entretanto, os dados que são tratados por tais aplicações têm alguns requisitos específicos e, portanto, necessitam de adaptações de esquemas criptográficos convencionais, como AES. No contexto de vídeos digitais, entre tais requisitos [3], podem ser citados:

- **Conformidade com os *padrões de codec* e *formato de vídeo***: O vídeo encriptado deve preservar o formato de compressão de vídeo e decodificadores padrões devem poder decodificar sem gerar qualquer erro. Ou seja, nenhuma mudança deve ser feita aos módulos de codificação de decodificação e o algoritmo deve operar abstraindo como esses módulos funcionam.

- **Controle de *qualidade de percepção***: A qualidade de percepção da informação original do vídeo deve ser reduzida no vídeo encriptado. Geralmente, a encriptação possui um mecanismo de controle de qualidade de percepção. Em algumas aplicações, o requisito de qualidade de percepção permite que o vídeo encriptado seja visualmente reconhecível, mas numa qualidade de percepção bastante inferior à do vídeo original.

- **Velocidade de processamento**: Diversas aplicações de vídeo, como videoconferência, streaming de vídeo, são em tempo real e exigem um algoritmo de encriptação que seja rápido o bastante para ser aplicado nessas condições. O algoritmo deve ser capaz de lidar com vídeos bastante grandes.

- **Compressão de vídeo**: É bastante desejável que o vídeo encriptado tenha um tamanho similar ao vídeo original quando ambos são processados pelo mesmo módulo codificador nas mesmas configurações.

Alguns algoritmos de encriptação de vídeo são específicos de codec [4, 5], enquanto outros são independentes do codec utilizado. Embora os específicos permitam otimizações que podem aumentar a compressibilidade do vídeo, eles representam uma limitação no tipo de vídeo utilizado pela aplicação. A maioria dos algoritmos independentes de codec aplicam transformações nos quadros a serem codificados pelos módulos codificadores, funcionando de maneira independente dos mesmos.

A área de pesquisa em encriptação de vídeos digitais está basicamente dividida em duas metodologias. A *Encriptação Seletiva* aplica o algoritmo de encriptação em partes específicas do fluxo de bits do vídeo. Spanos e Maples [6], por exemplo, propuseram um algoritmo dessa metodologia que encripta apenas os quadros do tipo I de cada grupo de quadros do MPEG. Entretanto, algoritmos dessa metodologia necessitam de mudanças nos módulos de codificação ou decodificação e, por isso, o vídeo não pode ser decodificado por codecs padrões, violando um dos requisitos importantes de aplicações de vídeo mencionados anteriormente.

A segunda metodologia é a *Encriptação Completa*, onde é aplicada em todo o fluxo de bits. Como o tamanho do fluxo de bits é muito grande, algum algoritmo de encriptação diferenciado é aplicado por questões de desempenho, evitando-se os convencionais, como AES. Existem abordagens baseadas em mapas caóticos [7] que possuem alta performance. Entretanto, como esses algoritmos não convencionais geralmente são simples, vários deles são mostrados inseguros contra alguns ataques. Além disso, diversas técnicas atuam após a codificação do vídeo, de maneira que o vídeo resultante não é decodificável por decodificadores padrões.

Para se adequar aos requisitos mencionados anteriormente, como conformidade com os padrões de codec e formato de vídeo, algumas técnicas executam os passos de encriptação antes da codificação. Essas abordagens podem usar a estratégia de encriptar o vídeo de maneira que a codificação tenha alta taxa de compressão. Entre tais abordagens, o método de encriptação [3] estudado por este trabalho é baseado em *transformações de permutação* e explora amplamente a dualidade de permutações em encriptação e compressão de dados.

1.2 OBJETIVO

Esse esquema criptográfico baseado em permutações faz uso da correlação espacial existente nos quadros do vídeo para aumentar a compressibilidade do dado encriptado. Entretanto, tal esquema é voltado para explorar apenas correlação espacial, assumindo que quadros consecutivos são bastante similares, não sendo viável para vídeos com cenas movimentadas.

Este trabalho tem como objetivo estudar extensões para o método de encriptação de vídeos [3], proposto por Socek et al., de maneira a melhorar tempo de processamento, taxa de compressão e qualidade de vídeo. Uma das ideias para aprimorar a compressibilidade do esquema é combiná-lo com técnicas de estimativa e compensação de movimento, que

aumentam a correlação temporal de quadros consecutivos, de maneira similar à forma como é feita em codecs de vídeo avançados, como o MPEG-4.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em sete capítulos. O primeiro apresenta o contexto, motivação e objetivo desta proposta. No segundo capítulo, conceitos básicos de encriptação, compressão de vídeos digitais são apresentados. O terceiro capítulo representa uma visão geral do método estudado por este trabalho. No quarto capítulo, uma análise do esquema, em termos de tempo de processamento, segurança, é detalhada. No quinto capítulo, as modificações e extensões são propostas. O sexto capítulo descreve os experimentos realizados e os resultados obtidos na avaliação das mudanças propostas. Por fim, no sétimo capítulo, as conclusões sobre este trabalho são apresentadas, bem como possíveis trabalhos futuros.

CAPÍTULO 2

CONCEITOS BÁSICOS DE PROCESSAMENTO DE VÍDEOS DIGITAIS

Este capítulo apresenta alguns conceitos básicos da área de processamento de vídeos digitais, com enfoque maior na área de codificação e encriptação de dados.

2.1 VÍDEOS DIGITAIS

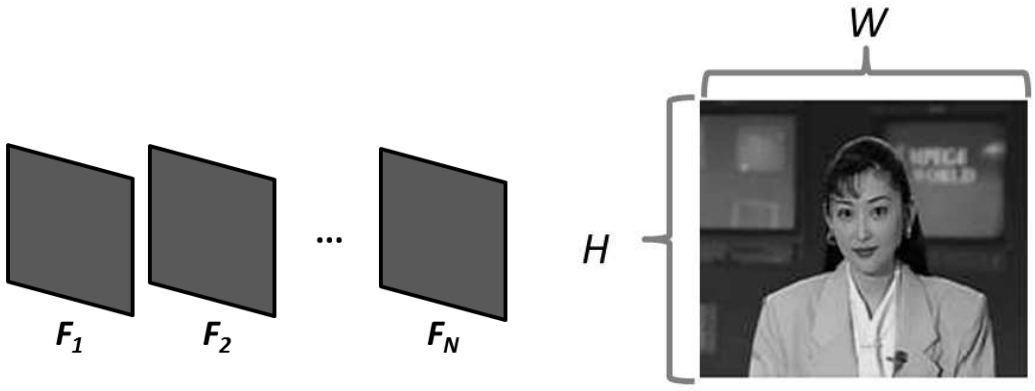
Um video digital pode ser entendido como uma sequência de quadros (*frames*), que podem ser considerados como imagens digitais. Como definido em [8], uma imagem é uma função bidimensional $f(x, y)$, na qual x e y são as coordenadas espaciais e $f(x, y)$ é a *intensidade* ou *nível de cinza* no ponto identificado pelas coordenadas x e y. No caso de uma imagem digital, ocorre que tanto x, y e $f(x, y)$ são valores discretos. O termo *pixel* será utilizado para representar os elementos de uma imagem digital e é equivalente a um ponto no espaço definido pela função da imagem.

Essa definição está associada ao caso de imagens em nível de cinza, entretanto é facilmente extensível para capturar conceitos de cor, bastando apenas que $f(x, y)$ seja visto como uma tupla cujos valores estão associados a coordenadas do espaço do modelo de cores utilizado. No caso do modelo RGB, por exemplo, cada valor de $f(x, y)$ seria uma tupla (R, G, B).

O *histograma* de uma imagem é a distribuição das intensidades dos pixels na mesma. Um histograma $h(x)$ de uma imagem pode ser visto como uma função sobre o nível de cinza x, tal que $h(x)$ representa o número de pixels de intensidade x na imagem.

A notação utilizada neste trabalho, sobretudo para descrever os algoritmos, será a seguinte (como ilustrada na Figura 2.1):

- O número total de quadros de um vídeo será representado pela variável N.
- As variáveis W e H representam, respectivamente, a largura e altura de cada quadro do vídeo. Os parâmetros W e H são iguais em todos os quadros de um mesmo vídeo.
- O i-ésimo quadro será representado F_i. Os quadros são indexados iniciando em 1 e terminando em N.
- Um pixel nas coordenadas (x, y) num quadro F_i será identificado por $F_i(x, y)$. O índice x está no intervalo $[0, W - 1]$, enquanto y está em $[0, H - 1]$.
- A intensidade de um pixel varia de 0 a I_{max}, de acordo com a resolução de intensidade utilizada. Na prática, se n bits são usados, $I_{max} = 2^n - 1$.



(a) (b)

Figura 2.1: Vídeo: (a) sequência de quadros, (b) quadro de vídeo

2.2 CODIFICAÇÃO E DECODIFICAÇÃO

Codec é um módulo de processamento de sinais responsável pela codificação e decodificação dos mesmos. É um termo utilizado na área de vídeos, imagens e sons. Dois tipos básicos de codec existem:

- **Sem perdas (*lossless*)**: Garante que o sinal codificado poderá ser decodificado com conteúdo idêntico ao original. Em geral, resulta numa taxa de compressão menor, uma vez que não pode descartar nenhuma informação no sinal original.

- **Com perdas (*lossy*)**: Permite que, no processo de codificação, haja perda de informações do sinal. Baseia-se na ideia de que alguns dados do sinal podem ser descartados sem afetar a percepção do conteúdo do sinal. Por exemplo, em áudio, algumas frequências são muito pouco perceptíveis à audição humana e portanto, se removidas, em geral, pouco prejudicará a compreensão do som. A maioria dos codecs desse tipo fornece um mecanismo de controle de qualidade de modo a reduzir ou aumentar a quantidade de dados que pode ser perdida para tornar o sinal mais compressível.

2.2.1 Correlação em vídeos digitais

Na codificação de dados, é importante poder representar a informação desejada com poucas unidades de informação. Existem informações que podem ser codificadas com menos bits do que outras. Por exemplo, um vídeo com todos os quadros idênticos é mais facilmente codificável em poucas unidades de informação do que um vídeo com quadros bastante distintos entre si.

Para formalizar a ideia de "complexidade" da informação a ser codificada, precisa-se de uma métrica que possa quantificar a informação de um sinal. Tendo em vista esse propósito, Shannon introduziu o conceito de *entropia* na área de informação [9], embora originalmente tenha sido proposto para a área de comunicação.

(a) (b)

Figura 2.2: Correlação espacial: (a) imagem original, (b) imagem permutada aleatoriamente com correlação espacial destruída

A entropia está associada ao grau de incerteza ou imprevisibilidade da informação, isto é, ao nível de ausência de redundância ou correlação de dados na mesma. Num vídeo, existem quatro tipos de correlação de dados que podem ser exploradas na codificação de vídeos [10, 8]:

- **Redundância de codificação** : é referente ao tamanho médio das palavras utilizadas para codificar os símbolos que ocorrem no dado a ser codificado. É aplicável a diversos tipos de dados, como imagem, som e texto. Por exemplo, se uma imagem em escala de cinza, tendo como conteúdo apenas pixels pretos e brancos, for codificada na resolução de intensidade de 8 bits, não seria necessário usar 8 bits para representar cada intensidade de pixel, já que só dois valores existem na imagem. Um único bit poderia ser usado para representar cada intensidade da imagem, mesmo estando numa resolução de intensidade de 8 bits.

  Codificadores de entropia em geral são capazes de eliminar bastante redundância de codificação. Um exemplo bastante conhecido é a codificação de Huffman [11], que define uma tabela de palavras de código, priorizando as menores palavras para os valores que mais ocorrem no dado.

- **Correlação espacial** : aplica-se a cada imagem separadamente (*intra-frame*) e é advindo da área de compressão de imagens. Também chamada de *redundância entre pixels*, a ideia é que pixels em coordenadas próximas têm grande probabilidade de terem intensidades similares. Isso se deve ao fato de que tais regiões de intensidade similar geram uma forma mais facilmente reconhecível pelo olho humano. Ao observar uma imagem com baixíssima correlação espacial, ela provavelmente será pouco reconhecível pelo olho humano. Isso de fato é explorado por algoritmos de encriptação para degradar a qualidade de percepção da imagem codificada, como mostrado na Figura 2.2.

  O conceito de correlação espacial pode ser usado facilmente por algoritmos de compressão de imagem. Como exemplo, uma sequência de pixels de mesma intensidade

(a) (b) (c)

Figura 2.3: Correlação temporal: (a) quadro anterior, (b) quadro atual, (c) diferença residual entre quadros (a) e (b) (centralizado no tom cinza médio $\frac{I_{max}}{2}$)

pode ser representada utilizando menos bits se for aplicada, por exemplo, a técnica de *Run Length*, isto é, codificar a sequência com apenas dois parâmetros: o valor da intensidade e o número de pixels consecutivos com tal intensidade.

- **Correlação temporal** : é um conceito que envolve mais de um quadro (*inter-frame*). Baseia-se no princípio de que quadros consecutivos tendem a ser bastante similares, pois, caso contrário, causariam uma sensação de movimento descontínuo nas transições de tais quadros. Logo, a transição de dois quadros é, em geral, composta de pequenos movimentos (*motion*), tais como translação de câmera, zoom, objetos na cena se movendo. A Figura 2.3 mostra, como exemplo, dois quadros consecutivos e a diferença residual entre ambos.

- **Redundância psicovisual** : consiste na existência de dados que o olho humano não consegue perceber facilmente, de maneira que, sua remoção faz pouca diferença na percepção de um quadro do vídeo. É mais subjetivo e difícil de estimar adequadamente.

2.2.2 Codificação de quadros

Quanto ao uso de correlação, existem codecs *apenas espaciais* (*spatial-only*) que utilizam apenas os tipos de correlação aplicável em cada imagem separadamente, como é o caso do MPNG e MJPEG, por exemplo. Já codecs mais avançados, como MPEG-4, utilizam também correlação temporal. Dessa forma, o processo de codificação de vídeos permite a existência de três tipos de quadros [12]:

- **Tipo I (*I-frame*)**: cuja compressão é realizada individualmente. Codecs que não utilizam correlação temporal possuem apenas este tipo de quadro.

- **Tipo P (*P-frame*)**: cuja compressão é realizada, expressando transformações em relação a um quadro anterior.

- **Tipo B (*B-frame*)**: cuja compressão é realizada, expressando transformações em relação a dois quadros: um anterior e outro posterior.

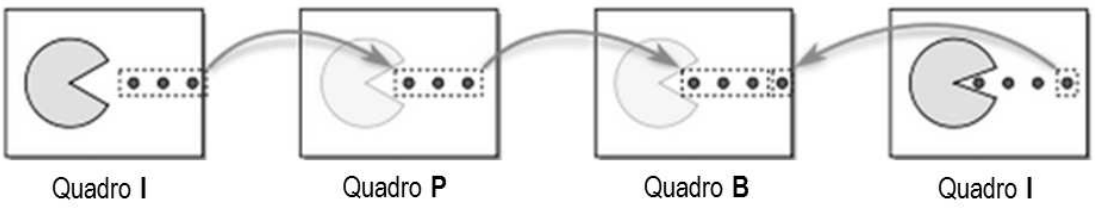


Figura 2.4: Exemplo de GOP com os três tipos de quadros

Em relação a quadros do tipo P e B, as transformações mencionadas podem ser vistas como o alinhamento dos quadros (compensar pelos movimentos que ocorreram na transição dos mesmos) e a codificação da diferença entre eles. Para realizar um bom alinhamento, técnicas de estimativa de parâmetros de movimento (*motion estimation*) e de compensação de movimento (*motion compensation*) podem ser utilizadas. Nesse contexto, os quadros usados para computar as transformações são chamados de *quadros de referência*. Note que um quadro do tipo B tem dois quadros de referência, enquanto um P tem apenas um. Em geral, além do residual em relação aos quadros de referência, vetores de movimento (*motion vectors*), calculados em fase de codificação, precisam estar no fluxo de bits do vídeo, para que na decodificação, possa ser reconstruído corretamente.

Embora quadros do tipo P e B sejam bem mais compressíveis que os do tipo I, eles não são auto-decodificáveis, isto é, necessitam dos quadros de referência para serem computados. Para permitir que se possa reproduzir um vídeo a partir de um ponto qualquer no vídeo, surgiu o conceito de *grupo de imagens* (GOP). Um grupo de imagens representa uma unidade auto-decodificável, de maneira que se saltamos para um quadro qualquer de um certo GOP, apenas quadros do mesmo grupo precisam ser buscados para poder reconstruir a partir do trecho escolhido. Para isso, o primeiro quadro de cada grupo de quadros é necessariamente do tipo I. Um exemplo de grupo de imagens com os três tipos de quadro pode ser visto na Figura 2.4.

O esquema básico de um codificador pode ser visto na Figura 2.5. O modelo temporal é responsável por guardar imagens previamente codificadas num *buffer*, realizar a estimativa de movimento, computar os vetores de movimento e calcular a diferença residual entre o quadro atual e o de referência após a compensação de movimento. No caso de um quadro do tipo I, o residual é o próprio quadro. O modelo espacial recebe o residual e utiliza algum algoritmo que é capaz de aproveitar correlação espacial para codificação. Um exemplo seria a transformada do cosseno, que é usada por diversos codificadores e gera como saída uma matriz onde os coeficientes mais significativos se acumulam numa determinada região, permitindo que o dado resultante tenha boa taxa de compressão usando um codificador de entropia comum.

2.3 PERMUTAÇÃO EM VÍDEOS DIGITAIS

Uma permutação de uma sequência s é uma bijeção de s em si mesmo [3], isto é, o mapeamento de um elemento em outro elemento da sequência, sem repetições do valor

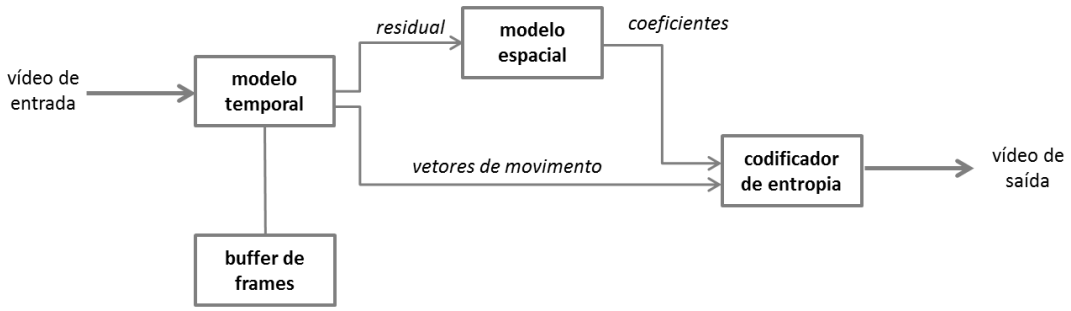


Figura 2.5: Esquema de um codificador de vídeo

mapeado e havendo um único mapeamento para cada elemento. Formalmente, define-se uma permutação P de uma sequência s como uma matriz:

$$P = \left[\begin{array}{cccc} i_0 & i_1 & i_2 & \ldots & i_{N-1} \end{array} \right]$$

onde:

- N é o tamanho da sequência e
- i_j, tal que $0 \leq j < N$, é o índice em que o j-ésimo elemento será mapeado.

A matriz de permutação tem as seguintes propriedades, por ser uma bijeção:

- $\forall x, y \in \{0, ..., N-1\}, x \neq y \rightarrow i_x \neq i_y$ e
- $\forall x \in \{0, ..., N-1\}, \exists y \in \{0, ..., N-1\} | i_y = x$.

Isso é válido para um quadro de vídeo, mesmo sendo bidimensional, bastando apenas representá-lo como uma sequência onde a imagem é percorrida linha por linha. A notação $P(F)$ será usada neste trabalho para representar a aplicação da permutação P no quadro F. A aplicação de permutação num quadro consiste em mandar o pixel de posição j para a de índice i_j da matriz de permutação P.

Permutações têm sido muito usadas em diversas técnicas de encriptação e compressão ao longo dos anos. Transformações baseadas em permutação são a base fundamental da criptografia moderna de chave simétrica, como ocorre em sistemas como AES (*Advanced Encryption System*) ou DES (*Data Encryption Standard*).

Transformações baseadas em permutação são também primitivas usadas em técnicas de compressão. A *Transformada de Burrows-Wheeler* (BWT) é um exemplo de tal transformação. Ela opera modificando blocos de texto, gerando um novo bloco que contém os mesmos caracteres, mas que é mais compressível por um algoritmo de compressão qualquer [13].

No contexto de encriptação de vídeos, há diversas formas de aplicar permutações nos dados de vídeo. Existem técnicas que escondem o conteúdo de todos os quadros, enquanto

(a) (b)

Figura 2.6: Permutação e compressibilidade: (a) imagem original PNG 176x144 22.3KB, (b) imagem '*quase-ordenada*' PNG 176x144 12.5KB

outras aplicam a permutação no fluxo de bits do vídeo. Permutações geradas por uma chave secreta podem ser usadas para embaralhar os pixels ou blocos de pixels em um quadro, num algoritmo de permutação pura.

Entre abordagens existentes, uma delas [14] consiste em embaralhar os coeficientes aleatoriamente da transformada discreta do cosseno de um vídeo codificado com MPEG. Outra estratégia conhecida [15] é aplicar a permutação nas palavras de código da tabela utilizada na codificação de Huffman.

Aplicar permutação nos pixels de uma imagem afeta a correlação espacial nas vizinhanças dos pixels permutados. Esse fato pode ser usado na encriptação de vídeo para destruir a correlação espacial, usando uma permutação aleatória, e tornar o vídeo ilegível para usuários não autenticados. Entretanto, isso prejudica drasticamente a compressibilidade da imagem. Por outro lado, se uma permutação de ordenação for usada, pixels com intensidades similares que estavam distantes na imagem original estarão próximos na imagem encriptada e a imagem resultante será bastante compressível.

De acordo com [3], "quadros ordenados, assim como '*quase-ordenados*', possuem forte correlação espacial"e, por isso, podem ser até mais compressíveis que o quadro original quando usando um codec somente espacial, como pode ser visto na Figura 2.6. O conceito de 'quase-ordenação' introduzido pelos autores refere-se à aplicação da permutação de ordenação do quadro anterior a um dado quadro. Para isso, é assumido que dois quadros consecutivos são bastante similares, de maneira que a aplicação de tal permutação resulta numa imagem quase totalmente ordenada com alguns pequenos ruídos.

CAPÍTULO 3

MÉTODO DE ENCRIPTAÇÃO DE VÍDEO BASEADO EM PERMUTAÇÕES

Este capítulo apresenta um estudo sobre o método de encriptação baseado em permutações, proposto por Socek et al. [3]. O propósito do esquema criptográfico é a transmissão de um vídeo de maneira a evitar que os dados transmitidos possam ser escutados e compreendidos por um usuário não autenticado.

3.1 ESQUEMA CRIPTOGRÁFICO

O esquema criptográfico assume a existência de dois canais de comunicação. O primeiro canal, chamado *ChS*, é um canal seguro, onde os dados trafegam usando algum protocolo de comunicação seguro, geralmente usando algum algoritmo de encriptação convencional. No segundo canal, *ChR*, os dados trafegam livremente, sem nenhuma proteção ou protocolo de segurança.

Ambos canais podem ser escutados por um usuário intruso, mas como o seguro utiliza um algoritmo de encriptação convencional, tem maior segurança que o canal comum. Por outro lado, o canal seguro, por usar um protocolo de segurança, implica num fluxo de bits maior (overhead do protocolo) a ser transmitido, o que pode prejudicar a taxa de transmissão do vídeo, podendo se tornar inviável para aplicações em tempo real, se o canal for usado para transmissão de muitos dados.

O esquema criptográfico utiliza um codificador somente espacial, abstraindo o seu funcionamento e como os dados são representados no fluxo de bits, sendo relevante apenas que o mesmo possua duas funções básicas: codificar e decodificar. Sendo assim, serão usadas na notação desse trabalho a expressão $E(F)$, para representar o quadro de saída F' da codificação do quadro F, e a expressão $D(F)$ para representar o quadro decodificado a partir de sua representação no fluxo de bits. Note que $D(E(F)) = F$ no caso de codificadores sem perdas.

Além disso, a encriptação ocorrerá como um passo antes da codificação, de maneira que o vídeo encriptado preserve ou melhore a correlação espacial dos pixels. Sendo assim, o esquema é como mostrado na Figura 3.1.

A chave criptográfica usada no sistema, que também define se o usuário é ou não autenticado, consiste no primeiro quadro do vídeo. Para isso, ele é transmitido inicialmente pelo canal seguro *ChS*. Uma vez transmitido, todos os próximos quadros serão transmitidos pelo canal comum *ChR* e serão encriptados usando a permutação de ordenação única do quadro anterior, isto é, o segundo quadro com a permutação do primeiro, o terceiro com a do segundo e assim por diante.

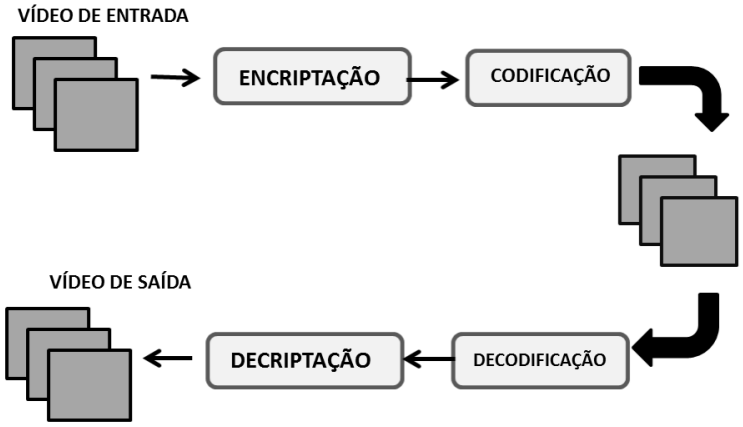


Figura 3.1: Visão geral do esquema criptográfico

3.1.1 Permutação de ordenação única

O esquema propõe a computação de uma permutação de ordenação como forma de encriptar os quadros do vídeo. Uma restrição de tal permutação é que ela deve ser única, isto é, sempre resulta na mesma permutação para um mesmo quadro. A permutação de ordenação é feita em relação às intensidades dos pixels da imagem e pode ser computada com uma variação simples do *quicksort*, como o Algoritmo 1 proposto pelos autores.

Este procedimento recebe uma cópia do quadro F, que será ordenado na computação da permutação, uma permutação inicial, que é a identidade, e dois inteiros que representam os extremos da imagem. Vale ressaltar que, no algoritmo, o quadro F e a permutação P são vistos como dados unidimensionais de tamanho $W \times H$.

Dessa maneira, a permutação de ordenação, quando aplicada ao próprio quadro, resultará numa imagem em forma de gradiente, que tem correlação espacial maior ou igual à original. A aplicação da permutação do quadro anterior, ao invés da do próprio quadro, tende a resultar num quadro 'quase-ordenado', uma vez que quadros consecutivos são em geral bastante similares, logo terá bom desempenho quando usada pelo módulo codificador.

3.1.2 Algoritmo de encriptação

Na notação usada a seguir, seja P uma permutação dos pixels calculada, define-se P^{-1} como sendo a permutação inversa a P, isto é, tal que $P^{-1}(P(F)) = F$. Isso significa que a permutação inversa "desfaz"as trocas de pixels realizadas por P na imagem.

No Algoritmo 2, é possível ver o pseudocódigo do método de encriptação. O pseudocódigo é aplicável tanto a codecs com perdas como sem perdas. Note que, no caso de um codificador com perdas, o encriptador precisa calcular a permutação sobre os quadros que serão decifrados pelo decriptador, como no pseudocódigo. Entretanto, no caso de codecs

Entrada: Quadro F, Permutação P, Inteiros *esquerda* e *direita*

início
  Inicialize $i \leftarrow esquerda - 1$, $j \leftarrow direita$, $v \leftarrow F[esquerda]$

  se $direita \leq esquerda$ **então**
    Retorne do procedimento
  fim

  repita
    $i \leftarrow i + 1$
    enquanto $F[i] < v$ **faça**
      $i \leftarrow i + 1$
    fim

    $j \leftarrow j - 1$
    enquanto $j > esquerda$ *e* $F[j] > v$ **faça**
      $j \leftarrow j - 1$
    fim

    se $i < j$ **então**
      Troque $F[i]$ por $F[j]$
      Troque $P[i]$ por $P[j]$
    fim
  até $i \geq j$

  Recursão com $F = F, P = P, esquerda = esquerda$ e $direita = i - 1$
  Recursão com $F = F, P = P, esquerda = i + 1$ e $direita = direita$
fim

Algoritmo 1: Variação do *quicksort* para cálculo da permutação de ordenação

Entrada: Vídeo $F_1, ..., F_N$
início
  Obtenha o quadro $F_1^{transmitido} = E(F_i)$
  Obtenha o quadro $F_1^{decriptado} = D(F_1^{transmitido})$
  Compute a permutação P_1 do quadro $F_1^{decriptado}$
  Transmita $F_1^{transmitido}$ pelo canal ChS

  para cada F_i, *onde* $i = 2, ..., N$ **faça**
    Obtenha o quadro $F_i^{transmitido} = E(P_{i-1}(F_i))$
    Obtenha o quadro $F_i^{decriptado} = P_{i-1}^{-1}(D(F_i^{transmitido}))$
    Transmita $F_i^{transmitido}$ pelo canal ChR

    se $i < N$ **então**
      Compute a permutação P_i do quadro $F_i^{decriptado}$
    fim
  fim
fim

Algoritmo 2: Algoritmo de encriptação de vídeo

sem perdas, pode-se calcular a permutação diretamente sobre os quadros de entrada.

3.1.3 Algoritmo de decriptação

O processo de decriptação é mais simples e pode ser visto no Algoritmo 3. Esse algoritmo é válido para codecs com ou sem perdas, desde que o vídeo tenha sido encriptado com o algoritmo correto.

Observe que o algoritmo de decriptação recebe como entrada os quadros codificados (e permutados, com exceção do primeiro) no processo de encriptação, portanto os quadros que são lidos nesse procedimento correspondem a $D(E(F_1))$ para o primeiro quadro recebido do canal ChS ou $D(E(P_{i-1}(F_i)))$, para qualquer quadro F_i seguinte ($i > 1$) vindo do canal ChR. Verificando tanto o algoritmo de encriptação como o de decriptação, nota-se a igualdade das expressões obtidas para P_i calculadas em ambos procedimentos, o que assegura que o esquema funciona corretamente.

3.2 EXTENSÕES PARA OS ALGORITMOS BÁSICOS

Algumas questões foram levantadas pelos autores e os mesmos propuseram algumas extensões com o intuito de lidar com tais situações especiais do algoritmo.

3.2.1 Controle de qualidade de percepção

Um dos requisitos específicos de aplicações de vídeo mencionados na introdução deste trabalho foi o controle da qualidade de percepção. É possível adaptar o esquema para contemplar esse requisito através da extensão *baseada em blocos*.

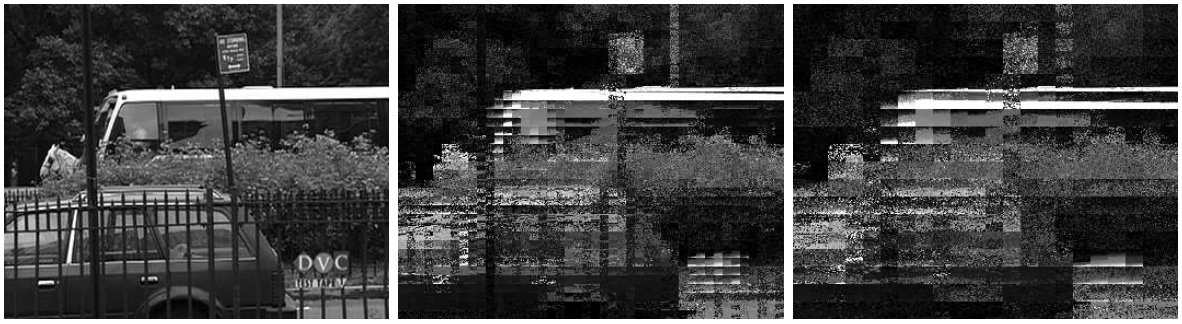
Entrada: Vídeo encriptado $F_1^{transmitido}, F_2^{transmitido}..., F_N^{transmitido}$

início
  Receba $F_1^{transmitido}$ pelo canal ChS
  Obtenha o quadro $F_1^{decriptado} = D(F_1^{transmitido})$
  Compute a permutação P_1 do quadro $F_1^{decriptado}$

  para cada $F_i^{transmitido}$, *onde* $i = 2, ..., N$ **faça**
    Obtenha o quadro $F_i^{decriptado} = P_{i-1}^{-1}(D(F_i^{transmitido}))$

    se $i < N$ **então**
      Compute a permutação P_i do quadro $F_i^{decriptado}$
    fim
  fim
fim

Algoritmo 3: Algoritmo de decriptação de vídeo



(a) (b) (c)

Figura 3.2: Abordagem baseada em blocos: (a) quadro original, (b) e (c) permutação usando abordagem baseada em blocos de tamanho 8x8 e 16x16 respectivamente

O parâmetro que define a qualidade de percepção nessa extensão é o tamanho dos blocos. A ideia da extensão consiste em dividir a imagem em blocos de tamanho fixo e a permutação de ordenação única calculada nos algoritmos seria composta pelas permutações de ordenação de cada bloco separadamente.

À medida que o tamanho do bloco aumenta, a qualidade de percepção do vídeo diminui. Se o tamanho do bloco é 1x1, a permutação será a identidade e a qualidade de percepção é igual à imagem original. Por outro lado, se o tamanho do bloco é tal que só haja um bloco na imagem, a qualidade de percepção é a mesma do algoritmo básico.

Essa extensão não só beneficia o requisito de qualidade de percepção, mas também permite que erros de 'quase-ordenação' ocasionados por movimentos de objetos na cena sejam suavizados, uma vez que diminui o espaço para troca de pixels na permutação de ordenação. Isso pode ser visto na Figura 3.2, onde os erros são mais notáveis quando se usa um bloco de tamanho maior.

3.2.2 Translação de câmera constante

Entrada: Permutação P e parâmetros de translação t_x e t_y
início
  Inicialize $inicio \leftarrow 0$ e $fim \leftarrow W \times H$
  Inicialize P' como uma matriz qualquer de mesmo tamanho que P

  para $0 \leq k < W \times H$ **faça**
    $i \leftarrow t_x + P[k] \mod W$
    $j \leftarrow t_y + \lfloor P[k]/W \rfloor$

    se $0 \leq j < H$ *e* $0 \leq i < W$ **então**
      $P'[inicio] = j \times W + i$
      $inicio \leftarrow inicio + 1$
    senão
      $fim \leftarrow fim - 1$
      $P'[fim] = (j \mod H) \times W + (i \mod W)$
    fim
  fim

  Retorne P'
fim

Algoritmo 4: Algoritmo de ajuste da permutação com translação de câmera constante

O algoritmo básico é bastante sensível à translação global de câmera, uma vez que o erro do uso da permutação do quadro anterior ocorrerá na imagem inteira. Visando tratar essa fragilidade, os autores propuseram uma extensão que trata esse problema.

Na extensão, o transmissor, antes de encriptar um dado quadro, verifica se houve uma translação global de câmera usando algum algoritmo de estimativa de movimento. Após isso, caso haja translação, ele transmite os parâmetros t_x e t_y ao receptor, notificando que na permutação do quadro anterior, deve ser computada uma translação com tais parâmetros.

O Algoritmo 4 mostra como implementar tal translação na permutação de ordenação. Na implementação, os pixels são permutados para o ponto da permutação P original reajustado pela translação. Caso o ponto reajustado esteja fora da imagem, ele é permutado para as últimas linhas da mesma (o que é representado pela variável fim no algoritmo).

3.2.3 Histograma dos quadros do vídeo visível

Um dos problemas de segurança do algoritmo original é que o vídeo encriptado não esconde o histograma dos quadros do vídeo. Isso pode revelar algumas informações sobre a imagem do quadro encriptado, como saber se a imagem original é escura ou clara, facilitando ao atacante reconstruir a imagem original que pode tentar unir pixels de intensidade similar para restaurá-la.

(a) (b) (c)

Figura 3.3: Escondendo o histograma do quadro: (a) quadro original, (b) algoritmo básico, (c) algoritmo básico com extensão de ocultação de histograma

Uma forma de ocultar o histograma, como proposta pelos autores, é, ao invés de codificar o quadro puro permutado, codificar a permutação aplicada à diferença de intensidade dos quadros. Dessa forma, assumindo que o atacante não possui o quadro anterior decifrado, o mesmo não tem como descobrir o histograma do quadro atual, nem sequer descobrir se uma dada intensidade ocorre no quadro ou não.

Para possibilitar a reconstrução correta dos quadros pelo decodificador, os autores definiram a diferença de dois quadros F e G como nas Equações 3.2.1 e 3.2.2.

$$\Delta(F,G)[x,y] = clip\left(F[x,y] - G[x,y] + \left\lfloor \frac{I_{max}}{2} \right\rfloor\right) \tag{3.2.1}$$

$$clip(x) = \begin{cases} I_{max}, & \text{se } x > I_{max} \\ x, & \text{se } 0 \leq x \leq I_{max} \\ 0, & \text{se } x < 0 \end{cases} \tag{3.2.2}$$

A ideia dessa função de diferença é centralizar a diferença de intensidades em $\left\lfloor \frac{I_{max}}{2} \right\rfloor$, de maneira que intensidades abaixo desse valor representam decréscimo de intensidade de F para G, enquanto valores acima são acréscimo de intensidade.

Essa função de corte, entretanto, é apenas válida para o caso de codecs com perdas, pois diferenças entre os quadros F e G acima de $\frac{I_{max}}{2}$ serão reduzidas a $\frac{I_{max}}{2}$. Isso pode impactar na qualidade de codificação do vídeo também.

Além disso, essa extensão influencia bastante na compressibilidade do vídeo. Basta lembrar que, para explorar correlação temporal na codificação de vídeos, a diferença residual é codificada como foi mostrado no capítulo anterior. Essa técnica tem um efeito similar, como pode ser notado na Figura 3.3 pela homogeneidade do quadro encriptado com a extensão.

3.2.4 Perda de quadros

O esquema básico foi construído assumindo que nenhum quadro será perdido na transmissão, uma vez que o processo de decriptação necessita do quadro anterior. Isso limita o uso de aplicações, pois torna o sistema não adequado a aplicações em que perdas de quadro sejam toleráveis, como videoconferência. Além disso, não permite que o vídeo possa ser reproduzido a partir de um ponto qualquer eficientemente, pois o esquema exige a decriptação de todos os quadros anteriores para isso.

A extensão proposta consiste em, aplicar a encriptação de cada GOP separadamente, de maneira que o quadro inicial de cada GOP é transmitido usando o canal *ChS*. Dessa forma, a decriptação, em caso de perda de quadros, apenas seria afetada para os quadros restantes do mesmo GOP. A reprodução a partir de um ponto qualquer poderia ser feita, realizando a decriptação começando no início do GOP desejado e, como o número de quadros de um grupo de imagens é suficientemente pequeno, poucos quadros seriam processados para reconstruir o quadro desejado.

CAPÍTULO 4

ANÁLISE DO ESQUEMA CRIPTOGRÁFICO

Este capítulo faz uma análise do sistema criptográfico, em relação a aspectos como tempo de processamento, espaço utilizado e segurança. Além disso, as limitações e fragilidades do esquema serão explicitadas.

4.1 ANÁLISE DE TEMPO DE PROCESSAMENTO E ESPAÇO UTILIZADO

Uma vez que velocidade de processamento é um dos requisitos específicos de esquema criptográficos para vídeos, nesta seção o tempo de execução desse esquema é estudado. Além disso, o custo de uso de memória necessário para o algoritmo é estimado.

O custo total de processamento em tempo e espaço do algoritmo básico de encriptação/decriptação consiste em:

- computação da permutação de ordenação única,
- computação da permutação inversa e
- aplicação da permutação em um quadro

A computação da permutação de ordenação única mostrada na seção anterior tem complexidade de tempo quadrático, igual a um algoritmo *Quicksort*, sendo assim $O(W^2 \times H^2)$. De fato, a complexidade é ainda menor, pois, conforme mencionado pelos autores, qualquer algoritmo de ordenação baseado em comparação de chaves poderia ser usado de maneira similar para calcular a permutação, como o *Mergesort* de custo $O(WH \times log(WH))$.

Para a extensão usando a abordagem baseada em blocos, a complexidade de tempo em função das dimensões de bloco B_x e B_y pode ser vista como a multiplicação do número de blocos pelo custo de ordenação de cada bloco, como mostrado na Equação 4.1.1.

$$T(W, H, B_x, B_y) = O\left(\left(\left\lceil \frac{W}{B_x} \right\rceil \times \left\lceil \frac{H}{B_y} \right\rceil\right) \times (B_x B_y \times log(B_x B_y))\right) \tag{4.1.1}$$

À medida que o tamanho de blocos tende a 1x1, a complexidade em função de W e H tende a ser linear, entretanto a qualidade de percepção tende a ser idêntica à do video original.

Embora não explicitada pelos autores do esquema, a computação da permutação inversa pode ser feita em tempo linear em $W \times H$ facilmente, requerendo apenas uma cópia de P. A ideia básica do algoritmo seria definir P' iterando na matriz de permutação

P, definindo, para cada índice i, $P'[P[i]] = i$. A complexidade de tempo de uma aplicação de permutação em um quadro também é $O(WH)$.

Sendo assim, o custo de tempo total do algoritmo básico é determinado pela computação da permutação de ordenação. Ainda assim, o custo computacional da técnica é adequado para aplicações em tempo real.

O espaço utilizado pelo algoritmo tem complexidade linear, ou seja, apenas um número constante de cópias de um quadro é necessário para a encriptação e decriptação de um vídeo.

A extensão de ocultação de histograma, por ser computável em tempo linear em $W \times H$ pelo cálculo da diferença dos quadros, não impacta na complexidade de tempo do esquema. Por outro lado, a extensão de translação de câmera constante causa um acréscimo no tempo de processamento dependente do algoritmo de estimativa de parâmetros de translação utilizado.

4.2 ANÁLISE DE SEGURANÇA

Nessa seção, o comportamento do sistema criptográfico sobre alguns tipos de ataques serão estudados. Formalmente, pode-se ver o sistema criptográfico dessa forma:

- a chave do sistema é a permutação de ordenação do quadro anterior,
- o texto puro a ser encriptado corresponde ao vídeo original e
- o cifrotexto é o quadro encriptado (permutado segundo a ordenação do quadro anterior)

4.2.1 Ataque por força bruta

Esse tipo de ataque, também chamado de ataque por busca exaustiva de chave, consiste em testar todas as chaves possíveis até encontrar a correta. Geralmente é usado quando o sistema criptográfico não contém nenhuma fraqueza a ser explorada.

Para mostrar que é seguro contra esse tipo de ataque, em geral, o espaço total de chaves é mostrado, pois no pior caso, um ataque por força bruta testa todo as chaves, sendo computacionalmente inviável. No caso do esquema criptográfico descrito no capítulo anterior, dois pontos podem ser atacados:

- a transmissão do quadro puro pelo canal ChS e
- a transmissão do quadro encriptado pelo canal ChR

Para o primeiro caso, assumindo que um esquema criptográfico convencional é usado, como AES com chaves de n-bits, tem-se que o tamanho do espaço de busca é 2^n. Para o segundo caso, o número de chaves é dado pelo número de permutações de ordenação válidas que geram um dado quadro encriptado. Para calcular o número de chaves, pode-se então notar que, para cada permutação distinta do quadro encriptado, existe uma única permutação de ordenação que transforma o texto puro no cifrotexto. Logo, esse

número coincide com o número de textos puros possíveis para um dado cifrotexto. Seja o quadro encriptado F, de dimensões $W \times H$, com histograma $h(x)$, o número de chaves Ω (quadros permutados distintos) é dado pela Equação 4.2.1.

$$\Omega = \frac{(W \times H)!}{\prod_{x=0}^{I_{max}} h(x)!} \tag{4.2.1}$$

Note que, na prática, usando resoluções de vídeo razoáveis, o número de chaves é bem maior do que o número de chaves do esquema criptográfico convencional. Entretanto, para histogramas bastante "estreitos", isto é, com apenas pouquíssimas intensidades com frequência não nula, esse número pode se tornar muito pequeno.

No pior caso, que seria um quadro com uma única intensidade, haveria apenas um único quadro chave possível (o próprio quadro) cuja permutação de ordenação certamente seria a chave. Sendo assim, para vídeos com histograma extremamente "estreitos", o sistema criptográfico deve ser considerado inseguro contra esse tipo de ataque. Por outro lado, na grande maioria das aplicações de vídeo, o número de permutações é maior do que o de chaves de um esquema criptográfico convencional de 128 bits, por exemplo.

4.2.2 Ataque por texto puro conhecido

Esse tipo de ataque assume que o atacante tem acesso a um texto puro e seu cifrotexto correspondente [16]. A ideia é que o mesmo pode descobrir uma relação entre o texto puro e cifrotexto e deduzir como a encriptação é realizada para quadros desconhecidos (descobrindo a chave utilizada ou parte dela). Por quadro desconhecido, entende-se como um quadro com nenhuma correlação conhecida (se é o quadro anterior, posterior, por exemplo) com o exemplo de texto puro e cifrotexto que o atacante tem acesso.

Métodos de encriptação baseado somente em permutações geradas por uma chave secreta são considerados fracos contra ataques de texto puro escolhido ou conhecido [17]. Um diferencial do uso de permutação dessa técnica, que a torna mais segura em relação às abordagens baseadas em permutação, é que a permutação da imagem (chave do sistema) é gerada pelo quadro a ser encriptado e não por uma chave secreta.

Isso significa que um par de cifrotexto e seu texto puro em si não dá nenhuma informação direta sobre a transformação de permutação usada para outro texto puro, pois as permutações dependem somente do texto puro anterior, que muda a todo momento de maneira imprevisível. Logo, ter acesso a exemplos de quadros e seus encriptados correspondentes não dá nenhuma informação sobre como a permutação será gerada para um quadro desconhecido.

4.2.3 Ataque por texto puro escolhido

Nesse tipo de ataque, é assumido que o atacante pode escolher um conjunto de texto puro e obter o conjunto de cifrotextos correspondentes a cada texto puro [16]. Para mostrar que o esquema é seguro, a mesma ideia do ataque de texto puro conhecido pode ser usada. Embora o atacante possa escolher diversos quadros e seus encriptados correspondentes, ele

não poderá descobrir um quadro desconhecido, pois a permutação depende diretamente do quadro desconhecido e não de uma chave secreta.

4.2.4 Ataque por cifrotexto escolhido

Esse tipo de ataque assume que o atacante pode decriptar um conjunto de cifrotextos à sua escolha[16]. No contexto do esquema, o atacante poderia descobrir a função de decriptação (transformação de permutação inversa) para cada quadro encriptado escolhido. Entretanto, não teria acesso à função de decriptação de quadros desconhecidos, pois também dependem diretamente do dado encriptado.

Por outro lado, embora seja fora do escopo desse tipo de ataque, se é conhecido que o quadro é sequencial a um exemplo que o atacante tenha à disposição, o mesmo consegue obter a função de decriptação do mesmo. Isso é um dos pontos de maior vulnerabilidade do sistema, uma vez que se o atacante descobre um dado quadro, o mesmo tem acesso a qualquer quadro posterior. A extensão de encriptação por GOP ameniza esse problema, pois o atacante só teria acesso aos próximos quadros do GOP do quadro descoberto.

4.3 LIMITAÇÕES DO ESQUEMA

Essa seção destaca as limitações existentes no esquema criptográfico e serve como motivação para as extensões propostas no próximo capítulo.

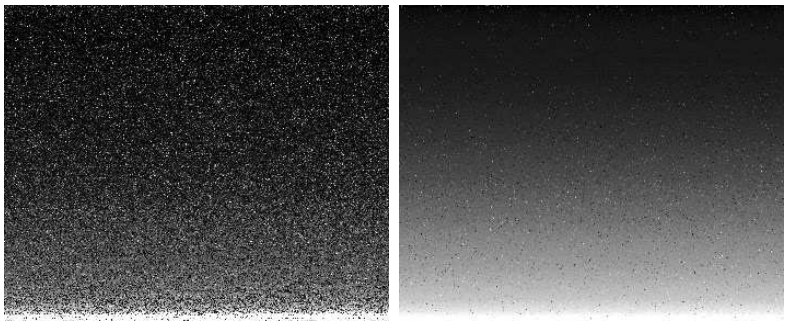
4.3.1 Vulnerabilidades de segurança

A análise de segurança feita na seção anterior mostrou que, na maioria das aplicações existentes, o sistema é seguro contra ataques de força bruta, texto puro conhecido/escolhido e cifrotexto escolhido. Entretanto, para algumas aplicações, qualquer informação sobre o vídeo original, mesmo que não se descubra o dado original propriamente, não deve ser revelada.

Primeiramente, o sistema criptográfico foi mostrado inseguro contra ataques de força bruta se o histograma do quadro é "estreito". Entretanto, com a extensão de ocultação do histograma, esse problema é resolvido. Ainda assim, essa extensão não esconde o histograma das diferenças de intensidade entre quadros consecutivos.

Se a aplicação não deseja revelar informação sobre a cena encriptada, tal como se a mesma possui pouco ou muito movimento, o sistema criptográfico também se mostra inadequado, pois essa informação pode ser estimada pela qualidade da "quase-ordenação"dos quadros, como pode ser visto na Figura 4.1.

Além disso, o histograma de alguns tipos de vídeos, como animações, paisagens, geralmente tem um formato conhecido e essa informação pode ser revelada usando o esquema básico. Se esse tipo de informação deve ser ocultada, o uso desse esquema criptográfico também não é encorajado.



(a) (b)

Figura 4.1: Revelando informação sobre cena: (a) cena movimentada - sequência *Bus*, (b) cena de pouco movimento - sequência *Deadline*

4.3.2 Restrições do módulo codificador

O esquema proposto é definido para a classe de codificadores apenas espaciais, podendo ser com ou sem perdas. Embora o esquema possa parecer genérico o suficiente para poder ser usado por qualquer codificador, aqui será mostrado que a técnica descrita não pode ser usado com codificadores em qualquer configuração.

Para os módulos codificadores, é necessário que um dado quadro seja codificado e imediatamente se possa saber o quadro codificado para poder calcular a permutação que será vista pelo decodificador. Isso significa que nenhum quadro pode ser codificado com base em quadros futuros, isto é, não pode haver nenhum quadro do tipo B.

Mais especificamente, não pode haver nenhum *atraso de codificação* no esquema utilizado. Alguns codificadores avançados utilizam o buffer no módulo temporal para guardar os quadros que estão na "fila"de codificação. A ideia de fazer isso é para poder escolher bons quadros de referência para utilizar correlação temporal e também para poder codificar quadros do tipo B. Para um quadro B ser codificado, é necessário que o quadro posterior já esteja disponível no fluxo de bits. Entretanto, isso gera um atraso de codificação de maneira que, no instante em que se codifica F_i, obtém-se o quadro codificado de $F_{i-atraso}$.

Quadros do tipo B são os que têm maior potencial de compressão, e são amplamente usados por codecs mais avançados, como MPEG-4. Logo, o esquema criptográfico perde um pouco seu potencial de compressão, pelo fato de não poder usar esse recurso. Ainda assim, o esquema pode fazer uso de quadros do tipo P, sendo aplicável a codecs temporais.

4.3.3 Sensibilidade a movimentos no vídeo

A performance em termos de qualidade de vídeo e compressibilidade está intimamente ligada à correlação temporal dos quadros do vídeo, uma vez que a qualidade da "quase-ordenação"depende disso e, portanto, é totalmente sensível à presença de muito movimento no vídeo. Isso implica na inviabilidade do esquema para vídeos com muito movimento.

Isso tenta ser amenizado com a extensão de translação de câmera constante. Porém,

ela é capaz de lidar apenas com movimento global e além disso, requer que o decriptador recebe informações de parâmetros de translação separadamente do vídeo, o que pode ser um tanto inconveniente. Outra preocupação é evitar que o atacante possa escutar informações de translação para não revelar dados sobre a cena.

CAPÍTULO 5

EXTENSÕES PROPOSTAS PARA O MÉTODO DE ENCRIPTAÇÃO

Neste capítulo, alguns dos problemas levantados na análise do sistema criptográfico do capítulo anterior são estudados e extensões são propostas para amenizar ou resolvê-los.

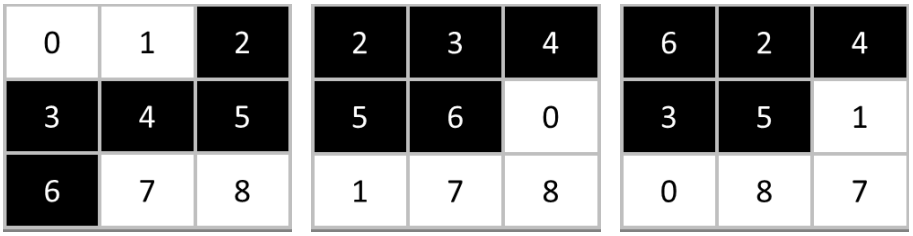
5.1 PERMUTAÇÃO DE ORDENAÇÃO ESTÁVEL

Na análise de complexidade de tempo do esquema, notou-se que o que determina na complexidade total de encriptação e decriptação de um dado quadro do vídeo no esquema é a computação da permutação de ordenação. No esquema original, os autores pensaram em um método usando abordagens de ordenação baseada em comparação de chaves, e o limite inferior para tais abordagens é $O(WH \times log(WH))$ [18].

Existe um algoritmo de ordenação para valores discretos, chamado *Counting Sort*, cuja complexidade de tempo é linear sobre o tamanho do dado a ser ordenado e do intervalo dos valores discretos possíveis [18]. Num quadro de vídeo, isso corresponde a $O(I_{max} + WH)$. Isso reduz à complexidade total do esquema a $O(I_{max} + WH)$, uma vez que as demais operações são lineares em WH.

Embora isso pareça afetar somente o tempo de processamento, a performance de compressão e qualidade de vídeo do sistema também é afetada, uma vez que a qualidade da 'quase-ordenação' depende diretamente da permutação. À primeira vista, pode-se imaginar que diferentes algoritmos de ordenação deveriam gerar o mesmo resultado. Diferentes algoritmos geram o mesmo quadro ordenado, porém podem utilizar permutações distintas. Basta observar que diferentes algoritmos de ordenação lidam valores de chaves iguais de maneira distintas.

Um algoritmo de ordenação é classificado como *estável* se ele garante que a ordem dos valores de chave igual é preservada no vetor ordenado [19]. Algoritmos de ordenação como



(a) (b) (c)

Figura 5.1: Comparação de permutação estável e instável: (a) quadro original com numeração dos pixels, (b) exemplo de permutação de ordenação estável, (c) exemplo de permutação de ordenação instável

Quicksort são, em geral, *instáveis*. Por outro lado, o *Counting Sort* é estável e o conceito de estabilidade de ordenação pode ser interpretado como correlação espacial num quadro de um vídeo. Isso tende a beneficiar a performance do sistema, pois, mesmo que haja erros de 'quase-ordenação', posições de intensidade igual na imagem de onde a permutação foi computada serão permutadas para posições próximas na imagem onde possivelmente ocorrerá erro de permutação. Dessa forma, ainda que não estejam ordenados, os pixels são de coordenadas próximas na imagem original e portanto, por correlação espacial, tendem a ser similares. Isso pode ser visto na Figura 5.1.

```
   Entrada: Quadro F
 1 início
 2     Inicialize h como um vetor com frequência zero para cada intensidade
 3     Inicialize P como um vetor de tamanho W × H
 4     Inicialize posicao ← 0
 5
 6     para 0 ≤ x < W faça
 7         para 0 ≤ y < H faça
 8             h[F[x][y]] ← h[F[x][y]] + 1
 9         fim
10     fim
11
12     para 1 ≤ x ≤ I_max faça
13         h[x] ← h[x] + h[x − 1]
14     fim
15
16     para 0 ≤ x < W faça
17         para 0 ≤ y < H faça
18             se F[x][y] > 0 então
19                 P[x] ← h[F[x][y] − 1]
20                 h[F[x][y] − 1] ← h[F[x][y] − 1] + 1
21             senão
22                 P[x] ← posicao
23                 posicao ← posicao + 1
24             fim
25         fim
26     fim
27
28     Retorne P
29 fim
```

Algoritmo 5: Algoritmo de permutação de ordenação baseado no *Counting Sort*

O *Counting Sort* se baseia no histograma do quadro do vídeo. A permutação de ordenação estável do quadro do vídeo pode ser computada usando o histograma cumulativo do mesmo. O pseudocódigo do algoritmo pode ser visto em detalhes no Algoritmo 5. As

linhas 6 - 10 computam o histograma da imagem, enquanto nas linhas 12 - 14, o histograma cumulativo é computado. O histograma cumulativo indica o *offset* que um pixel de uma dada intensidade será permutado. O valor do *offset* é reajustado a cada pixel que é processado e com isso, a permutação é definida para todos os pixels da imagem.

5.2 COMPENSAÇÃO DE MOVIMENTO

O algoritmo descrito no capítulo anterior utiliza a dualidade de permutações para manter ou aumentar a correlação espacial de um quadro. Entretanto, é assumido que a permutação do quadro anterior aplicado no quadro posterior resulta numa 'quase-ordenação', o que depende da correlação temporal dos quadros.

Para poder aprimorar a correlação temporal de quadros consecutivos, técnicas de alinhamento de imagens podem ser utilizadas. Uma questão similar ocorre na compressão de vídeos utilizando codecs mais avançados, onde técnicas de estimativa e compensação de movimento são utilizadas de maneira que a diferença residual dos quadros seja bastante compressível pelo módulo espacial do codificador. Esse mesmo princípio será aplicado nessa extensão: se o quadro anterior é compensado antes da computação da permutação, é mais provável que ela gerará uma 'quase-ordenação' mais adequada ao próximo quadro.

Técnicas de estimativa de movimento são basicamente divididas em dois métodos principais [20] : *backward motion estimation* (BME) e *forward motion estimation* (FME). FME se baseia tanto no quadro atual como em quadros transmitidos anteriormente. Entretanto, como o quadro atual não é conhecido pelo lado decodificador, os vetores de movimento precisam ser codificados no fluxo de bits ou transmitidos separadamente. BME, por outro lado, tem como base para a computação dos parâmetros de movimento apenas quadros anteriores que estão disponíveis em ambos os lados da transmissão.

Embora FME resulte em estimativa de movimento mais precisa, exige que o codificador utilizado suporte a existência de vetores de movimento ou que tais vetores sejam transmitidos separadamente. Isso afetaria a classe de codecs aplicáveis ao esquema, portanto as técnicas BME serão preferidas. Uma desvantagem de BME é que ambos os lados da transmissão precisam computar os parâmetros de movimento e devem usar o mesmo algoritmo, que deve ser determinístico. Como existem algoritmos de estimativa de movimento que podem ser aplicados em tempo real, isso não viola o requisito de tempo de processamento.

A ideia utilizada por essa extensão será estimar o próximo quadro com base nos quadros anteriores antes de computar a permutação de ordenação a ser aplicada no mesmo. Como uma técnica BME será utilizada, o quadro será previsto, assumindo preservação linear de movimento. Essa suposição é válida devido às altas taxas de quadro usadas em conteúdo de vídeo atualmente [21]. Seja F o quadro a ser encriptado, F_{-1} e F_{-2} os dois últimos quadros encriptados e decriptados pelo esquema. Se os parâmetros de movimento são computados de F_{-1} para F_{-2}, o quadro F pode ser estimado pela extrapolação de tais parâmetros, como visto na Figura 5.2. Após isso, a permutação a ser aplicada em F é computada sobre a previsão do quadro F.

Um princípio similar foi aplicado em [21], entretanto estimando F como interpolação de F_{-1} e F_{+1}. Essa técnica, entretanto, se baseia no uso de quadros do tipo B e, como

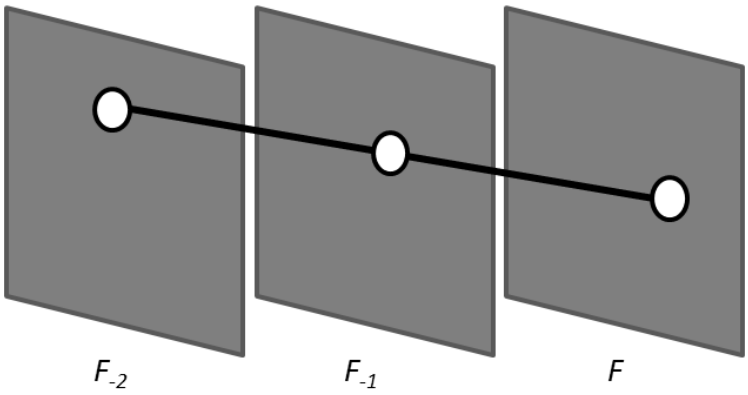


Figura 5.2: Movimento linear: princípio de preservação dos parâmetros de movimento entre quadros consecutivos

mostrado no capítulo anterior, esse recurso não pode ser usado no esquema original. Os autores da técnica sugerem que isso pode ser feito sem usar quadros do tipo B, se a estimativa de F é realizada como extrapolação de F_{-1} e F_{-2}, como explicado anteriormente.

5.2.1 Algoritmos de encriptação e decriptação estendidos

Para essa extensão, a notação utilizada será estendida:

- $F_i^{previsto}$ corresponde à previsão do quadro F_i com base nos seus dois quadros anteriores.
- *prever_quadro* é uma sub-rotina que tendo dois quadros como parâmetros, retorna a previsão do próximo quadro. Sua implementação será detalhada posteriormente.

O pseudocódigo do algoritmo de encriptação pode ser visto no Algoritmo 6. Para codecs sem perdas, a previsão de quadros pode ser feita diretamente nos quadros de entrada. O Algoritmo 7 mostra o processo de decriptação estendido.

5.2.2 Predição do quadro usando estimativa de movimento

Esta seção detalha como a função *prever_quadro* pode ser implementada usando algoritmos de estimativa de movimento. A ideia é basicamente computar parâmetros de estimativa de movimento sobre as imagens dos dois quadros anteriores e estimar o quadro atual como extrapolação desses parâmetros. Em relação aos algoritmos de estimativa de movimento utilizáveis, é importante que:

- **o algoritmo seja determinístico** : essa condição é necessária para garantir que a permutação computada nos dois lados da comunicação será exatamente a mesma.
- **os parâmetros de movimento são extensíveis linearmente** : isso é uma condição importante para a extrapolação do quadro. Isso significa que qualquer

Entrada: Vídeo $F_1, ..., F_N$
início
  Obtenha o quadro $F_1^{transmitido} = E(F_1)$
  Transmita $F_1^{transmitido}$ pelo canal ChS

  Obtenha o quadro $F_1^{decriptado} = D(F_1^{transmitido})$
  Compute a permutação P_1 do quadro $F_1^{decriptado}$

  para cada F_i, *onde* $i = 2, ..., N$ **faça**
    Obtenha o quadro $F_i^{transmitido} = E(P_{i-1}(F_i))$
    Transmita $F_i^{transmitido}$ pelo canal ChR

    Obtenha o quadro $F_i^{decriptado} = P_{i-1}^{-1}(D(F_i^{transmitido}))$
    Obtenha o quadro $F_{i+1}^{previsto} = prever\_quadro(F_i^{decriptado}, F_{i-1}^{decriptado})$

    se $i < N$ **então**
      Compute a permutação P_i do quadro $F_{i+1}^{previsto}$
    fim
  fim
fim

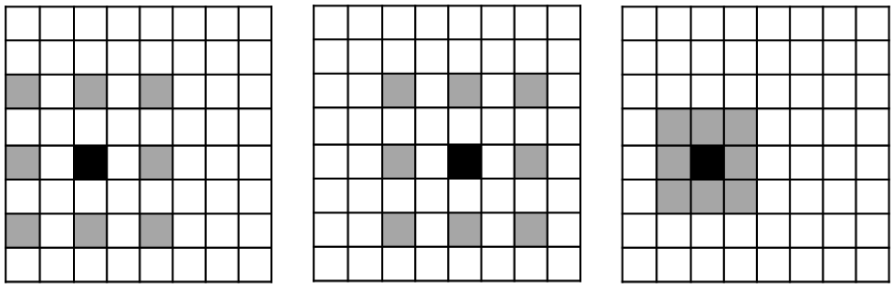
Algoritmo 6: Algoritmo de encriptação de vídeo com compensação de movimento

Entrada: Vídeo encriptado $F_1^{transmitido}, F_2^{transmitido}, ..., F_N^{transmitido}$
início
  Receba $F_1^{transmitido}$ pelo canal ChS
  Obtenha o quadro $F_1^{decriptado} = D(F_1^{transmitido})$
  Compute a permutação P_1 do quadro $F_1^{decriptado}$

  para cada $F_i^{transmitido}$, *onde* $i = 2, ..., N$ **faça**
    Obtenha o quadro $F_i^{decriptado} = P_{i-1}^{-1}(D(F_i^{transmitido}))$
    Obtenha o quadro $F_{i+1}^{previsto} = prever\_quadro(F_i^{decriptado}, F_{i-1}^{decriptado})$

    se $i < N$ **então**
      Compute a permutação P_i do quadro $F_{i+1}^{previsto}$
    fim
  fim
fim

Algoritmo 7: Algoritmo de decriptação de vídeo com compensação de movimento



(a) (b) (c)

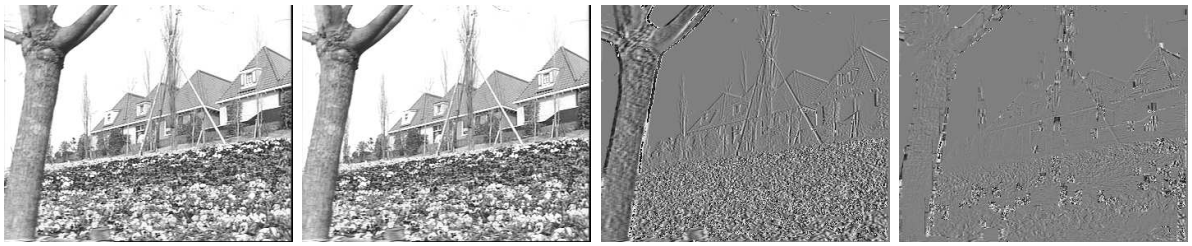
Figura 5.3: Busca logarítmica 2D (pixel preto representa centro da janela, enquanto os cinzas são centros dos blocos candidatos): (a) estado inicial, (b) estado posterior caso o melhor bloco seja o cujo centro está à direita do centro anterior, (c) estado posterior caso o melhor bloco seja aquele cujo centro está no centro da janela atual.

quadro futuro pode ser estimado usando os parâmetros, assumindo que os parâmetros são preservados entre quadros consecutivos.

Como o esquema original é destinado a aplicações em tempo real, é recomendado que o algoritmo de estimativa de movimento não seja muito computacionalmente custoso. Uma abordagem sugerida seria o uso de estimativa de movimento baseada em blocos, que é amplamente utilizada em codificação de vídeo. Existem inúmeras implementações que dão bons resultados com algoritmos bastante eficientes. Além disso, abordagens baseadas em bloco, em geral, podem fazer uso de computação paralela intensiva, uma vez que as computações de parâmetros de movimento para blocos distintos são totalmente independentes. Apesar de que qualquer algoritmo de estimativa de movimento com as características citadas possa ser utilizado, esse trabalho estuda o cenário em que um algoritmo de casamento de blocos é usado para a estimativa e compensação de movimento.

Algoritmos de casamento de blocos buscam encontrar, para cada bloco de uma dada imagem, o bloco de melhor casamento de intensidade dos pixels em uma outra imagem de referência. Dois algoritmos de casamento de blocos serão considerados nesse trabalho. O primeiro é conhecido como *Busca Completa* (*Full Search*) [22] e considera todos os blocos candidatos em uma região retangular limitada, garantindo o melhor casamento na região. Como isso pode ser computacionalmente pesado, diversos algoritmos [23, 24, 25] foram propostos para diminuir o número de posições a serem consideradas, entretanto sendo sub-ótimos. Um desses algoritmos será considerado: a busca logarítmica 2D [22].

A *Busca Logarítmica* (*2D Logarithmic Search*) inicia com um tamanho de janela e considera a cada passo os 9 pontos extremos da janela. Se o melhor casamento dos blocos for no centro da janela, seu tamanho é reduzido pela metade, caso contrário, o centro da janela é movido para o ponto de melhor casamento. O algoritmo pára quando o tamanho da janela for reduzido a 1. Note que a cada passo, o algoritmo escolhe um bloco de casamento melhor ou igual ao do passo anterior. Embora o algoritmo seja bastante rápido, ele pode parar num mínimo local. Um exemplo de execução do algoritmo pode ser visto na Figura 5.3.



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

Figura 5.4: Residual entre quadros consecutivos - sequência *Flower*: (a) quadro anterior, (b) quadro atual, (c) residual de (b) e (a) - PNG 148 KB, (d) residual entre (b) e (a) com movimento compensado - PNG 113 KB

Existem diferentes implementações para a função de similaridade de blocos para avaliar o melhor casamento dos mesmos. As métricas mais utilizadas são o *SAD* (soma das diferenças absolutas), o *MSE* (média dos erros quadráticos) e *SSE* (soma dos erros quadráticos).

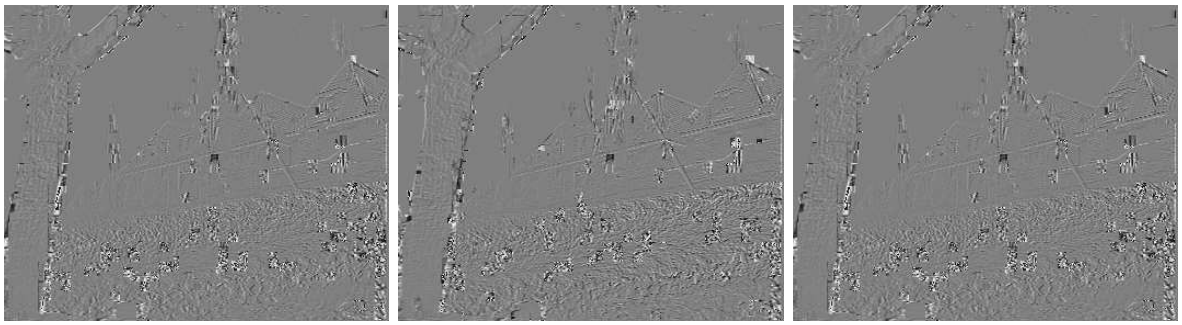
Para prever o próximo quadro, uma abordagem similar à proposta em [21] pode ser utilizada. Assumindo que os parâmetros de movimento são os de um algoritmo de casamento de blocos entre os quadros F_{-2} e F_{-1}, tais parâmetros contêm os vetores de movimento de um bloco de F_{-1} para outro de F_{-2}. Se a orientação dos vetores de movimento for invertidas, para cada bloco de F_{-1}, tem-se o bloco onde se estima que o mesmo esteja no quadro a ser previsto F. Dessa forma, tal bloco na imagem F tem dois blocos de referência e pode ter cada pixel estimado como a média da intensidade dos dois pixels correspondentes nos dois blocos de referência.

5.3 ENCRIPTAÇÃO DA DIFERENÇA RESIDUAL DOS QUADROS

Como visto no segundo capítulo, a abordagem mais convencional de explorar correlação temporal é codificando a diferença residual entre quadros consecutivos pelo módulo espacial. A ideia dessa extensão é aplicar os conceitos do módulo temporal, como parte do algoritmo de encriptação, sobre o codificador espacial do esquema criptográfico, que funciona de maneira similar ao módulo espacial de um codificador avançado.

Uma questão relevante é se a diferença residual dos quadros em si pode ser considerada uma encriptação. Como explicado em [3], codificar a diferença residual dos quadros não é suficiente para encriptar um vídeo, pois em geral, a qualidade de percepção da imagem residual é alta, principalmente em vídeos com bastante movimento. Dessa forma, o residual ainda precisa ser encriptado.

No contexto de compressão de vídeo, nota-se que o uso de permutação no método de encriptação tem como objetivo transformar a correlação temporal do vídeo em correlação espacial. Embora tenha outro objetivo, a extensão de ocultação de histograma explora a redundância de informação entre quadros consecutivos. Entretanto, a extensão de ocultação de histograma codifica a diferença sobre as imagens permutadas por uma mesma permutação secreta (a do quadro anterior), assumindo que as mesmas são temporalmente correlacionadas.



(a) (b) (c)

Figura 5.5: Três últimos quadros residuais (com compensação de movimento) - sequência *Flower*: (a) antepenúltimo quadro residual, (b) penúltimo quadro residual, (c) último quadro residual

O esquema de compensação de movimento definido na seção anterior pode ser combinado com a técnica de ocultação de histograma para explorar mais adequadamente a correlação temporal de quadros consecutivos. A ideia dessa extensão é usar a compensação de movimento antes de calcular o residual de dois quadros com o intuito de diminuir a quantidade de informação do residual e, além disso, codificá-lo permutado com a ordenação do residual anterior.

Para compreender o princípio dessa técnica, na Figura 5.4 o residual entre quadros consecutivos com e sem compensação de movimento (usando o algoritmo mostrado na seção anterior) é mostrado. Nota-se que, após a compensação de movimento, o residual da imagem tende a ser menor. Na Figura 5.5, pode-se perceber que, mesmo com compensação de movimento, o residual dos quadros é temporalmente correlacionado. Dessa forma, o residual de quadros consecutivos pode ser encriptado com a permutação do residual anterior de maneira a possivelmente melhorar a sua correlação espacial.

5.3.1 Algoritmos de encriptação e decriptação estendidos

Para definir os algoritmos, a notação será estendida:

- $F_i^{residual}$ corresponde ao quadro que representa a diferença residual entre quadros consecutivos. Na prática, se $i > 2$, $F_i^{residual} = F_i - F_i^{previsto}$, senão, $F_i^{residual} = F_i - F_{i-1}$. $F_i^{residual\_decriptado}$ corresponde à versão de $F_i^{residual}$ que será vista pelo lado decriptador (a codificação com perdas pode torná-lo diferente de $F_i^{residual}$).

- *residual* é uma sub-rotina que, tendo dois quadros como parâmetros, retorna a diferença residual entre os mesmos.

- *reconstruir* é uma sub-rotina que, tendo o quadro anterior e o residual do atual em relação a ele, retorna o quadro atual reconstruído.

Sendo assim, o algoritmo de encriptação e decriptação pode ser estendido. Os Algoritmos 8 e 9 mostram o pseudocódigo para a encriptação e decriptação usando essa

Entrada: Vídeo $F_1, ..., F_N$

início

  Obtenha o quadro $F_1^{transmitido} = E(F_1)$

  Transmita $F_1^{transmitido}$ pelo canal ChS

  Obtenha o quadro $F_1^{decriptado} = D(F_1^{transmitido})$

  Compute a permutação P_1 do quadro $F_1^{decriptado}$

  Obtenha o quadro $F_2^{transmitido} = E(P_1(F_2))$

  Transmita $F_2^{transmitido}$ pelo canal ChR

  Obtenha o quadro $F_2^{decriptado} = P_1^{-1}(D(F_2^{transmitido}))$

  Obtenha o quadro $F_3^{previsto} = prever\_quadro(F_2^{decriptado}, F_1^{decriptado})$

  Obtenha o quadro $F_2^{residual\_decriptado} = residual(F_2^{decriptado}, F_1^{decriptado})$

  Compute a permutação P_2 do quadro $F_2^{residual\_decriptado}$

  para cada F_i, *onde* $i = 3, ..., N$ **faça**

    Obtenha o quadro $F_i^{residual\_original} = residual(F_i, F_i^{previsto})$

    Obtenha o quadro $F_i^{transmitido} = E(P_{i-1}(F_i^{residual\_original}))$

    Transmita $F_i^{transmitido}$ pelo canal ChR

    Obtenha o quadro $F_i^{residual\_decriptado} = P_{i-1}^{-1}(D(F_i^{transmitido}))$

    Obtenha o quadro $F_i^{decriptado} = reconstruir(F_i^{previsto}, F_i^{residual\_decriptado})$

    Obtenha o quadro $F_{i+1}^{previsto} = prever\_quadro(F_i^{decriptado}, F_{i-1}^{decriptado})$

    se $i < N$ **então**

      Compute a permutação P_i do quadro $F_i^{residual\_decriptado}$

    fim

  fim

fim

Algoritmo 8: Encriptação de vídeo sobre os residuais dos quadros

técnica. No algoritmo, apenas a partir do terceiro quadro, o residual é encriptado, pois no segundo quadro, tem-se o primeiro residual, o qual não tem nenhuma permutação de ordenação de residual anterior para ser permutado. O segundo quadro é encriptado como no algoritmo básico, pois é o primeiro residual do vídeo, não tendo, portanto, nenhuma permutação de residual anterior.

Entrada: Vídeo encriptado $F_1^{transmitido}, F_2^{transmitido}, ..., F_N^{transmitido}$
início
  Receba $F_1^{transmitido}$ pelo canal ChS
  Obtenha o quadro $F_1^{decriptado} = D(F_1^{transmitido})$
  Compute a permutação P_1 do quadro $F_1^{decriptado}$

  Receba $F_2^{transmitido}$ pelo canal ChR
  Obtenha o quadro $F_2^{decriptado} = P_1^{-1}(D(F_2^{transmitido}))$
  Obtenha o quadro $F_3^{previsto} = prever\_quadro(F_2^{decriptado}, F_1^{decriptado})$
  Obtenha o quadro $F_2^{residual\_decriptado} = residual(F_2^{decriptado}, F_1^{decriptado})$
  Compute a permutação P_1 do quadro $F_2^{residual\_decriptado}$

  para cada $F_i^{transmitido}$, *onde* $i = 3, ..., N$ **faça**
    Obtenha o quadro $F_i^{residual\_decriptado} = P_{i-1}^{-1}(D(F_i^{transmitido}))$
    Obtenha o quadro $F_i^{decriptado} = reconstruir(F_i^{previsto}, F_i^{residual\_decriptado})$
    Obtenha o quadro $F_{i+1}^{previsto} = prever\_quadro(F_i^{decriptado}, F_{i-1}^{decriptado})$

    se $i < N$ **então**
      Compute a permutação P_i do quadro $F_i^{residual\_decriptado}$
    fim
  fim
fim

Algoritmo 9: Decriptação de vídeo sobre os residuais dos quadros

A ideia para o cálculo do quadro residual é utilizando a diferença formalizada pela extensão de ocultação de histograma (Equações 3.2.1 e 3.2.2). Entretanto ela apenas é aplicável para codecs com perdas. Para codificadores sem perdas, a diferença residual pode ser diretamente calculado pela diferença dos quadros (usando aritmética modular sobre 2^n, onde n é a resolução de bits da intensidade do pixel). A reconstrução a partir do residual consiste na aplicação do procedimento inverso. Se o residual é codificado centralizado em $\frac{I_{max}}{2}$, basta subtrair o valor de cada pixel de $\frac{I_{max}}{2}$ e somar o valor resultante à intensidade do pixel da imagem anterior correspondente. No caso de codecs sem perdas, deve-se usar aritmética modular no processo inverso também.

CAPÍTULO 6

EXPERIMENTOS E RESULTADOS

Este capítulo detalha os experimentos realizados para avaliar o desempenho das modificações propostas neste trabalho. Os experimentos avaliam a eficácia do método em termos de compressão e qualidade de vídeo. A avaliação da segurança do vídeo encriptado está fora do escopo desses experimentos.

6.1 MÉTRICAS DE COMPRESSÃO E QUALIDADE DO VÍDEO

A métrica mais conhecida para avaliar compressão de um codificador de vídeo é a *taxa de compressão*. A *taxa de compressão* é a razão entre o tamanho do fluxo de bits do vídeo codificado e do vídeo original. Como uma comparação entre taxas de compressão será feita, a taxa de redução do tamanho do fluxo de bits será avaliada.

No caso de um codificador com perdas, a qualidade da codificação de vídeo será medida com o PSNR médio (*peak signal-to-noise ratio*) dos quadros decodificados. O PSNR é uma métrica bastante utilizada em codificação de sinais. Sendo I_{max} a intensidade máxima de acordo com a resolução de bits utilizada, o PSNR de uma imagem F_{ruido} em relação a uma imagem original $F_{original}$ é dado pelas Equações 6.1.1 e 6.1.2.

$$PSNR = 10 \times log_{10}\left(\frac{I_{max}^2}{MSE}\right) \tag{6.1.1}$$

$$MSE = \frac{\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\left(F_{ruido}(x,y) - F_{original}(x,y)\right)^2}{W \times H} \tag{6.1.2}$$

Vale notar que para imagens idênticas o valor do PSNR é indefinido (MSE se torna zero e é denominador da expressão). Quanto maior o valor do PSNR, melhor tende a ser a qualidade da imagem codificada, isto é, menor é o ruído em relação à informação original.

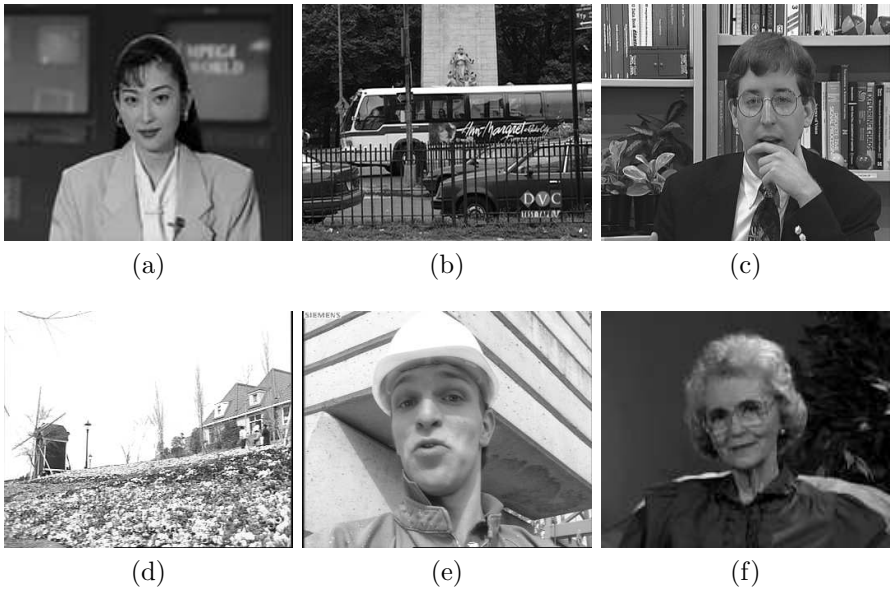
6.2 SEQUÊNCIAS DE VÍDEO UTILIZADAS

As amostras de vídeo utilizadas nos experimentos foram obtidas de um conjunto de sequências padrões para realização de experimentos de processamento de vídeo. Para o experimento, as sequências precisaram ser convertidas sem perdas para escala de cinza, onde o nível de cinza é dado pela média das intensidades de cada cor.

Algumas das sequências estavam no formato CIF, enquanto outras no formato QCIF (com um quarto da resolução em área). O intuito de ter sequências em resoluções distintas é avaliar a eficácia do método em resoluções de vídeo distintas. Na Tabela 6.1, o número

Nome da sequência	Número de quadros	Formato	Resolução
Akiyo	300	QCIF	176x144
Bus	150	CIF	352x288
Deadline	1374	CIF	352x288
Flower	250	CIF	352x288
Foreman	300	CIF	352x288
Grandma	870	QCIF	176x144

Tabela 6.1: Dados das sequências de vídeo utilizadas



(a) (b) (c)

(d) (e) (f)

Figura 6.1: Sequências de vídeo utilizadas: (a) *Akiyo*, (b) *Bus*, (c) *Deadline*, (d) *Flower*, (e) *Foreman*, (f) *Grandma*

de quadros e a resolução de cada sequência podem ser vistos e a Figura 6.1 mostra um exemplo de quadro de cada sequência.

Uma das questões relevantes para as extensões propostas é quão movimentada é a sequência do vídeo. Dessa forma, as sequências foram categorizadas em relação ao movimento existente na mesma.

As sequências *Akiyo* e *Grandma* são de muito pouco movimento, de maneira que quadros consecutivos são quase idênticos. A sequência *Deadline* possui um pouco mais de movimento e é menos uniforme. As demais sequências *Bus*, *Flower* e *Foreman* contém movimentos significativos e são usadas em comparação de algoritmos de estimativa de movimento, por exemplo.

A sequência *Flower* possui bastante movimento, entretanto é relativamente simples, uma vez que corresponde a uma translação de câmera quase uniforme. Por outro lado, *Bus* tem bastante movimento e é bem mais complicada, pois há movimento de zoom, translação global de câmera e ainda movimentos locais (*local motion*). *Foreman* é uma

cena menos movimentada que as duas anteriores, entretanto não é muito uniforme, pois a câmera balança um pouco e há movimentos repentinos, mais difíceis de prever.

6.3 AVALIAÇÃO DAS EXTENSÕES PROPOSTAS

Cada extensão será avaliada separadamente, considerando o esquema original como referência para comparação.

6.3.1 Permutação de ordenação estável

Nesse experimento, as sequências foram encriptadas usando o algoritmo básico visto no capítulo 3. Sem usar nenhuma extensão, a única variação foi a computação da permutação de ordenação que é feita com *Quicksort*, como proposto pelos autores do esquema, e com a variação do *Counting Sort*, como proposto pela extensão.

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	*Grandma*	*Deadline*	*Bus*	*Flower*	*Foreman*
Variação do *Quicksort*	3.124	13.578	100.512	18.694	20.887	29.148
Proposto	2.912	13.465	98.929	18.540	20.772	28.746

Tabela 6.2: Comparação do tamanho do fluxo de bits (em KB) para diferentes métodos de permutação de ordenação numa encriptação usando codec MPNG

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	*Grandma*	*Deadline*	*Bus*	*Flower*	*Foreman*
Variação do *Quicksort*	1.017	2.908	31.893	7.842	11.177	10.514
Proposto	978	2.879	30.573	7.700	10.415	10.243

Tabela 6.3: Comparação do tamanho do fluxo de bits (em KB) para diferentes métodos de permutação de ordenação numa encriptação usando codec MJPEG

O experimento foi realizado primeiramente usando o codificador sem perdas MPNG. Em todas as sequências houve diminuição no tamanho do fluxo de bits em relação ao esquema original, como pode ser visto na Tabela 6.2. A comparação de PSNR não é necessária, pois o codificador é sem perdas.

Os resultados de tamanho do fluxo de bits e PSNR para o codificador com perdas MJPEG (fixando as configurações do codec) podem ser vistos nas Tabelas 6.3 e 6.4. Em todos os exemplos, houve melhora tanto no PSNR médio como no tamanho do fluxo de bits.

O maior benefício dessa extensão consiste no tempo de processamento que é significativamente reduzido, pois a complexidade de tempo de todo o esquema é determinada

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	*Grandma*	*Deadline*	*Bus*	*Flower*	*Foreman*
Variação do *Quicksort*	38,8331	38,6844	36,8875	35,9372	36,9866	36,2588
Proposto	39,2015	38,8709	37,11	35,9453	37,2788	36,3606

Tabela 6.4: Comparação do PSNR médio (em dB) para diferentes métodos de permutação de ordenação numa encriptação usando codec MJPEG

pela computação da permutação de ordenação.

6.3.2 Compensação de movimento

Como os benefícios dessa extensão é voltada para sequências com movimento, é esperado que os resultados relevantes do experimento sejam visíveis nas três sequências de movimento significativo. O experimento foi realizado para o codificador MJPEG, testando três configurações. A primeira consiste no algoritmo de encriptação original. A segunda consiste na extensão usando o algoritmo de casamento de blocos *Busca Completa* numa região 16x16. A terceira consiste no algoritmo estendido computando os vetores de movimento com a *Busca Logarítmica*.

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	Grandma	Deadline	Bus	Flower	Foreman
Original	1.017	2.908	31.893	7.842	11.177	10.514
Busca Logarítmica 2D	1.005	2.905	30.758	6.907	8.953	10.015
Busca Completa 16x16	1.022	2.982	31.633	6.162	7.729	9.663

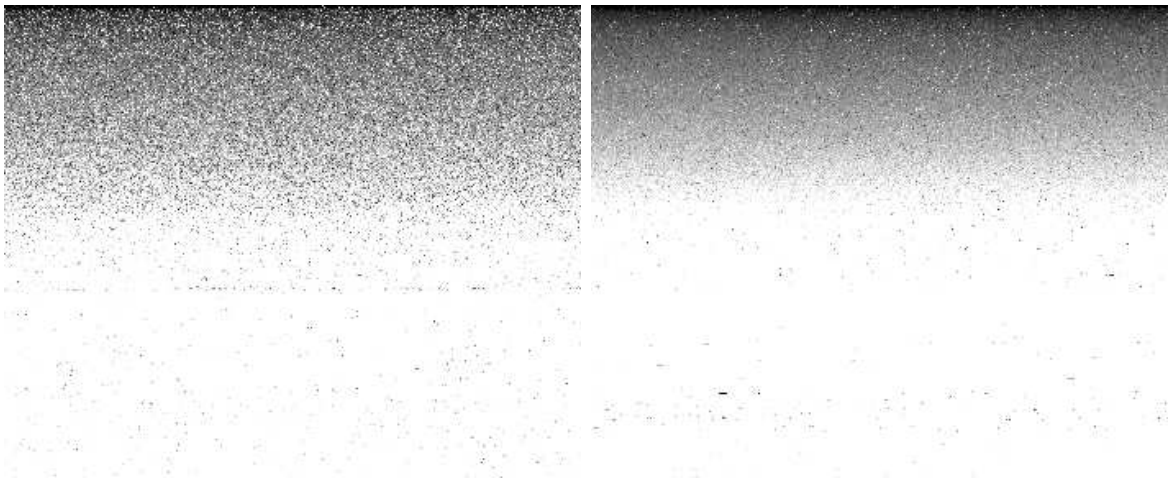
Tabela 6.5: Comparação do tamanho do fluxo de bits (em KB) para a extensão de compensação de movimento numa encriptação usando o codec MJPEG

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	Grandma	Deadline	Bus	Flower	Foreman
Original	38,8331	38,6844	36,8875	35,9372	36,9866	36,2588
Busca Logarítmica 2D	38,8408	38,6776	36,919	35,9624	37,1386	36,2756
Busca Completa 16x16	38,7873	38,6001	36,8264	35,9992	37,3005	36,2953

Tabela 6.6: Comparação do PSNR médio (em dB) para a extensão de compensação de movimento numa encriptação usando codec MJPEG

Algoritmo	Flower	Bus	Foreman
Proposto (*Busca Completa 16x16*)	30,84%	21,42%	8,09%
Proposto (*Busca Logarítmica 2D*)	19,89%	11,92%	4,74%

Tabela 6.7: Comparação de taxa de redução do tamanho do fluxo de bits para a extensão de compensação de movimento numa encriptação usando o codec MJPEG



(a) (b)

Figura 6.2: Qualidade da permutação no quadro 4 da sequência *Flower*: (a) algoritmo original, (b) algoritmo estendido

Para as versões estendidas, foi usado o tamanho de bloco como *8x8* e a métrica de casamento de blocos utilizada (função de custo a ser minimizada) foi o *SAD* (soma das diferenças absolutas).

Os resultados de tamanho do fluxo de bits e PSNR para o codificador com perdas MJPEG podem ser vistos nas Tabelas 6.5 e 6.6. A redução do tamanho do fluxo de bits foi bastante significativa em todas as sequências de muito movimento, sobretudo pelo fato de tais sequências serem bastante pequenas. A Tabela 6.7 mostra a redução percentual relativa ao esquema original. Houve também uma melhora na qualidade do vídeo nos três exemplos de cenas de movimento significativo.

Note que para sequências de pouco movimento, a diferença no tamanho do fluxo de bits é muito pouco relevante. Há casos de sequências de pouco movimento em que o estendido tem desempenho ligeiramente pior, mas isso é justificável por pequenos erros de predição de quadro e também pelo fato de existir muito pouco movimento em tais cenas.

É interessante notar que o ganho com o uso da técnica foi proporcional à qualidade da estimativa de movimento utilizada. Dependendo dos requisitos da aplicação em termos de tempo de processamento e tamanho do fluxo de bits, o algoritmo de estimativa de movimento mais adequado pode ser utilizado.

A sequência *Flower* foi a de melhor desempenho da extensão, o que é esperado, uma

vez que o movimento nesse vídeo é bastante linear. Na Figura 6.2, pode-se notar a melhora da qualidade da *'quase-ordenação'* com a técnica. Por outro lado, *Foreman* é uma sequência com pouco movimento linear, e foi a que resultou em menor melhora.

6.3.3 Encriptação da diferença residual de quadros

A melhora dessa extensão também tende a ser relevante em vídeos com movimento significativo. O algoritmo básico com a extensão de ocultação de histograma é comparado com a versão com compensação de movimento, de maneira similar ao experimento anterior. As Tabelas 6.8 e 6.9 mostram a comparação em relação a tamanho do fluxo de bits e qualidade de vídeo. O ganho foi bastante similar ao do experimento anterior, como atestado pela Tabela 6.10, significando que o princípio de compensação também é válido para o uso com a extensão de ocultação de histograma.

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	Grandma	Deadline	Bus	Flower	Foreman
Original	720	2.486	25.577	7.589	12.108	10.594
Busca Logarítmica 2D	718	2.503	25.525	6.698	9.709	9.997
Busca Completa 16x16	718	2.572	25.925	5.936	8.588	9.603

Tabela 6.8: Comparação do tamanho do fluxo de bits (em KB) para a extensão de compensação de movimento numa encriptação do residual usando codec MJPEG

Algoritmo	Quantidade de movimento na sequência					
	Pouco		Médio	Muito		
	Akiyo	Grandma	Deadline	Bus	Flower	Foreman
Original	39,6183	38,907	37,1604	31,4628	33,0015	36,0641
Busca Logarítmica 2D	39,6057	38,8741	37,112	32,5151	34,8649	36,0146
Busca Completa 16x16	39,5919	38,7997	37,0544	33,8136	35,8655	35,9848

Tabela 6.9: Comparação do PSNR médio (em dB) para a extensão de compensação de movimento numa encriptação do residual usando codec MJPEG

Algoritmo	Flower	Bus	Foreman
Proposto (*Busca Completa 16x16*)	29,97%	21,78%	9,35%
Proposto (*Busca Logarítmica 2D*)	19,81%	11,74%	5,63%

Tabela 6.10: Comparação de taxa de redução do tamanho do fluxo de bits para a extensão de compensação de movimento numa encriptação do residual usando codec MJPEG

Pelos experimentos, a extensão de ocultação de histograma em geral permite uma taxa de compressão e qualidade de vídeo maior para sequências com pouco movimento, o que é esperado, pois em tais casos, a diferença residual é bastante pequena entre quadros consecutivos. A compensação de movimento permitiu aumento significativo (de até quase 10%) na qualidade de vídeo para cenas movimentadas.

CAPÍTULO 7

CONCLUSÃO E TRABALHOS FUTUROS

A técnica de encriptação estudada por este trabalho é bastante adequada aos requisitos de aplicações de encriptação de vídeo. Seus pontos fracos de seguranças conhecidos são toleráveis pela grande maioria das aplicações existentes.

Um dos pontos fundamentais da técnica é que ela tende a melhorar a correlação espacial dos quadros do vídeo, podendo ser vista, dessa forma, como uma técnica para compressão de vídeos. O uso de permutações por essa técnica consegue transformar a correlação temporal do vídeo em correlação espacial.

Este trabalho descreveu uma melhora para a computação da permutação de ordenação, de forma que a performance em tempo de processamento, taxa de compressão e qualidade de imagem foi melhorada. O tempo de processamento do sistema criptográfico como um todo foi reduzido à complexidade linear sobre o tamanho do vídeo.

O método, no entanto, é voltado para vídeos com baixa quantidade de movimento, uma vez que seu desempenho é intimamente dependente da qualidade da sua 'quase-ordenação', o que depende da correlação temporal de quadros consecutivos. Entretanto, mostrou-se que o esquema pode facilmente incorporar um mecanismo de compensação de movimento que permite aprimorar significativamente o desempenho do algoritmo, em termos de taxa de compressão para sequências com grande quantidade de movimento.

A extensão de compensação de movimento estudada não compromete a independência do codec utilizado, de maneira que qualquer codificador somente espacial pode ser usado com o algoritmo estendido. O trabalho mostrou também que tal esquema pode ser combinado com a ocultação de histograma, melhorando sua performance.

Para a estimativa de movimento utilizada, apenas abordagens baseadas em blocos foram estudadas. Entretanto, existem diversos outros métodos, como os baseados em pixel, os de processamento em multiresolução, cuja aplicabilidade pode ser estudada num trabalho futuro.

Além disso, no contexto de estimativa de movimento, diversos parâmetros podem ser calculados e esse trabalho focou apenas em translação. Incorporar outros parâmetros, como rotação, zoom, pode aprimorar os resultados da técnica e esse pode ser um alvo de pesquisa futura.

As técnicas de estimativa de movimento estudadas se baseiam apenas em métodos BME para não violar a independência de codec do esquema. Entretanto, os métodos FME geram estimativa de movimento bem mais precisa, além de permitir que o lado decodificador não precise computar quaisquer parâmetros de movimento. Uma outra área de trabalho futuro seria a construção de uma versão do método de encriptação que permita explorar técnicas FME, sendo específica para codificadores mais avançados, que permitem a codificação de vetores de movimento no fluxo de bits, de maneira a ter desempenho melhor de compressão e qualidade de vídeo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

[2] F. Liu and H. Koenig, "A novel encryption algorithm for high resolution video," in *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '05, (New York, NY, USA), pp. 69–74, 2005.

[3] D. Socek, S. Magliveras, D. Ćulibrk, O. Marques, H. Kalva, and B. Furht, "Digital video encryption algorithms based on correlation-preserving permutations," *EURASIP J. Inf. Secur.*, vol. 2007, pp. 10:1–10:8, Jan. 2007.

[4] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm," in *Proceedings of the 6th ACM International Conference on Multimedia*, MULTIMEDIA '98, (New York, NY, USA), pp. 81–88, 1998.

[5] C. Shi and B. Bhargava, "An efficient MPEG video encryption algorithm," in *Proceedings of the The 17th IEEE Symposium on Reliable Distributed Systems*, SRDS '98, (Washington, DC, USA), p. 381, IEEE Computer Society, 1998.

[6] T. Maples and G. Spanos, "Performance study of a selective encryption scheme for the security of networked, real-time video," in *Proceedings of the 4th International Conference on Computer Communications and Networks*, ICCCN '95, (Washington, DC, USA), p. 2, IEEE Computer Society, 1995.

[7] S. Li, G. Chen, and X. Zheng, "Chaos-based encryption for digital images and videos," in *Multimedia Security Handbook* (B. Furht and D. Kirovski, eds.), ch. 4, pp. 133–167, CRC Press, LLC, 2004.

[8] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 3rd ed., 2006.

[9] S. Ihara, *Information Theory for Continuous Systems*. Series on Probability and Statistics, World Scientific, 1993.

[10] S. Esakkirajan, T. Veerakumar, and P. Navaneethan, "Adaptive vector quantization based video compression scheme," in *Multimedia, Signal Processing and Communication Technologies, 2009. IMPACT '09. International*, pp. 40–43, March 2009.

[11] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, Sept. 1952.

[12] K. Mayer-Patel, L. Le, and G. Carle, "An MPEG performance model and its application to adaptive forward error correction," in *Proceedings of the 10th ACM International Conference on Multimedia*, MULTIMEDIA '02, (New York, NY, USA), pp. 1–10, 2002.

[13] M. Burrows, D. J. Wheeler, M. Burrows, and D. J. Wheeler, "A block-sorting lossless data compression algorithm," tech. rep., Digital Systems Research Center, Palo Alto, California, USA, 1994.

[14] L. Tang, "Methods for encrypting and decrypting MPEG video data efficiently," in *Proceedings of the 4th ACM International Conference on Multimedia*, MULTIMEDIA '96, (New York, NY, USA), pp. 219–229, 1996.

[15] B. Bhargava, C. Shi, and S. Wang, "MPEG video encryption algorithms," *Multimedia Tools and Applications Journal*, vol. 24, pp. 57–79, Sept. 2004.

[16] J. Katz and Y. Lindell, *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[17] S. Li, C. Li, G. Chen, N. Bourbakis, and K. Lo, "A general cryptanalysis of permutation-only multimedia encryption algorithms. IACR's Cryptology ePrint Archive: Report 2004/374," 2004.

[18] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd ed., 2001.

[19] E. C. Horvath, "Stable sorting in asymptotically optimal time and extra space," *J. ACM*, vol. 25, pp. 177–199, Apr. 1978.

[20] R. M. Armitano, D. A. F. Florencio, and R. W. Schafer, "The motion transform: a new motion compensation technique," in *Proceedings of the Acoustics, Speech, and Signal Processing. 1996 IEEE International Conference - Volume 04*, ICASSP '96, (Washington, DC, USA), pp. 2295–2298, IEEE Computer Society, 1996.

[21] S. Klomp and J. Ostermann, *Motion Estimation at the Decoder*. InTech, Apr 2011.

[22] S. Jamkar, S. Belhe, S. Dravid, and M. S. Sutaone, "A comparison of block-matching search algorithms in motion estimation," in *Proceedings of the 15th International Conference on Computer Communication*, ICCC '02, pp. 730–739, International Council for Computer Communication, 2002.

[23] L. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol*, vol. 6, pp. 313–317, 1996.

[24] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol*, vol. 4, pp. 438–442, Aug 1994.

[25] S. Zhu and K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).