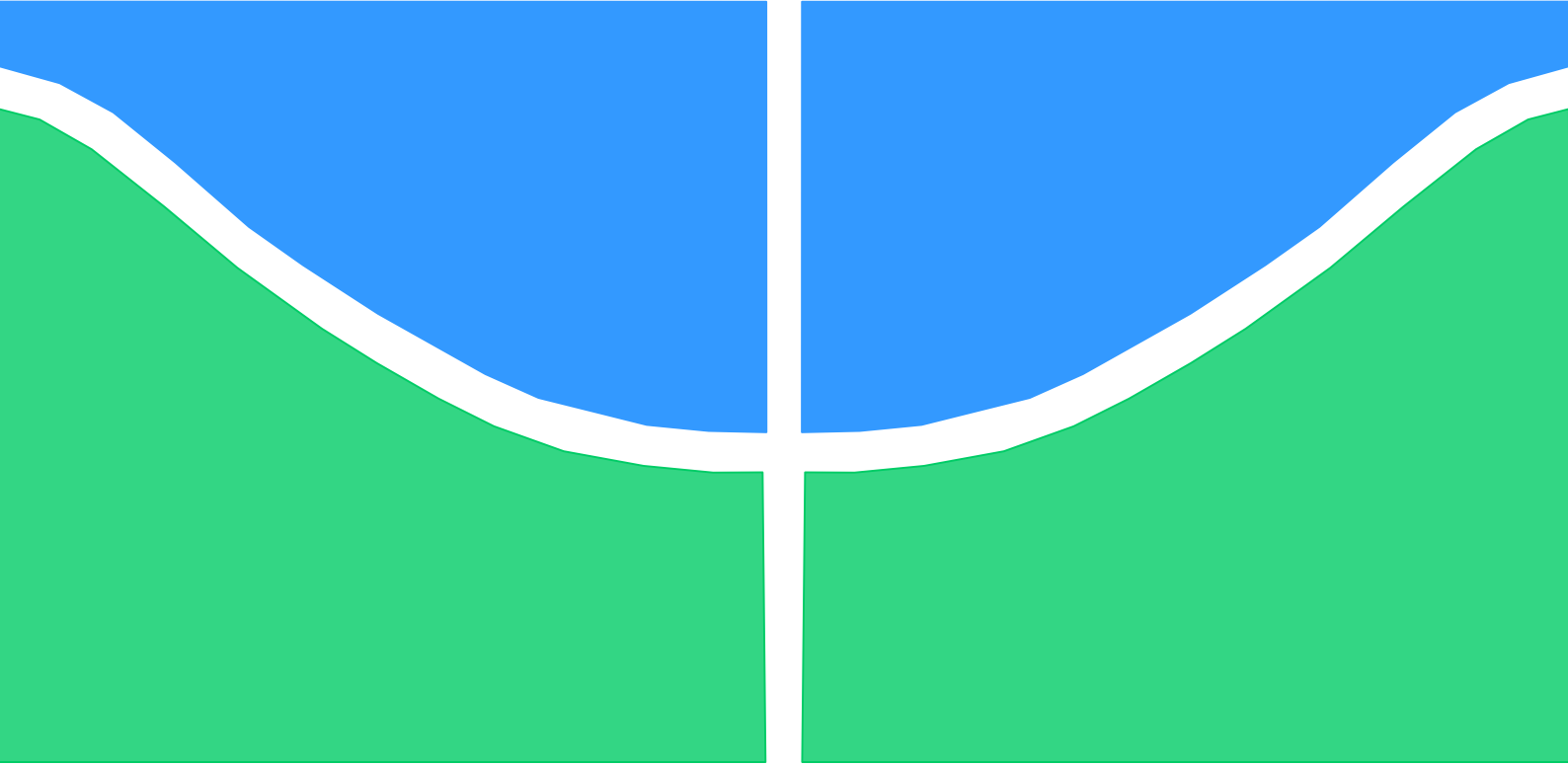


TRABALHO DE CONCLUSÃO DE CURSO

DETECÇÃO DE ADULTERAÇÕES NOS CANAIS DE ÁUDIO E VÍDEO DE MÍDIAS DIGITAIS UTILIZANDO MARCA D'ÁGUA

Brunno de Albuquerque Castro

Brasília, julho de 2016

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TRABALHO DE CONCLUSÃO DE CURSO

DETECÇÃO DE ADULTERAÇÕES NOS CANAIS DE ÁUDIO E VÍDEO DE MÍDIAS DIGITAIS UTILIZANDO MARCA D'ÁGUA

Brunno de Albuquerque Castro

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Eletricista

Banca Examinadora

Prof.ª Mylène C. Q. de Farias, ENE/UnB _____
Orientadora

Prof. Ricardo Zelenovsky, ENE/UnB _____
Examinador interno

Prof. Edson Mintsu Hung, ENE/UnB _____
Examinador interno

Agradecimentos

Gostaria de agradecer primeiramente à minha mãe, meu exemplo de dedicação, que me proporcionou todo o suporte e amor até esse momento.
À minha namorada, Gabriela, pelo carinho e apoio, principalmente nos momentos mais difíceis.
Ao meu grande amigo Luiz Henrique pela ajuda, sem a qual, eu não teria a oportunidade de fazer esse trabalho.
À professora Mylène Farias pela orientação, disponibilidade, boa vontade e dedicação.
Ao professor Marco Antônio Egito, que, na função de coordenador, me auxiliou no momento em que mais precisei.
Por fim, aos meus familiares e amigos pela compreensão durante todo o processo.

Brunno de Albuquerque Castro

RESUMO

É proposto um algoritmo para proteger tanto o conteúdo visual quanto os canais de áudio de vídeos digitais utilizando uma técnica de marca d'água por modulação do índice de quantização (QIM). Através dessa técnica, marcas espaciais e temporais são inseridas em ambos os canais de forma a detectar adulterações não autorizadas na mídia. Uma grande variedade de adulterações é suportada, cuja detecção é feita a partir da análise das marcas extraídas. Toda a análise é feita sem a referência do conteúdo original.

Palavras-chave: adulterações em vídeo, adulterações em áudio, marca d'água.

ABSTRACT

In this work, we propose an algorithm to protect both the visual content and the audio channels of a digital video using the watermarking technique of quantization index modulation (QIM). In the proposed technique, spatial and temporal marks are inserted into both channels in order to detect unauthorized tampering in the media. A wide variety of tamper attacks can be detected with the proposed technique, i.e. by performing an analysis of the extracted marks. The analysis is performed without needing the reference or the original content.

Keywords: video tampering, audio tampering, watermarking.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Mídias digitais desempenham um papel cada vez mais importante na atualidade. Com a disponibilidade cada vez maior de aparelhos eletrônicos portáteis que permitam aos usuários gravar e produzir suas próprias mídias, hoje praticamente tudo é registrado por meio de fotografia, vídeos e áudios. As mídias hoje servem como provas e documentação de fatos ocorridos da mesma forma que antigamente documentos escritos desempenhavam este papel. Ao contrário do texto, no entanto, as imagens proporcionam ao espectador uma absorção mais direta e imediata da mensagem, e essa provavelmente é a razão para que a fotografia tenha se tornado uma forma de registro e informação tão popular. Com o avanço da tecnologia, mídias mais complexas, como vídeos, vêm contribuindo ainda mais para a popularização de recursos visuais, especialmente após o advento de mídias digitais, uma vez que estas mídias são mais acessíveis e facilmente compartilhadas através dos diversos dispositivos aos quais temos acesso hoje [1].

A principal demanda por mídias audiovisuais é na indústria de marketing e entretenimento, seja para Internet, televisão, cinema ou música. No entanto, esta demanda também vem servindo como fonte importante para transmissão de informações, como no ensino e, principalmente, no jornalismo (profissional ou amador). Os acontecimentos em qualquer lugar do mundo são hoje transmitidos com enorme facilidade através da Internet, o que permite o compartilhamento de informações de usuário para usuário sem necessariamente passar por uma empresa de comunicação. Apesar de toda essa facilidade em fazer registros, com a criação de hardwares e softwares de interface cada vez mais simples mesmo para o usuário leigo, existe também uma facilidade enorme em adulterar estas mídias de forma a distorcer o conteúdo transmitido, comprometendo a sua autenticidade. Inicialmente estas alterações eram feitas para aprimorar a qualidade das imagens, melhorando aspectos que não haviam ficado como desejados durante a captura da imagem, sem, entretanto, alterar a informação transmitida. Com o passar do tempo, modificações fraudulentas passaram a ser feitas de forma a adulterar deliberadamente a mensagem transmitida.

Nem sempre é fácil para um olhar não treinado detectar essas adulterações, e, para tal, diversos métodos computacionais vêm sendo desenvolvidos nos últimos anos para detectar possíveis fraudes. *Lin* desenvolve técnicas baseadas em marca d'água para solucionar problemas na sincronização de vídeos causadas por diferentes tipos de adulterações [2]. *Wang* busca identificar características próprias de cada vídeo que possam ser quantificadas, medidas e usadas para detectar adulterações no vídeo [3]. *Cross* e *Mobasseri* desenvolvem um algoritmo para detecção de adulterações em vídeos comprimidos com codificação MPEG [4]. *Hou* e *Tang* utilizam marcas frágeis para autenticar a integridade do vídeo sem a comparação com o vídeo original [5], enquanto *Chen* e *Zhao* utilizam marcas semi-frágeis na construção de um algoritmo de autenticação de conteúdo [6].

As técnicas presentes na literatura para detecção de ataques a vídeos, até então, costumavam focar em apenas um tipo de adulteração, normalmente adulterações locais. Ainda são necessárias ferramentas que identifiquem de forma mais eficiente vários tipos de adulterações

simultaneamente, tanto no domínio espacial quanto no domínio temporal. Ronaldo Rigoni, em sua dissertação de mestrado de 2013 [7], propõe um algoritmo de proteção a vídeos utilizando marca d'água com base na modulação por índice quantizado (QIM) [8]. O QIM é um algoritmo simples, capaz de detectar um grande número de adulterações, em comparação com outras técnicas descritas na literatura. O sistema proposto por Ronaldo Rigoni consiste na inserção simultânea de uma marca espacial e uma marca temporal no canal de vídeo e uma replicação da marca temporal no canal de áudio. A replicação da marca no canal de áudio tem como objetivo criar uma redundância permitindo que a marca temporal seja identificada mesmo quando o sinal de vídeo sofre degradações. Este processo concede uma maior robustez à proteção da componente de vídeo, porém não protege a componente de áudio.

O presente trabalho busca estender o trabalho de Rigoni de forma a prover ao áudio a mesma proteção dada à componente de vídeo. Assim, o algoritmo desenvolvido neste trabalho se baseia, em grande parte, no algoritmo proposto por Rigoni, sendo que as principais diferenças consistem na utilização de marcas d'água nos canais de áudio. Com isso, é possível proteger não somente o conteúdo do vídeo, mas também do áudio.

Este documento está dividido da seguinte forma. No Capítulo 2 é apresentada um breve resumo dos métodos empregados para adulteração de mídias audiovisuais e seus respectivos exemplos. No Capítulo 3 é explicado o conceito de marca d'água e suas possíveis aplicações práticas. No Capítulo 4, são descritos o algoritmo proposto e a metodologia empregada. No Capítulo 5, são discutidos os resultados observados após aplicação do algoritmo. No Capítulo 6 são colocadas as conclusões e possibilidades de trabalhos futuros.

2 ADULTERAÇÃO DE MÍDIAS DIGITAIS

A adulteração de mídias (*tampering*), maliciosa ou não, é extremamente comum em imagens e vídeos distribuídos digitalmente. As adulterações vão desde ajustes de imagens para fins de entretenimento ou publicidade, grandes montagens na indústria cinematográfica, ou até alterações fraudulentas em fotografias, vídeos e áudios, que comprometem a integridade da informação. No entanto, a adulteração de imagens não é algo recente. Muito antes da disseminação das mídias digitais este tipo de trabalho já era feito. Abaixo, são citados alguns exemplos de imagens históricas manipuladas para fins políticos.

Figura 2.1: Foto adulterada à esquerda com primeiro ministro do Canadá e a Rainha Elizabeth. Foto original à direita com Rei George VI (Fonte: *www.fourandsix.com*).

As imagens da Figura 2.1 são datadas de 1939. Na imagem à direita estão o então primeiro ministro do Canadá, William Lyon Mackenzie King, a Rainha Elizabeth e o Rei George VI no terraço de um hotel em Banff, Alberta. Posteriormente esta imagem foi adulterada (ver imagem à esquerda), retirando o Rei do quadro. Esta imagem adulterada foi usada como *poster* na campanha política do primeiro ministro. Acredita-se que esta alteração foi feita, porque se acreditava que uma imagem contendo apenas o ministro e a Rainha passaria uma impressão de mais poder.

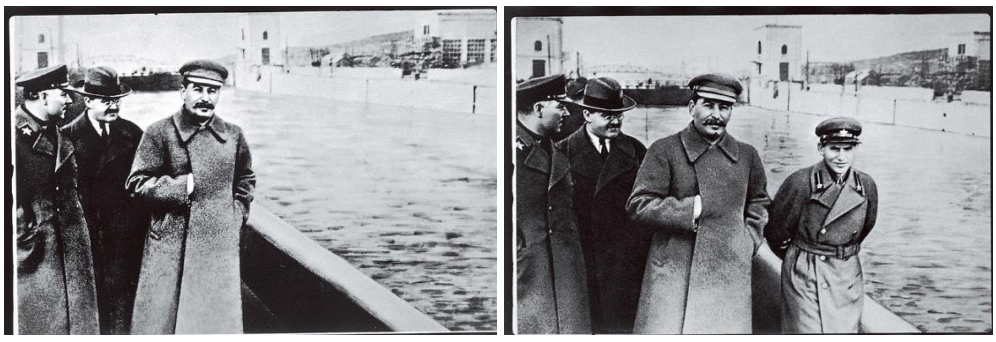


Figura 2.2: Foto original à direita. Foto adulterada à esquerda com a remoção comissário Nikolai Yezhov (Fonte: *www.fourandsix.com*).

A imagem à direita da Figura 2.2 é uma fotografia onde, originalmente, Stalin aparecia ao lado do comissário Nikolai Yezhov, o qual acabou se tornando um inimigo de Stalin, preso e executado. O governo soviético nesta época tinha o hábito de alterar fotografias para fins de censura. Desta forma, na imagem à esquerda da Figura 2.2, o original foi adulterado removendo o comissário por causa de interesses políticos na ocasião.

Atualmente, as ferramentas para adulteração estão cada vez mais sofisticadas. Desta forma, também são necessárias novas ferramentas para identificar tais alterações. Neste capítulo, são descritos vários exemplos de alterações em imagens, vídeos e áudios.

2.1 Adulteração em imagens

Dentre os diversos tipos de adulterações (ou ataques) que podem ser feitos em imagens, podemos citar a composição, o copiar-e-colar, borramentos, alteração do padrão de cores, entre outros.

- Composição: Em uma composição, duas ou mais imagens são sobrepostas para formar uma nova imagem que combine elementos das imagens originais. Para que isto seja realizado de forma não perceptível, as imagens devem ser rotacionadas, redimensionadas, ou terem apagadas as partes que apresentem heterogenicidade entre os limites de uma imagem e outra. Por melhor que seja este trabalho, o resultado final pode deixar vestígios, uma vez que as bordas da composição nunca serão completamente homogêneas, o que permite a sua detecção.

Figura 2.3: Exemplo de composição de imagens (Fonte: *www.zevendesign.com*).

Na Figura 2.3 é mostrado um exemplo de composição onde 5 imagens distintas são utilizadas para gerar uma composição destas imagens. Com o objetivo de deixar a composição mais uniforme, as imagens podem ser redimensionadas, rotacionadas ou terem características alteradas, como luminosidade e padrão de cor.

- Copiar e Colar: Nesta adulteração, um segmento da imagem é copiado e então colado em outro local da própria imagem. Isto permite esconder partes indesejadas na imagem ou duplicar elementos. Os métodos de detecção para estes casos consistem em buscar áreas de

grande similaridade. Este tipo de detecção é difícil, pois estas replicações podem ser de qualquer tamanho ou se repetir diversas vezes, em vários locais da imagem.

Figura 2.4: Exemplo de adulteração do tipo Copiar e Colar
(Fonte: *thelede.blogs.nytimes.com*).

As imagens da Figura 2.4 mostram um exemplo da adulteração Copiar e Colar. A imagem original (à esquerda) retrata um lançamento de mísseis no Irã, em 2008. Já a imagem à direita é uma adulteração do original que teve fins políticos, com objetivos de exagerar a situação observada na foto original. Observe que as áreas circuladas na imagem são idênticas.

O fotojornalismo vem sendo foco de muitas fraudes nos últimos anos, como observado no exemplo apresentado nas imagens da Figura 2.5. Neste caso, o fotógrafo *freelancer* Narciso Contreras copiou elementos presentes na imagem para esconder uma câmera no canto inferior esquerdo, possivelmente por motivos meramente estéticos. A adulteração foi descoberta e o fotógrafo foi desligado da agência Associated Press, que distribuiu a foto.

Figura 2.5: Foto original acima. Foto adulterada por Copiar e Colar abaixo
(Fonte: *www.ap.org*).

- Adulterações Locais: Neste tipo de adulteração, pequenas modificações são realizadas em um segmento específico da imagem, como, por exemplo, recortes, distorções e borramentos. Este tipo de modificação é particularmente comum na mídia ou para fins de entretenimento. Um exemplo muito disseminado no cotidiano é a adulteração de fotos destinadas à publicidade, onde modelos ou objetos são modificados para parecer mais atraentes para o consumidor.

Figura 2.6: Adulteração comum feita pela indústria da moda [9].

Nas imagens da Figura 2.6 é apresentado um exemplo de adulterações locais. A imagem à esquerda é a original que contém uma modelo, enquanto que a imagem à direita é a imagem adulterada, na qual várias alterações localizadas foram feitas para que a modelo parecesse mais magra e, desta forma, atendesse aos padrões estéticos exigidos pela indústria da moda. Este tipo de indústria, assim como as agências de publicidade, utiliza-se muito destas adulterações, por vezes realizadas sem o conhecimento ou autorização da modelo retratada.

2.2 Adulteração de vídeos

As adulterações em vídeos tendem a ser mais complexas do que as alterações feitas em fotografias, pois, além das alterações locais (espaciais), são feitas adulterações temporais, de mais difícil detecção. Existem várias aplicações práticas para a manipulação de vídeos, por exemplo na indústria cinematográfica. Mas também existem manipulações fraudulentas que visam modificar a mensagem passada pelo vídeo, manipular o entendimento do telespectador, ou simplesmente burlar os mecanismos de segurança que os protegem contra cópias indevidas. Abaixo são citadas algumas modalidades de manipulação destas mídias [7] [2].

- Adulterações locais: São feitas de forma semelhante às adulterações de imagens. São feitas alterações localizadas em um ou mais quadros do vídeo. Todas as adulterações citadas acima (composição, copiar e colar) são também aplicáveis a vídeos.

- Adulterações globais: Modificações que alteram a imagem como um todo. Entre os exemplos, o ajuste de brilho, contraste, saturação. Estas modificações também são aplicáveis a imagens. Outros exemplos de adulterações globais em vídeos são a modificação de seu formato ou dimensões.
- Adulterações temporais ou de sincronização: Quando se fala em adulteração de vídeos, este é um dos principais métodos empregados e representa também um grande desafio para sua detecção. Diferente das alterações descritas até aqui, que modificam uma imagem estática ou quadros isolados dentro de um vídeo, esta modalidade altera a disposição dos quadros em um vídeo ao longo do tempo, retirando, acrescentando ou mudando a ordem destes. A grande dificuldade encontrada para detecção destas adulterações é que, uma vez alterada a linha temporal do vídeo, a presença da marca d'água pode não ser percebida pelo detector, sendo essa uma das maiores vulnerabilidades da proteção de mídias por meio de sua inserção. Para que esta dificuldade seja contornada, a marca d'água deve apresentar redundância o suficiente nos demais quadros para que, mesmo que a ordem seja alterada, ela ainda continue sendo encontrada pelo detector. A seguir são citadas algumas formas de fazer este tipo de adulteração.
- Duplicação de quadros: Quadros são copiados e colados subsequentemente dentro de um vídeo, muitas vezes de forma imperceptível ao olho humano, porém permite confundir os métodos de detecção da marca d'água devido ao aumento do número de quadros por segundo e comprometem a sincronização perfeita da marca.
- Embaralhamento de quadros: Este tipo de adulteração altera os quadros de posição de forma não perceptível ao expectador, porém também pode levar a uma dessincronização da marca d'agua.
- Remoção de quadros: alguns quadros são removidos e com isso partes da marca ficam ausentes, dificultando sua detecção.
- Inserção de quadros: são inseridos novos quadros, geralmente feitos por meio de composição ou retirados de outras fontes, comprometendo a ordem dos quadros. No entanto, este tipo de adulteração costuma ser identificado mais facilmente, uma vez que, normalmente, os quadros inseridos não estão marcados.

2.3 Adulteração de áudio

Os estudos sobre adulterações em áudio, em comparação com aqueles sobre ataques a imagens ou vídeos, ainda são muito escassos. Os ataques a mídias de áudio podem ser feitos para fins de pirataria ou falsificação de evidências (como gravações telefônicas ou escutas). Verificar a autenticidade de áudios nestes casos é de extrema importância e é um dos focos deste trabalho. Alguns exemplos de como estas mídias podem ser adulteradas são descritos abaixo [10].

- Adição de ruídos: Este tipo de adulteração pode ser feita de forma intencional ou não. É usada, por exemplo, quando se quer deliberadamente tornar o áudio mais difícil de se decifrar. Porém, um ruído pode também aparecer de forma não intencional como

consequência de uma gravação amadora ou pela interferência do meio, comprometendo a qualidade do original.

- Deleção ou Anulação: Um segmento do áudio é deletado, o que permite tanto esconder imperfeições pontuais em uma mídia como esconder de forma proposital uma parte que não se queira mostrar para o ouvinte.
- Composição: Dois áudios são somados de forma a alterar o conteúdo original. Isto é feito, por exemplo, na criação de músicas eletrônicas. Também pode ser uma maneira de fraudar um áudio inserindo algo que não estivesse originalmente no áudio.
- Distorção: É uma das formas mais usadas de alteração em áudios; o espectro do áudio original pode ser alterado, modificando seu timbre ou frequência, permitindo uma enorme quantidade de modificações nas mídias auditivas.

3 MARCA D'ÁGUA

A técnica de marca d'água (em inglês, *watermarking*) consiste em adicionar informações aos dados de uma mídia, geralmente com o objetivo de proteger o conteúdo desta e os direitos autorais de seu criador.

Como não são parte do conteúdo original, marcas d'água constituem em alterações do sinal hospedeiro. A maioria dos sistemas de marcação busca fazer com que essas alterações sejam imperceptíveis.

A inserção de uma marca d'água imperceptível é feita através de técnicas que aplicam pequenas modificações (ligadas à informação da marca) nos dados a serem marcados. Essas modificações são sutis a ponto de o usuário não as perceber. Posteriormente, as informações inseridas podem ser recuperadas por técnicas que detectam essas modificações.

A existência de mais de uma possibilidade de se escrever certo conjunto de dados, cria-se a oportunidade de se inserir uma marca baseada na escolha dessas opções. Quanto maior as possibilidades de escrita, maior a quantidade de informações que a marca pode ter.

A inserção e extração da marca podem ocorrer não apenas no domínio espacial, mas também nos domínios de transformadas, como a de cosseno discreto, Fourier ou wavelet [11] [12]. Cada uma dessas possui propriedades que são exploradas nas diferentes técnicas de marcação.

Neste capítulo, são apresentadas as aplicações, bem como as classificações de marcas d'água encontradas na literatura. É descrito o modelo básico de funcionamento de uma marca d'água, e explicado o algoritmo de modulação por índice quantizado, técnica utilizada neste trabalho.

3.1 Aplicações de Marca d'Água

Marca d'água possui diferentes aplicações, geralmente com a finalidade de aumentar a segurança ou controle de mídias digitais. Listamos a seguir algumas delas:

- Direitos autorais: As informações de direitos autorais podem ser inseridas no sinal através de uma marca d'água visível ou secreta. Isso faz com que o dono possa provar, numa eventual disputa, ser o possuidor dos direitos daquele conteúdo.
- Autenticação: Uma marca d'água pode ser sensível a adulterações do sinal. Com isso, ela pode ser usada para verificar adulterações na mídia e, consequentemente, sua autenticidade.
- Monitoramento de transmissão: A segurança na transmissão de conteúdo é sempre tema para empresas que se preocupam com a violação de seus direitos. Marcas d'água podem ser utilizadas, por exemplo, em sistemas automáticos que verificam se anúncios estão sendo transmitidos conforme contratado [13].
- Controle sobre cópia: Dispositivos eletrônicos podem identificar em mídias se há informações que definem a proteção contra a cópia ou armazenamento daquele conteúdo. Basta um bit de informação para indicar se é permitida ou não a cópia [12].

- Impressão digital: Um sistema de impressão digital insere uma identificação secreta em cada uma das cópias distribuídas legalmente a seus consumidores. Com isso, se cópias ilegais forem encontradas, é possível extrair essa informação e rastrear o consumidor que fez a distribuição.

3.2 Classificações de Marca d'Água

Vários conceitos são atribuídos aos diferentes tipos de marca d'água. Geralmente são requerimentos demandados pelas variadas aplicações que marcas d'águas possuem. Alguns desses conceitos são utilizados para classificar as técnicas de marcação existentes. Dentre as classificações utilizadas pela literatura, destacamos as seguintes [12] [14] [15] [16]:

- Visível: Uma marca visível é a aquela perceptível ao usuário do conteúdo. É feita para deixar claro a identidade de quem possui os direitos sobre a mídia marcada. Um exemplo comum são as logomarcas.
- Invisível: É aquela imperceptível sob condições normais de visualização. É uma característica importante para aplicações que buscam não afetar a experiência perceptiva dos usuários, mantendo a qualidade do sinal. A maioria das pesquisas focam nesse tipo de marca.
- Robusta: A robustez de uma marca é a medida de quão difícil é degradá-la sem prejudicar significativamente o sinal hospedeiro. Uma marca robusta se mantém identificável mesmo após processamentos comuns (como compressão com perda, filtragem, reamostragem, mudança de formato), que, em geral, degradam marcas d'água. Segundo Langelaar *et al.*, em geral, uma marca é mais robusta quanto maior a profundidade da técnica de inserção e maior a quantidade de bits de informação da marca [12].
- Frágil: Diferente de marcas robustas, marcas frágeis são sensíveis a processos que modificam, mesmo que de forma sutil, os dados da mídia hospedeira. Por isso, são utilizadas com outros objetivos, dentre os quais, temos a confirmação de autenticidade e integridade da mídia. Ou seja, qualquer alteração, por mínima que seja, deve ser percebida por uma marca frágil. Uma falha na sua extração indicaria que a mídia foi adulterada.
- Semi-frágil: Marcas d'água desse tipo são menos sensíveis a adulterações que marcas frágeis. Elas são usadas para detectar alterações mais significativas. Segundo Abdullatif *et al.*, uma marca desse tipo é robusta contra modificações casuais, porém frágil contra ataques maliciosos [16].

3.3 Modelo Básico de Marca d'Água

O modelo básico de funcionamento de uma marca d'água digital é dividido em dois subsistemas: inserção e extração.

O processo de inserção, ilustrado a seguir, tem por entradas o sinal original a ser marcado e as informações da marca d'água a serem inseridas nesse sinal. A chave secreta é uma entrada opcional que aumenta a segurança do processo. O sinal marcado é então gerado através do algoritmo de inserção da marca.

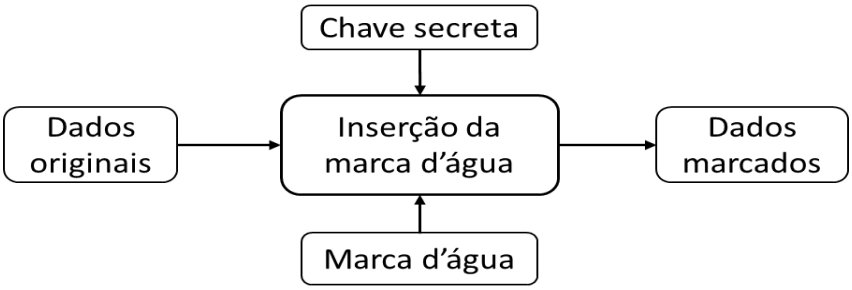


Figura 3.1: Inserção da marca d'água.

No processo de extração, ilustrado abaixo, o sinal marcado é utilizado em um algoritmo que irá extrair a marca possivelmente adulterada. Se a chave secreta foi utilizada na inserção da marca, ela deve ser utilizada também na extração da marca. Dependendo da técnica, o sinal original pode ou não ser utilizado na extração da marca. Se o sinal original não é utilizado, a detecção é chamada de cega ou sem referência, caso contrário, é chamada de com referência.

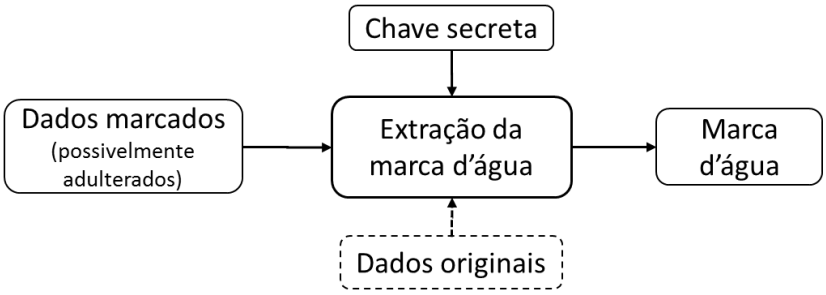


Figura 3.2: Extração da marca d'água.

Se a marca utilizada for do tipo frágil, ou semi-frágil, a marca extraída é, então, comparada com a original para verificar se houve alguma adulteração e confirmar ou não a autenticidade do sinal.

3.4 Modulação por Índice Quantizado

O algoritmo de modulação por índice quantizado (QIM, do inglês *Quantization Index Modulation*) foi criado em 2001 por Chen *et al* [8]. De acordo com Seo *et al*, o algoritmo QIM é um método de inserção de marcas d'água que utiliza técnicas de quantização, nas quais a mensagem da marca é índice para a escolha do quantizador dentre um conjunto de possíveis quantizadores [17]. Para realizar essa quantização, é escolhido um valor adequado para o degrau de quantização (Δ) de forma a obter uma boa capacidade de inserção sem prejudicar

substancialmente a qualidade da imagem. O quantizador utilizado pelo QIM obedece à seguinte equação:

$$Q(x_i, \Delta) = \left\lfloor \frac{x_i}{\Delta} \right\rfloor \Delta, \tag{3.1}$$

na qual $\lfloor \cdot \rfloor$ é a operação de arredondamento em direção ao menor inteiro (conhecida como chão, ou *floor*). A operação Q é aplicada a cada amostra do sinal x_i.

Em seguida, o sinal marcado é gerado a partir da seguinte expressão geral:

$$s(x_i, m) = Q(x_i, \Delta) + d(m), \tag{3.2}$$

na qual $d(m)$ é uma função de modulação de um bit.

A extração é feita utilizando a seguinte equação:

$$\hat{m} = \hat{x} \; mod \; \Delta, \tag{3.3}$$

na qual *mod* corresponde à operação definida simplificadamente como resto da divisão inteira, $\hat{m}$ é a marca extraída e $\hat{x}$ o sinal possivelmente adulterado.

O QIM foi o método escolhido por Rigoni em sua dissertação de mestrado. De acordo com ele, a escolha foi feita por este método possuir maior capacidade de inserção, menor distorção e maior capacidade de localização da marca inserida, quando comparado aos outros presentes na literatura. Além disso, o QIM permite detectar adulterações com granularidade de 1 pixel mantendo a alta taxa de inserção [7].

4 ALGORITMO PROPOSTO

O algoritmo desenvolvido nesse trabalho é, em grande parte, baseado no algoritmo proposto por Ronaldo Rigoni, em sua dissertação de mestrado (Ronaldo Rigoni, 2013). Em sua dissertação, Ronaldo propõe um algoritmo de proteção de vídeo utilizando uma técnica de marca d'água, que é baseada na operação de modulação por índice quantizado (QIM), que foi descrita no Capítulo 3.

Ronaldo descreve seu o algoritmo como simples e robusto, possuindo o diferencial de detectar uma maior gama de adulterações em comparação com a maioria das técnicas presentes na literatura, as quais costumam se limitar a aplicações mais específicas.

O algoritmo de Ronaldo, consiste na inserção simultânea de uma marca espacial e uma marca temporal no canal de vídeo e uma replicação da marca temporal no canal de áudio. A replicação no canal de áudio busca dar maior robustez à proteção do vídeo, criando uma redundância no canal de áudio como uma alternativa à possível degradação da marca temporal no canal de vídeo.

Neste trabalho, além de testar o algoritmo desenvolvido por Ronaldo, procuramos explorar mais a utilização de marcas d'água nos canais de áudio de um vídeo digital. Em síntese, além das marcas descritas no parágrafo anterior, inserimos nos canais de áudio uma marca semelhante à marca espacial utilizada no canal de vídeo. Com isso, buscamos, além da proteção do vídeo, a proteção do áudio.

Nas sessões seguintes, detalhamos o funcionamento do algoritmo desenvolvido neste trabalho. Ele é dividido em duas partes principais. A primeira parte consiste na proteção dos canais de áudio e vídeo, através da geração e inserção das marcas d'água. A segunda parte consiste na detecção de adulterações de tipos variados através da extração e análise das marcas inseridas.

4.1 Proteção dos Canais de Áudio e Vídeo

A proteção contra adulterações tanto na parte visual quanto auditiva de um arquivo de vídeo digital é feita através da inserção de uma marca d'água em cada um desses conteúdos. O esquemático a seguir representa de modo simplificado esse processo.

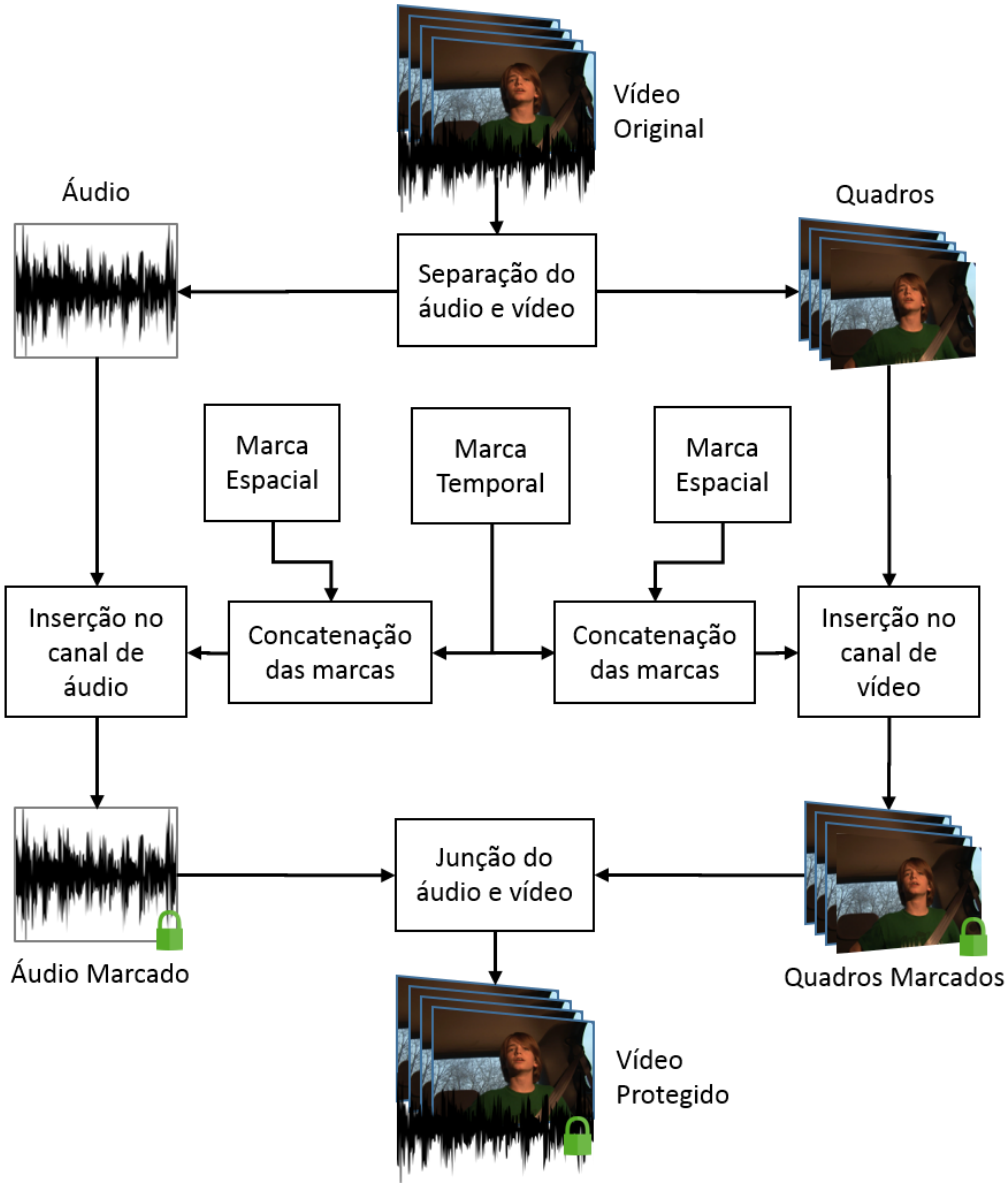


Figura 4.1: Diagrama do processo de proteção dos vídeos.

Nas sessões a seguir, detalhamos a geração e a inserção das marcas nos sinais de vídeo e áudio.

4.1.1 Geração das Marcas d'Água

A geração das marcas d'água é feita a partir de um gerador de números pseudoaleatórios e uma chave secreta, a qual apenas os detentores dos direitos autorais possuem. Essa chave é uma semente (*seed*) que serve para inicializar o gerador de números pseudoaleatórios. Assim, só quem possui essa chave pode gerar a mesma sequência de números da marca.

O vídeo a ser marcado é separado em suas componentes visuais (quadros) e suas componentes auditivas (áudio). Para os quadros, é gerada uma marca espacial e uma temporal. A espacial é usada para proteger o conteúdo espacial de cada quadro do vídeo. Para não haver persistência visual, a marcas são diferentes para cada quadro. Elas possuem as mesmas dimensões dos quadros e uma profundidade de 5 bits. Ou seja, para cada pixel do vídeo, há uma marca de 5 bits. Isso confere uma proteção de granularidade de um pixel.

Para a marca temporal, é gerado um número aleatório Φ que identifique cada quadro do vídeo. Esses números possuem tamanho de 16 bits. Para formar a marca temporal, essa sequência de bits é repetida até preencher as dimensões do quadro. Tendo assim, um bit de marca para cada pixel.

Para o áudio, também é gerada marca espacial e uma temporal. Não é comum utilizar-se do termo "espacial" para uma marca no áudio, visto que este não possui dimensões espaciais como os quadros de um vídeo, porém, por falta de um termo melhor que a identifique, assim será chamada ao longo desse trabalho.

A marca espacial no áudio é uma extensão do que é feito com o vídeo. É uma proposta desse trabalho, e visa conceder uma proteção a cada amostra do áudio. É gerada uma marca para cada amostra, com uma profundidade de bits que depende das características do áudio. Neste trabalho, utilizamos para um som estéreo (dois canais) de 16 bits por amostra uma marca de 3 bits de profundidade.

Para a marca temporal do áudio, os mesmos números usados para identificar os quadros do vídeo são usados para identificar os trechos de áudio equivalentes. Esses números também são sequência de 16 bits que são repetidas até, nesse caso, haver um bit para cada amostra do trecho.

4.1.2 Inserção das Marcas d'Água

Para cada quadro do vídeo, tem-se então uma matriz com a marca espacial M_s com as mesmas dimensões do quadro e profundidade de 5 bits e uma matriz com a marca temporal M_t, também com as mesmas dimensões, mas com profundidade de 1 bit. Essas marcas são inseridas em vídeos do tipo RGB24, que possuem 8 bits de informação para cada uma das três cores que formam o sistema RGB (*Red*, *Green* e *Blue*).

Para realizar a inserção das marcas nos quadros, é necessário primeiro concatená-las, ou seja, uni-las. Para isso, cada unidade da marca temporal $M_t[x, y]$ é inserida em uma posição aleatória de uma unidade da marca espacial $M_s[x, y]$. Assim, uma marca de 6 bits é formada a partir dos 5 bits da $M_s[x, y]$ mais 1 bit da $M_t[x, y]$, sendo que esse bit temporal tem uma posição aleatória. Essa posição aleatória do bit temporal é produzida pela mesma chave secreta que gerou as marcas, ou seja, por um gerador de números pseudoaleatórios. O processo de concatenação das marcas é ilustrado na Figura 4.2.

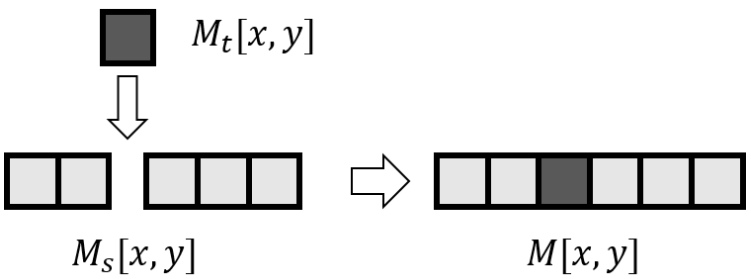


Figura 4.2: Concatenação das marcas binária temporal e espacial do vídeo.

Como resultado da união das duas marcas, tem-se uma marca concatenada M. Cada unidade $M[x, y]$ de 6 bits é, então, dividida em três partes de 2 bits. Essas três partes são convertidas em valores decimais, $M_R[x, y]$, $M_G[x, y]$ e $M_B[x, y]$, para serem inseridos no sinal, utilizando a técnica QIM. Mais especificamente, estas marcas serão inseridas em cada uma das três componentes de cor (RGB) do pixel. A Figura 4.3 ilustra a separação da marca nas três componentes de cor.

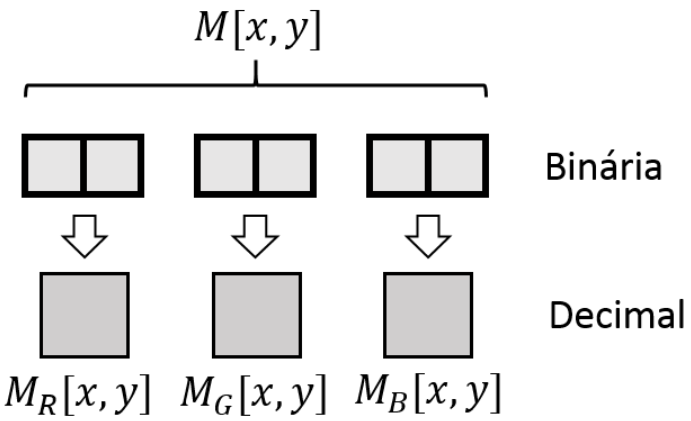


Figura 4.3: Separação da marca do vídeo para as três componentes de cor.

Neste algoritmo, é utilizada uma forma modificada da técnica QIM. Nessa modificação, a função de modulação é uma função identidade, ou seja, $d(m) = m$ na Equação (3.2). Essa foi a tática pensada por Ronaldo Rigoni para aumentar a capacidade de inserção, permitindo inserir diretamente um número inteiro, ou seja, uma maior quantidade de bits.

Assim, para cada componente de cor do quadro, a marca correspondente é inserida de acordo com a seguinte expressão:

$$F_m[x, y, cor] = Q(F[x, y, cor], \Delta) + M_{cor}[x, y], \tag{4.1}$$

na qual, F é o quadro (do inglês, *frame*), F_m é o quadro marcado, cor é uma das três cores (R, G ou B), M_{cor} é a marca decimal correspondente àquela cor e $[x, y]$ representam a posição do pixel no quadro. A função $Q(\cdot)$ quantiza os valores originais por um fator Δ, que neste trabalho tem valor de $\Delta = 4$, suficiente para inserir as marcas decimais que variam de 0 a 3 (equivalente a 2 bits).

Finalizada a proteção dos quadros, a próxima etapa é a marcação do áudio. O procedimento é feito de forma semelhante ao adotado com os quadros. Para isso, subdivide-se o áudio em trechos cujo tempo total equivalha ao tempo dos respectivos quadros do vídeo. O tempo de um quadro é definido pelo inverso da taxa de quadros do vídeo (*framerate*). Da mesma forma como os quadros possuem informações divididas em diferentes cores (RGB), um trecho de áudio possui diferentes canais. Os mais comuns são áudios estéreos, que são compostos por dois canais (esquerdo e direito), e serão o tipo utilizado neste trabalho.

Desta forma, para cada trecho de áudio há uma marca temporal, criada com a mesma sequência de 16 bits que identifica o quadro equivalente. Essa sequência é repetida até haver um bit para cada amostra. Também, para cada amostra, há uma marca espacial de 3 bits, a qual é concatenada com o bit temporal. Como nos quadros, o bit temporal é inserido em uma posição aleatória com relação aos bits da marca espacial. A concatenação do bit temporal com os bits espaciais é ilustrada na Figura 4.4.

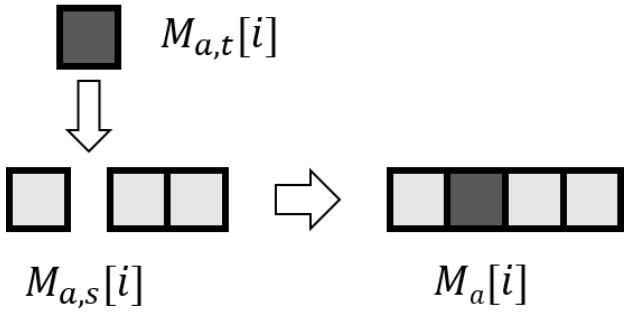


Figura 4.4: Concatenação das marcas binárias temporal e espacial no áudio.

A marca binária do áudio M_a resultante da concatenação é dividida em duas partes, uma para cada canal de áudio (esquerdo e direito), com 2 bits cada, e, em seguida, são convertidas para valores decimais, M_e e M_d. A Figura 4.5 ilustra a separação das marcas para cada canal.

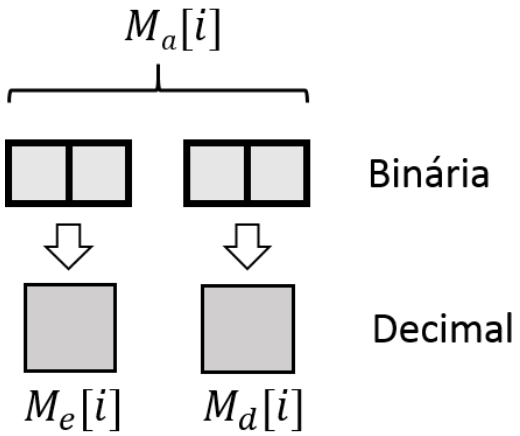


Figura 4.5: Separação da marca do áudio para os canais esquerdo e direito.

As amostras de um áudio decodificado são valores que variam de -1 a 1. Esses valores possuem níveis de quantização baseados na quantidade de bits usados para representá-las (*bit depth*). A Equação (4.2) expressa essa relação.

$$step = \frac{2}{2^{bitdepth}}, \tag{4.2}$$

na qual $step$ representa o valor entre dois níveis de quantização, ou seja, o degrau de quantização, que é definido na Equação (4.2) como a divisão do comprimento da faixa de valores que as amostras de áudio podem assumir (de -1 a 1) pela quantidade de níveis de quantização possíveis para representar essas amostras $2^{bitdepth}$. Para um áudio de *bit depth* = 16 bits, são $2^{16} = 65536$ níveis de quantização, com espaçamento de $step = 3{,}05 \cdot 10^{-5}$.

Em seguida, a inserção da marca d'água para cada canal de áudio é efetuada de acordo com a seguinte expressão:

$$T_m[i, canal] = Q(T[i, canal], \Delta) + step \cdot M_{canal}[i], \tag{4.3}$$

na qual, T é o trecho do áudio original, T_m é o trecho do áudio marcado, $canal$ é um dos dois canais (direito ou esquerdo) do áudio, M_{canal} é a marca correspondente àquele canal e i representa a posição da amostra no trecho. A função $Q(\cdot)$ quantiza os valores originais por um fator $\Delta = 4 \cdot step$, suficiente para inserir as marcas que variam de 0 a 3 (equivalente a 2 bits). Nessa utilização do QIM, a função de modulação tem forma $d(m) = step \cdot m$.

4.2 Detecção de Adulterações

A detecção de adulterações é feita a partir da extração e análise da marca d'água da mídia possivelmente adulterada. O diagrama de blocos deste processo é apresentado na Figura 4.6 e detalhado nas seções seguintes.

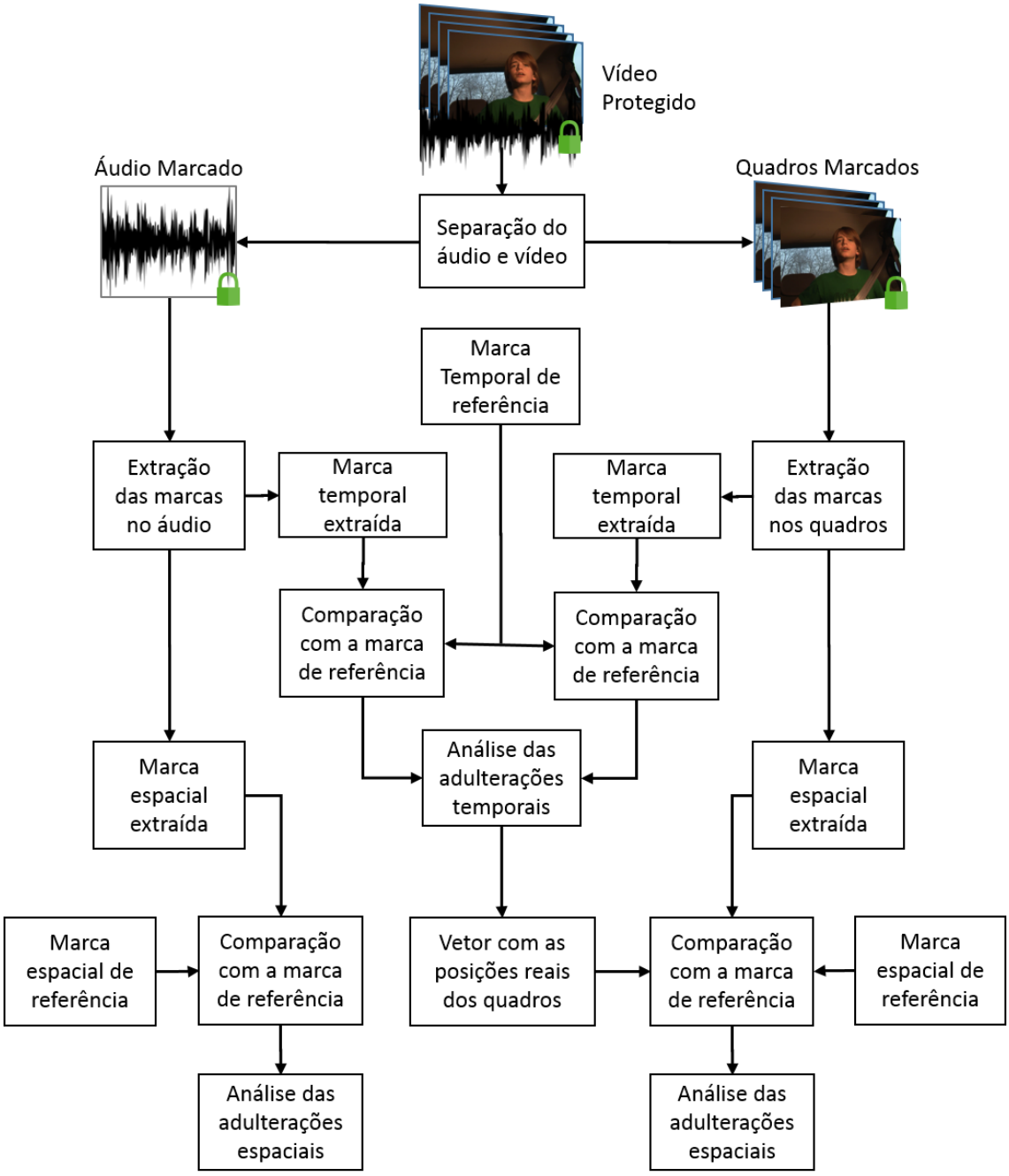


Figura 4.6: Diagrama do processo de detecção de adulterações em vídeo e áudio.

4.2.1 Extração das Marcas d'Água

O vídeo previamente marcado e possivelmente adulterado tem seus conteúdos visual (quadros) e sonoro (áudio) separados, para extração da marca de cada um. Para cada quadro, a marca é extraída implementando-se a seguinte expressão.

$$\widehat{M}_{cor}[x, y] = F_m[x, y, cor] \; mod \; \Delta, \tag{4.4}$$

na qual $\widehat{M}_{cor}$é a marca extraída de uma das três cores, F_m é o quadro (frame) marcado e mod é a operação que retorna o resto da divisão entre o valor do pixel e Δ (que neste trabalho possui valor igual a 4). O valor da marca para cada cor que em decimal varia de 0 a 3 é convertido

para binário (2 bits) e, então, as três são concatenadas (justapostas), formando uma marca $\widehat{M}_v$ de 6 bits. A concatenação é ilustrada na Figura 4.7.

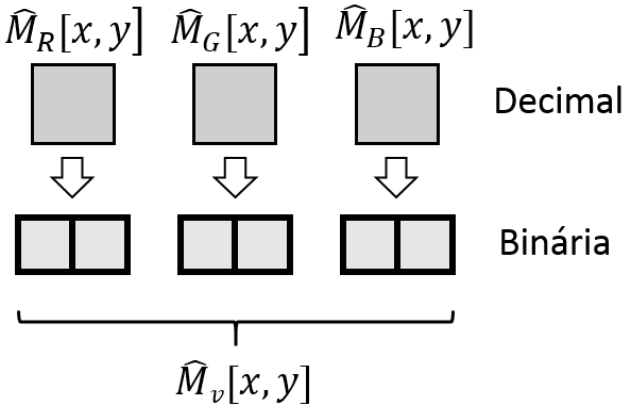


Figura 4.7: Concatenação da marca extraída do vídeo.

Em seguida, é realizada a separação das marcas temporal e espacial. Para isso, é necessário utilizar a mesma chave secreta que foi utilizada na proteção do vídeo. Com essa chave, é gerada uma matriz com as posições do bit temporal para cada pixel do quadro. Com essa informação, sabe-se dos 6 bits qual é o bit temporal e quais são os 5 bits espaciais. Separando-os, tem-se então as marcas binárias.

Por fim, a marca espacial $\widehat{M}_{v,s}$ é obtida convertendo os valores binários para a base decimal. Já para a marca temporal $\widehat{M}_{v,t}$, primeiramente é feita a identificação da sequência de 16 bits, $\widehat{\Phi}$. Como explicado na proteção, essa sequência é replicada por todo o quadro. Assim, cada um dos 16 bits da sequência $\widehat{\Phi}$ é obtido pela moda de todos seus bits equivalentes na matriz $\widehat{M}_{v,t}$. Ou seja, se redimensionarmos a matriz $\widehat{M}_{v,t}$ em um vetor unidimensional, cada bit da sequência de 16 bits, $\widehat{\Phi}$, seria obtido da seguinte forma:

$$\widehat{\Phi}(i) = moda\left(\widehat{M}_{v,t}(i + k \cdot 16)\right), \tag{4.5}$$

com k sendo um número inteiro que começa com 0 e varia até varrer todas as posições possíveis. A utilização da moda (medida estatística que retorna o valor mais frequente em uma sequência) torna a marca temporal robusta, visto que ela pode ser recuperada corretamente mesmo que parte da marca seja adulterada por ataques espaciais. Na Figura 4.8, é ilustrada a conversão da marca temporal repetida $\widehat{M}_{v,t}$ na sequência de 16 bits, $\widehat{\Phi}$. A sequência de 16 bits, $\widehat{\Phi}$, é, em seguida, convertida para a base decimal, $\hat{\phi}$. Com isso, um vetor, chamado de vetor de marcas temporais, é formado com o número $\hat{\phi}$ de cada quadro, identificando esse quadro.

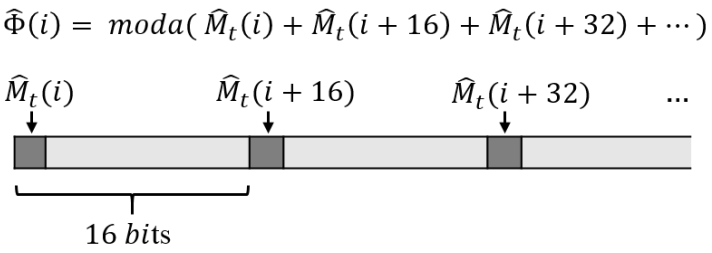


Figura 4.8: Formação da marca temporal de 16 bits.

A extração da marca do áudio é feita de maneira semelhante. A marca é extraída implementando-se a seguinte expressão:

$$\widehat{M}_{canal}[i] = T_m[i, canal] \; mod \; \Delta, \tag{4.6}$$

na qual $\widehat{M}_{canal}$ é a marca extraída de um dos dois canais, T_m é o trecho de áudio marcado e mod é a operação que retorna o resto da divisão entre o valor da amostra e Δ (no nosso caso, $\Delta = 4 \cdot step$).

As marcas de cada canal (esquerdo e direito) são divididas por $step$ (valor entre dois níveis de quantização do som), gerando marcas com valores inteiros. Em seguida, as duas marcas são convertidas para binário (2 bits por amostra) e concatenadas, formando uma marca $\widehat{M}_a[i]$ de 4 bits por amostra. A concatenação das marcas extraídas do áudio é ilustrada na Figura 4.9.

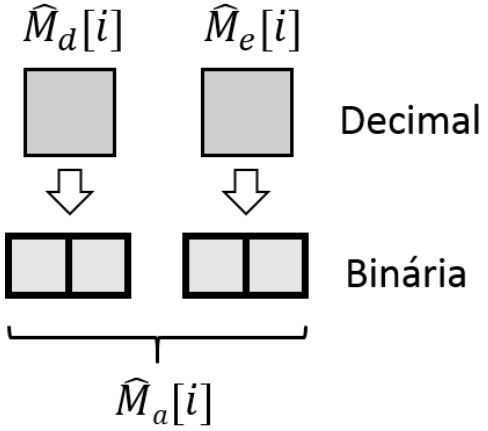


Figura 4.9: Concatenação da marca extraída do áudio.

Com a chave secreta, são geradas as posições dos bits temporais. Com isso, as marcas são separadas, formando uma marca espacial de 3 bits e uma temporal de 1 bit por amostra. A marca espacial $\widehat{M}_{a,s}$ é obtida após a conversão para decimal. O mesmo procedimento para marca temporal do vídeo é feito para o áudio: é identificada a sequência de 16 bits $\widehat{\Phi}$ de cada trecho de áudio a partir da moda dos bits equivalentes na matriz $\widehat{M}_{a,t}$ e criado um vetor com os números convertidos em decimais $\hat{\phi}$.

4.2.2 Análise das Adulterações Temporais

Extraídas as marcas temporais decimais $\hat{\phi}$, é feita uma comparação delas com as marcas de referência ϕ. As marcas de referência ϕ são iguais às marcas inseridas na mídia antes de sofrerem qualquer adulteração. São as marcas originais, que são novamente geradas através do gerador de números pseudoaleatórios e da chave secreta, a qual inicializa esse gerador.

Para a análise das adulterações temporais, um vetor de referência é gerado com os números da marca inicialmente inserida nos quadros do vídeo e trechos do áudio. Cada número desses, que representa um quadro (e o trecho de áudio equivalente), é comparado com os números das marcas temporais extraídas (tanto a do áudio quanto a do vídeo). Com essa comparação, é criado um vetor identificando os quadros (e trechos de áudio) adulterados. A Figura 4.10 ilustra um exemplo de comparação entre as marcas de cada quadro e o resultado indicando uma troca de posição entre os quadros 5 e 7.

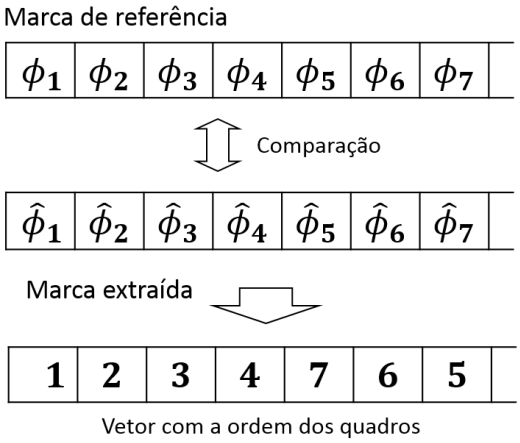


Figura 4.10: Análise das marcas temporais em vídeo com o 5º e 7º quadros trocados.

O vetor produzido com os valores indicativos de cada quadro é analisado pelo algoritmo para identificar e classificar as adulterações temporais sofridas. Valores encontrados mais de uma vez indicam uma duplicação de quadros. Valores esperados e não encontrados indicam a remoção dos quadros. Valores trocados de posição entre si indicam a troca de quadros.

Esse vetor com a numeração dos quadros é importante também na detecção de adulterações espaciais em mídias que sofreram também adulterações temporais. A análise das adulterações espaciais é explicada na próxima sessão. Por fim, um conjunto de quadros é produzido, com o mesmo conteúdo, mas com a identificação das adulterações temporais escritas sobre os respectivos quadros adulterados.

4.2.3 Análise das Adulterações Espaciais

Para a análise das adulterações espaciais, marcas de referência são criadas para o áudio e para o vídeo, utilizando a mesma chave secreta para marcar a mídia. Para cada pixel do vídeo há uma marca, assim como para cada amostra de áudio. Com isso, as comparações são feitas pixel a pixel, no caso do vídeo, ou amostra a amostra, no caso do áudio.

Comparando, identificam-se os pixels (ou as amostras) em que a marca espacial extraída não condiz com a de referência. Assim, com uma granularidade de um pixel (ou uma amostra), adulterações espaciais são detectadas.

Identificados os quadros atacados, é calculada a porcentagem de pixels adulterados para cada um desses quadros. Com essa porcentagem, é possível estimar se a adulteração é do tipo local ou global. Essa classificação também pode ser efetuada para o áudio como um todo. Nesse caso, feita de forma diferente: o ataque é considerado global se houver adulterações em todos os trechos do áudio. Por fim, um conjunto de quadros é produzido, com o mesmo conteúdo, mas com as adulterações espaciais evidenciadas em vermelho.

Se há a possibilidade de a mídia ter sofrido ataques temporais, primeiramente é feita uma análise com as marcas temporais, produzindo um vetor com as posições reais dos quadros (ou trechos de áudio), como explicado na sessão anterior. Com isso, é feita a correta comparação entre as marcas espaciais de referência e as extraídas equivalentes, indicadas pelo vetor.

5 SIMULAÇÕES E RESULTADOS

O algoritmo descrito no capítulo anterior foi implementado utilizando o software MATLAB®, plataforma com linguagem própria muito utilizada na área de processamento de sinais. As marcas são geradas utilizando o gerador de números pseudoaleatórios próprio do MATLAB®. A chave secreta que inicializa esse gerador é um número inteiro não negativo de tamanho qualquer. Também foi utilizado nas simulações o software *ffmpeg*, codificador/decodificador de código aberto, para separar e unir os canais de áudio e vídeo.

5.1 Descrição dos testes

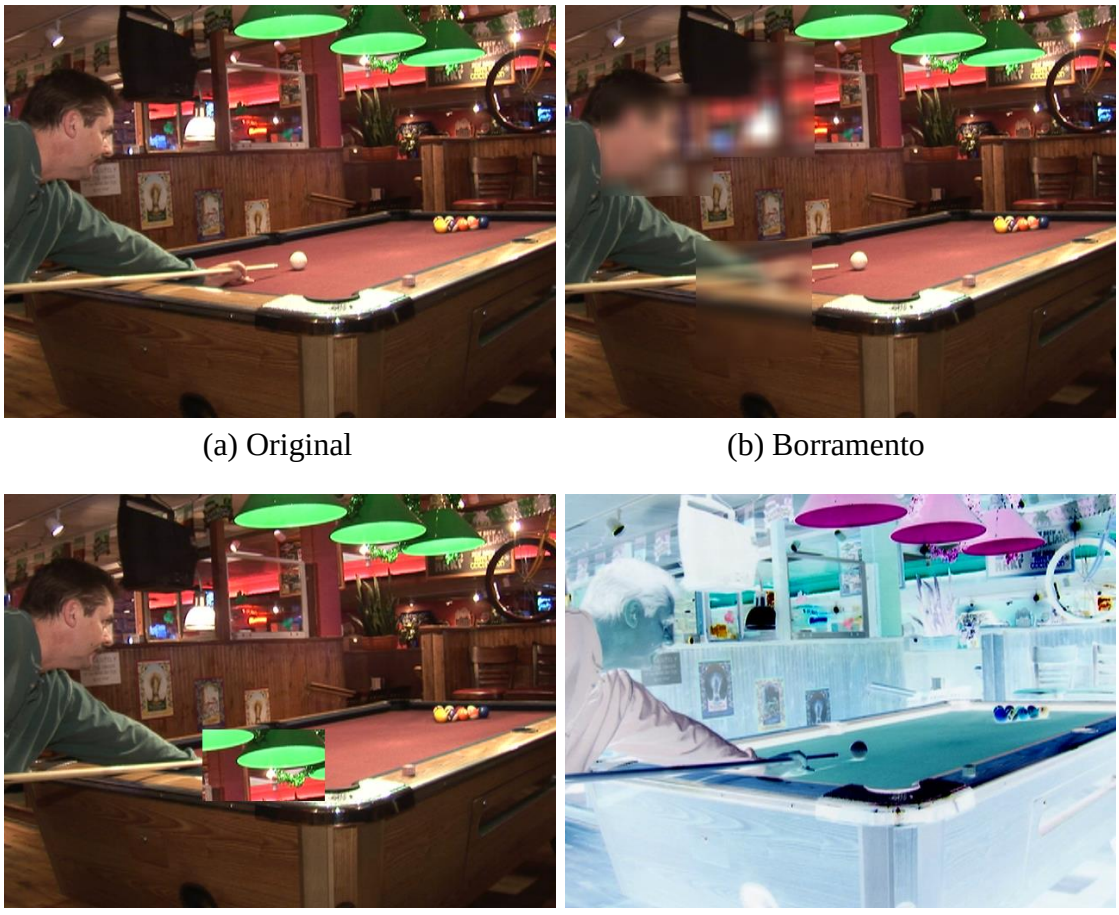
Para as simulações, foi utilizado um conjunto de 12 vídeos retirados da biblioteca digital *The Consumer Digital Video Library*, a qual disponibiliza seu conteúdo para pesquisa e desenvolvimento [18]. Um quadro de cada vídeo é disponibilizado na Figura 5.1. Os vídeos escolhidos possuem diferentes características espaciais (textura) e temporais (movimento). Todos possuem encapsulamento do tipo AVI com canais de áudio e vídeo descomprimidos.

Figura 5.1: Um quadro de cada vídeo utilizado: *Ballpit, Bells, Cartalk, Diner, Drummer, Flower, Guitar, Magic, Music, Pathsong, Pool, Waterfall*.

Ao total, foram testados 10 tipos de ataques. Dentre eles, três ataques espaciais feitos apenas nos quadros: Borramento, Inversão de Cores e Copiar e Colar. Quatro ataques feitos apenas em áudio: Anulação de Trecho, Distorção, Composição e Adição de Ruído Branco Gaussiano. E três ataques temporais feitos tanto em vídeo quanto em áudio: Remoção de Quadros, Duplicação de Quadros e Troca de Quadros. Aqui, entende-se quadros tanto como os quadros do vídeo quanto os trechos correspondentes em áudio.

5.2 Ataques espaciais na componente de vídeo

Para os três tipos espaciais, os quadros atacados são escolhidos aleatoriamente. No caso do Borramento, para cada quadro atacado, uma quantidade de até 3 retângulos com dimensões aleatórias tem suas áreas borradas. Isso é feito com um filtro espacial 21x21 que tira a média da vizinhança de cada pixel dos blocos atacados. Um exemplo desse ataque é mostrado na Figura 5.2(b). No caso da adulteração do tipo Copiar e Colar, uma região aleatória do quadro é copiada e colada em uma outra posição. Um exemplo desse ataque é mostrado na Figura 5.2(c). No caso de Inversão, as cores do quadro são invertidas, ou seja, para cada pixel é calculado o seu complemento em termos de cores. Sendo 255 o valor máximo para cada componente de cor do pixel, o complemento em cores é obtido subtraindo-se o valor original do valor máximo (255) daquela cor. Um exemplo desse ataque é mostrado na Figura 5.2(d).



(a) Original (b) Borramento

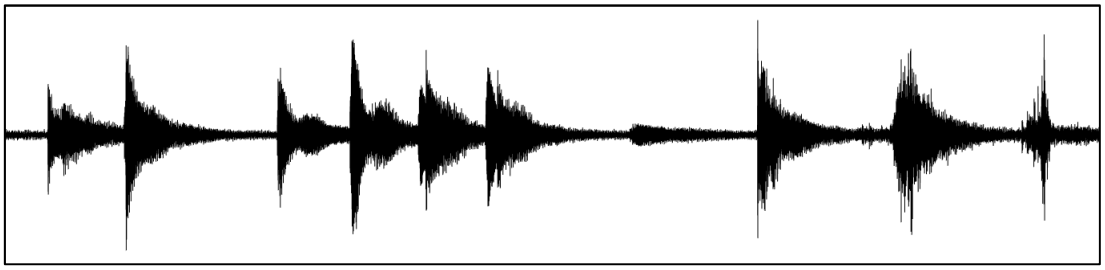
(c) Copiar e Colar (d) Inversão de Cores

Figura 5.2: Primeiro quadro do vídeo *Pool* e suas adulterações.

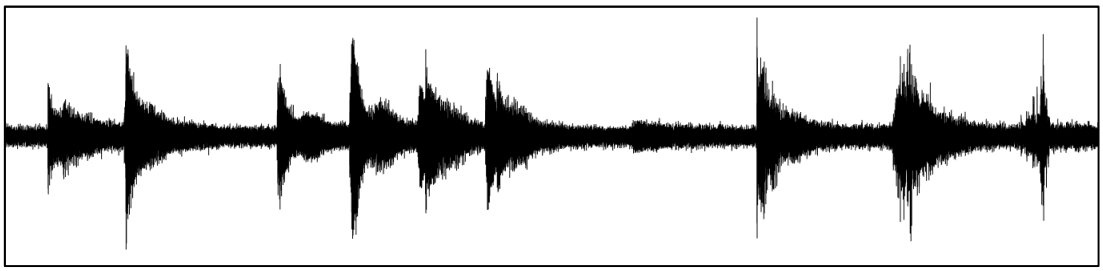
Aqui, para os ataques espaciais, cuja detecção não envolve a proteção do canal de áudio, foram escolhidos tipos diferentes daqueles simulados por Ronaldo. Buscando assim novos testes a essa modalidade de detecção. Outros três ataques (estes, simulados por Ronaldo) foram utilizados, mas não foram analisados com a mesma profundidade. São eles: Recorte (de blocos do quadro), Espelhamento (com relação ao eixo horizontal do quadro) e Adição de Ruído Sal-e-Pimenta.

5.3 Ataques na componente de áudio

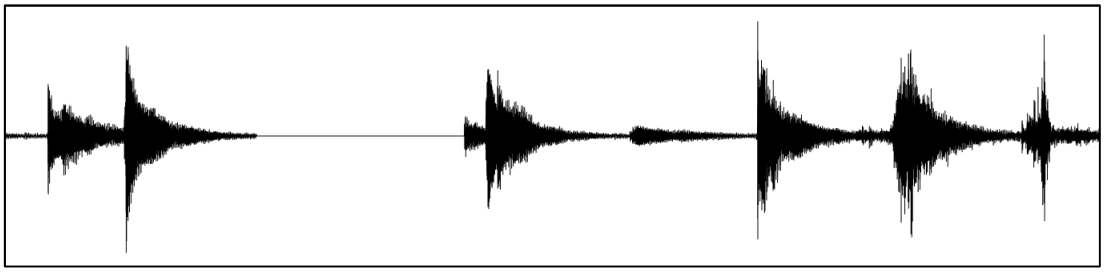
Para os ataques em áudio, a Adição de Ruído Branco Gaussiano adiciona um ruído branco (com espectro amplo) gaussiano (de distribuição normal) de 20 dB (com relação à energia média do sinal) ao áudio. Um exemplo desse tipo de ataque é ilustrado na Figura 5.3(b) com um ruído mais forte, de Relação Sinal-Ruído (RSR) igual a 10 dB, para facilitar a visualização.



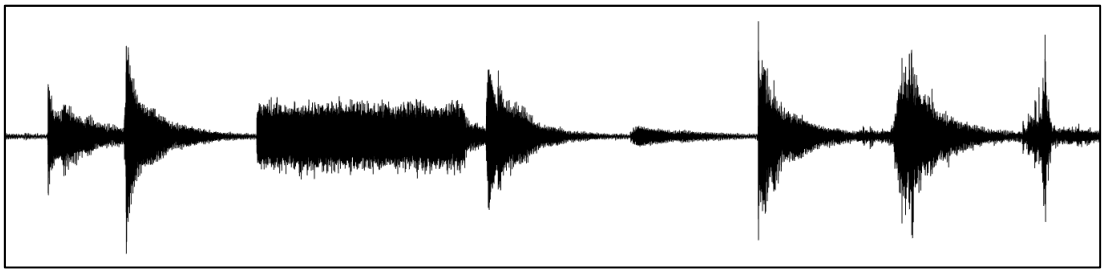
(a) Trecho original



(b) Adição de Ruído Branco Gaussiano com RSR = 10 dB

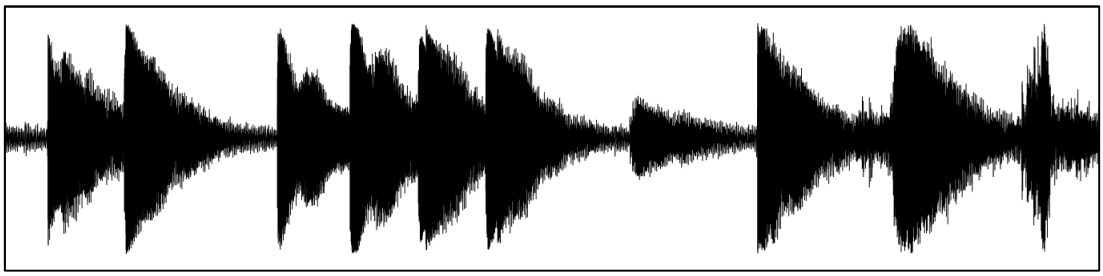


(c) Anulação de Trecho



(d) Composição

Figura 5.3: Exemplos de adulterações para um trecho de áudio do vídeo *Bells*.



(e) Distorção

Figura 5.4: Exemplo de adulteração por Distorção para um trecho de áudio do vídeo *Bells*.

O ataque do tipo Anulação de Trecho é feito ao levar a zero as amostras presentes em um trecho do áudio com posições de início e fim escolhidas de forma aleatória. Nesse caso, o comprimento do áudio não é alterado, pois o trecho é anulado e não retirado. Um exemplo desse tipo de ataque é ilustrado na Figura 5.3(c).

No caso de adulteração por Composição, um trecho do áudio é substituído por um outro qualquer de mesmo comprimento. Nas simulações, foi utilizado um ruído branco de energia média igual à do sinal original para fazer essa substituição. Um exemplo desse ataque é ilustrado na Figura 5.3(d).

Por fim, a adulteração por Distorção, uma distorção no sinal é criada de acordo com a seguinte equação, que tem efeito de ampliar as amostras de menor intensidade do áudio.

$$\hat{A}[i] = sign(A[i]) \cdot max(A) \cdot \left(1 - \alpha e^{-|\,A[i]\,|/\max(A)}\right). \tag{5.1}$$

Nesta expressão, $sign(A[i])$ é o sinal de $A[i]$, ou seja, $A[i]/|A[i]|$. O valor máximo do áudio, $max(A)$, é utilizado para que a função atue de forma semelhante para áudios de diferentes potências. Assim, a resposta dessa função é ilustrada pela curva na Figura 5.5, onde MAX $= max(A)$. Para as simulações, foi utilizado um fator de distorção α variando aleatoriamente entre 5,5 e 8,5. Um exemplo de adulteração por Distorção é ilustrado na Figura 5.4(e).

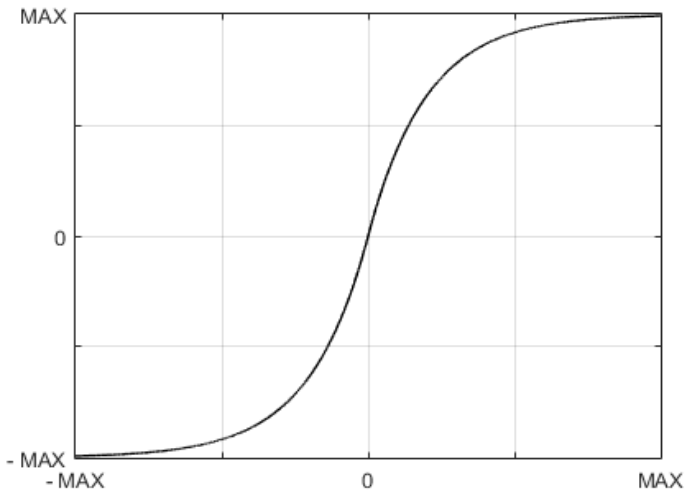


Figura 5.5: Curva da adulteração por Distorção.

5.4 Ataques temporais nas componentes de áudio e vídeo

Os ataques temporais são feitos em vídeo e em áudio simultaneamente. Para cada ataque, são escolhidos aleatoriamente 50 quadros do vídeo. Os quadros e correspondentes trechos de áudio são atacados igualmente, sendo removidos, duplicados ou trocados de posição.

No caso da Remoção de Quadros, os quadros escolhidos (e correspondentes trechos de áudio) são removidos da mídia. Essa adulteração causa uma redução na duração do vídeo.

Na Duplicação de Quadros, os quadros escolhidos (e correspondentes trechos de áudio) são duplicados e suas cópias são inseridas nas posições subsequentes. Essa adulteração causa um aumento na duração do vídeo.

Na Troca de Quadros, metade dos quadros escolhidos (e correspondentes trechos de áudio) são trocados de posição com a outra metade. Essa adulteração não altera a duração do vídeo.

5.5 Detecção de ataques espaciais nos quadros

Os primeiros testes foram feitos para verificar a capacidade do algoritmo de identificar adulterações espaciais locais e globais nos quadros atacados. Após feita a inserção da marca nos vídeos, alguns quadros foram selecionados e atacados. O algoritmo, então, é usado para detectar as adulterações. Nesse processo, os pixels identificados como adulterados são marcados em vermelho. Nas imagens a seguir, são apresentados alguns exemplos destes ataques.

(a) Quadro original de ‘*Drummer*’ (b) Ataque por Borramento (c) Detecção

(d) Quadro original de ‘*Bells*’ (e) Ataque por Copiar e Colar (f) Detecção

Figura 5.6: Detecção de ataques espaciais para os vídeos *Drummer* e *Bells*.

(g) Quadro original de ‘*Ballpit*’ (h) Inversão de Cores (i) Detecção

(j) Quadro original de ‘*Guitar*’ (k) Ataque por Sal-e-Pimenta (l) Detecção

(m) Quadro original de ‘*Flower*’ (n) Ataque por Recorte (o) Detecção

(p) Quadro original de ‘*Waterfall*’ (q) Ataque por Espelhamento (r) Detecção

Figura 5.7: Detecção de ataques espaciais para os vídeos *Ballpit, Guitar, Flower* e *Waterfall*.

Na primeira coluna das Figuras 5.6 e 5.7, temos os quadros originais, na segunda, os quadros atacados e, na terceira, a detecção em vermelho. Na Figura 5.6(b), três regiões foram borradas. Na Figura 5.6(e), uma região do quadro foi copiada e colada em outra posição. Na Figura 5.7(h), as cores do quadro foram invertidas. Na Figura 5.7(k), foi adicionado um ruído do tipo sal-e-pimenta. Na Figura 5.7(n), três regiões foram recortadas, ou seja, seus pixels foram levados a zero. Na Figura 5.7(q), o quadro foi espelhado com relação ao seu eixo horizontal.

A detecção das adulterações por Borramento, Recorte e Copiar e Colar mostram a capacidade do algoritmo na detecção de ataques localizados. No caso do ataque por Recorte, ele identificou

100% das adulterações pois o algoritmo foi configurado para não inserir marcas com valores nulos, assim, se algum pixel for anulado, ele irá identificar. A detecção da adição de ruído Sal-e-Pimenta demonstra que o algoritmo é capaz de detectar até mesmo ataques com granularidade de um pixel. Como o algoritmo também foi configurado para não inserir marcas de valor máximo possível, tantos os pontos pretos quanto os brancos são sempre identificados. Na detecção da Inversão de Cores, a eficiência também foi de 100%, isso é devido a um número par de níveis que a marca pode possuir em cada componente de cor ($\delta = 4$). O complemento de valores assim nunca resulta nele mesmo, pois não há um valor central. Logo, a marca é sempre adulterada.

Em seguida, prosseguimos com a análise da eficiência de detecção do algoritmo. Para isso, foram feitos testes para todos os 12 vídeos, nos quais todos os quadros foram atacados. Para cada quadro atacado foi calculada a taxa de detecção, que constitui na razão entre a área detectada como adulterada e a área adulterada de fato. Os testes foram feitos para os três ataques espaciais: Borramento, Inversão de Cores e Copiar e Colar. Os resultados são mostrados nos gráficos das Figura 5.8 e Figura 5.9, sendo que cada gráfico corresponde aos resultados obtidos para um único vídeo.

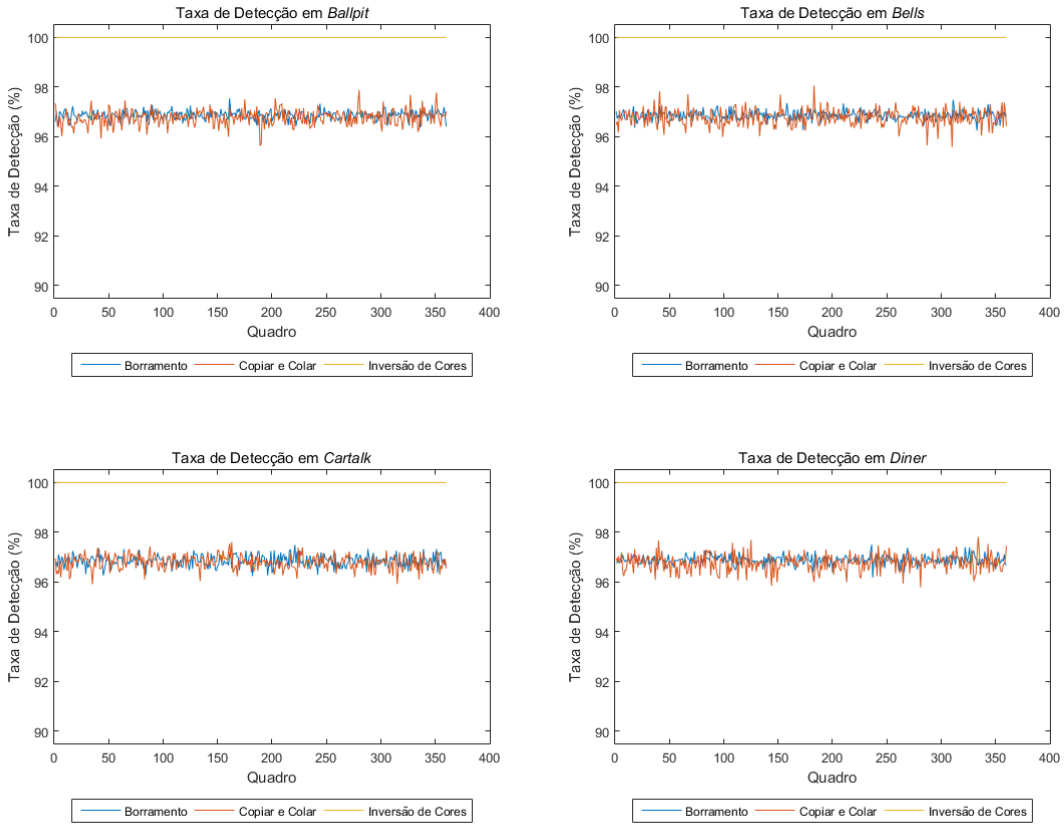


Figura 5.8: Gráficos com as taxas de detecção de adulterações espaciais para os vídeos *Ballpit*, *Bells*, *Cartalk* e *Diner*.

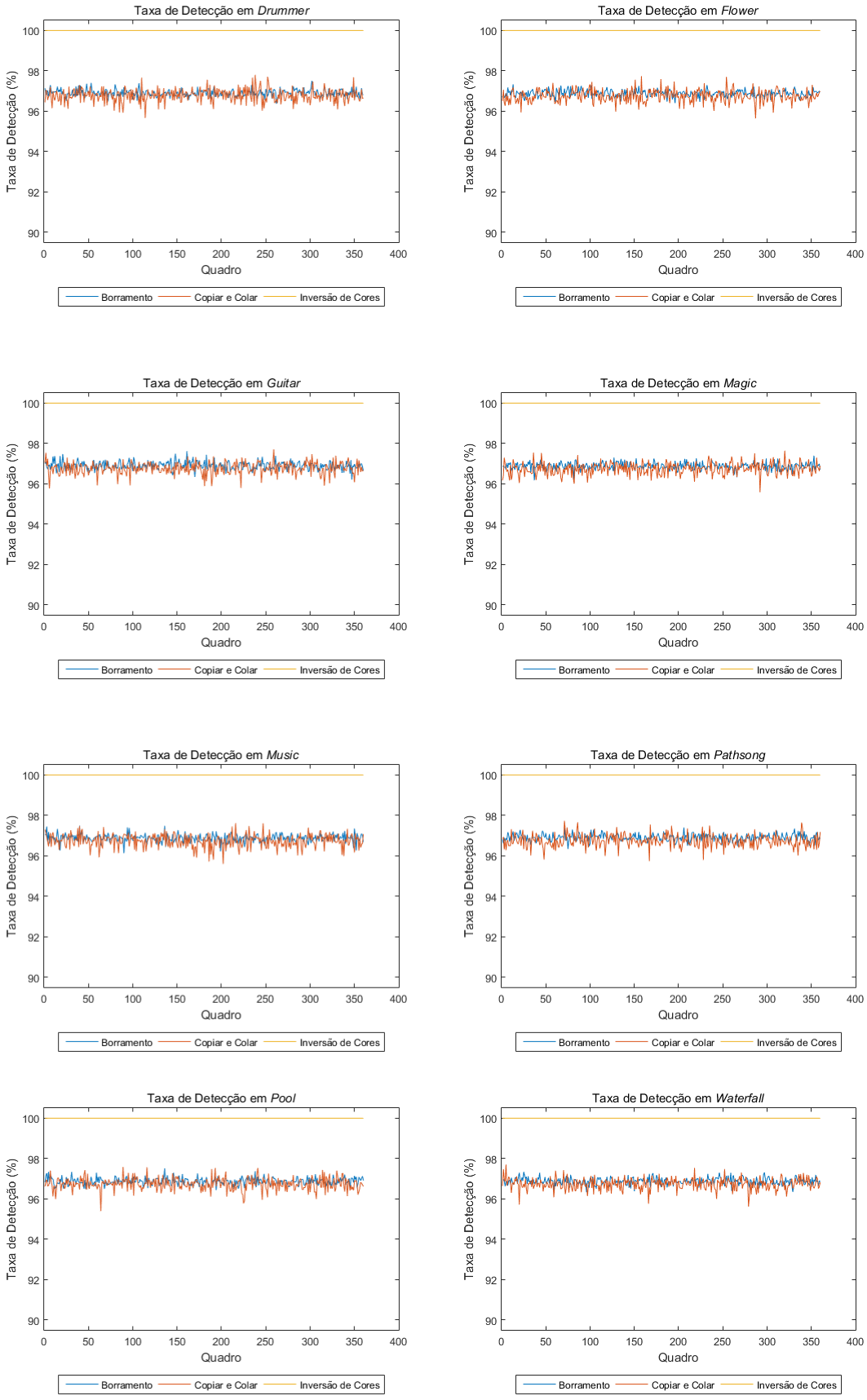


Figura 5.9: Gráficos com as taxas de detecção de adulterações espaciais para os vídeos, *Drummer*, *Flower*, *Guitar*, *Magic*, *Music*, *Pathsong*, *Pool* e *Waterfall*.

Os resultados para cada vídeo foram semelhantes. A taxa média de detecção para as adulterações tanto por Borramento quanto por Copiar e Colar ficaram próximas de 97% para todos os vídeos. Esse valor pode ser explicado analisando o número de valores possíveis da marca espacial. Ao ser gerada a marca espacial possui um valor inicial binário de 5 bits para cada pixel. Isso significa um total de $2^5 = 32$ possibilidades. Logo, a chance de uma marca adulterada de forma aleatória possuir o mesmo valor da original é de 1/32, ou 3,125%. Próxima da média de erro nessas detecções.

Para as adulterações por Inversão de Cores, a taxa de detecção foi sempre 100%. Como mencionado antes, isso é devido ao número par de níveis que a marca pode possuir em cada componente de cor ($\delta = 4$). O complemento de cada componente de cor resulta também no complemento da marca com relação ao seu valor máximo. Sendo 4 níveis de marca (por cor), o complemento do 1º nível é igual ao do 4º, e vice-versa, e o do 2º nível é igual ao do 3º, e vice-versa. Não havendo assim a possibilidade de a marca adulterada coincidir com a original.

As taxas médias de detecção, calculadas sobre todos os quadros de todos os vídeos, são ilustradas no gráfico da Figura 5.10.

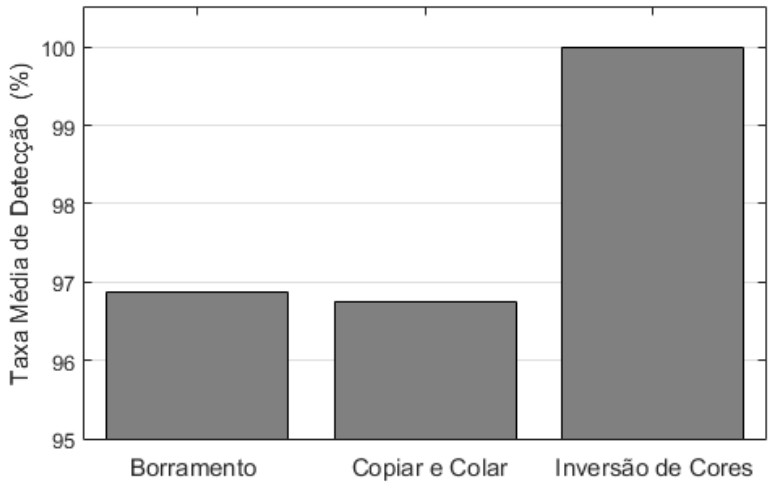


Figura 5.10: Taxa média de detecção para três tipos de adulterações espaciais.

5.6 Detecção de ataques no canal de áudio

Os áudios testados foram extraídos dos mesmos 12 vídeos utilizados para os ataques espaciais. Antes dos testes de detecção de adulterações, foi feito um estudo da qualidade dos áudios marcados. É importante fazer essa análise para se identificar o impacto da técnica de proteção na qualidade do áudio. Para isso, foi calculada a Relação Sinal-Ruído (RSR) para diferentes tamanhos de marca d'água. Como descrito no Capítulo 4, a marca é gerada em números binários e dividida para os dois canais de áudio. Assim, foram inseridas marcas de 2, 4, 6, 8 e 10 bits. Os valores são mostrados na tabela a seguir.

Tabela 5.1: Relação Sinal-Ruído para diferentes marcas e vídeos.

ÁUDIOS	Relação Sinal-Ruído (dB)				
	2 bits	4 bits	6 bits	8 bits	10 bits
Flower	52,69	46,00	38,94	33,44	27,95
Cartalk	52,99	46,31	39,25	33,78	28,25
Magic	56,83	50,15	43,09	37,59	32,10
Guitar	58,21	51,52	44,47	38,96	33,47
Bells	61,37	54,69	47,63	42,12	36,63
Pool	65,62	58,94	51,87	46,38	40,88
Ballpit	66,38	59,69	52,64	47,14	41,64
Music	70,47	63,79	56,72	51,22	45,73
Drummer	71,21	64,52	57,46	51,96	46,47
Diner	71,35	64,67	57,61	52,11	46,61
Pathsong	71,44	64,76	57,70	52,20	46,71
Waterfall	74,09	67,40	60,34	54,84	49,35
MÉDIA	**64,39**	**57,70**	**50,64**	**45,15**	**39,65**

Para cada valor da tabela, foi calculada a média da RSR (Relação Sinal-Ruído) para 10 inserções diferentes, usando chaves distintas. Os áudios da tabela estão ordenados de acordo com a energia média de seus sinais. Logo, os primeiros, de menor energia, sofrem maior impacto, apresentando menor RSR. Pode-se notar também que a RSR cai entre 5 e 7dB com o aumento de 2 bits da marca. Assim, buscando o menor impacto no áudio, o tamanho da marca deve ser o menor. Como a marca de 2 bits só possui um 1 bit de marca espacial, e desejamos que haja ao menos um bit de marca espacial para cada canal de áudio (esquerdo e direito), a marca escolhida foi a de 4 bits.

Concluída essa análise, prosseguimos com as simulações para a avaliação da eficiência de detecção do algoritmo no caso de adulterações em áudio. Foram 4 tipos de ataques, Adição de Ruído Branco Gaussiano, Composição, Anulação de Trecho e Distorção, para os 12 áudios. Para cada simulação, foi calculada a taxa de detecção, que constitui na razão entre o número de amostras detectadas como adulteradas e o número de amostras adulteradas de fato. Os resultados foram dispostos na Tabela 5.2, sendo que cada valor foi obtido pela média de 20 ataques.

Observa-se da Tabela 5.2, que as adulterações por Adição de Ruído Branco Gaussiano, Composição e Distorção tiveram taxas de aproximadamente 87,5%. Esse valor pode ser compreendido analisando o número de valores possíveis da marca espacial (a marca utilizada para essa detecção). Ao ser gerada, essa marca possui um valor inicial binário de 3 bits para cada amostra de áudio. Isso significa que uma marca adulterada aleatoriamente possui um total de $2^3 = 8$ possibilidades. Logo, a chance de ela possuir o mesmo valor da original é de 1/8, ou 12,5%. Próxima da média de erro dessas detecções. No caso do ataque por Anulação de Trecho, o algoritmo identificou 100% das adulterações, pois ele foi configurado para não inserir marcas com valores nulos, assim, se algum pixel for anulado, ele irá identificar.

Tabela 5.2: Média das taxas de detecção de adulterações em áudio.

ÁUDIOS	Ruído Branco	Composição	Anulação	Distorção
Ballpit	87,50	87,49	100,00	87,48
Bells	87,50	87,46	100,00	87,31
Cartalk	87,50	87,52	100,00	88,26
Diner	87,50	87,51	100,00	87,21
Drummer	87,51	87,51	100,00	87,19
Flower	87,50	87,49	100,00	86,74
Guitar	87,49	87,49	100,00	87,96
Magic	87,49	87,50	100,00	87,64
Music	87,49	87,48	100,00	87,29
Pathsong	87,50	87,50	100,00	87,41
Pool	87,51	87,51	100,00	87,61
Waterfall	87,50	87,49	100,00	87,43
MÉDIA	**87,50**	**87,50**	**100,00**	**87,46**

Para possibilitar uma análise mais visual, um gráfico na Figura 5.11 é apresentado com esses mesmos dados.

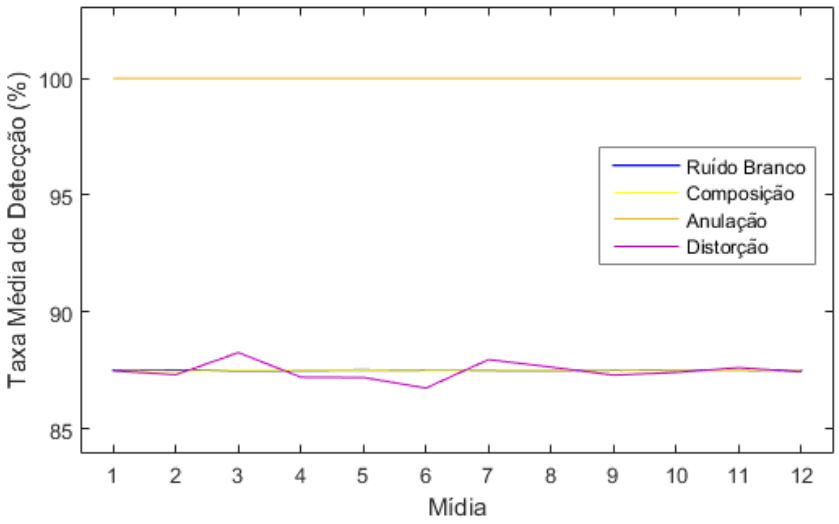


Figura 5.11: Gráfico das taxas médias de detecção de adulterações em áudio.

A partir do gráfico da Figura 5.11, pode-se notar que, em comparação com as outras, a adulteração por Distorção tem a curva com maior variação. Isso acontece, pois é a única adulteração que, de fato, depende da mídia atacada.

5.7 Detecção de ataques temporais em áudio e vídeo

A detecção de ataques temporais é feita tanto para vídeo quanto para áudio. Assim, nos testes realizados aqui, as adulterações temporais foram aplicadas igualmente nos quadros do vídeo e nos trechos correspondentes do áudio. É mais coerente pensar que uma adulteração irá atuar de

forma igual nos dois canais, porém o algoritmo verifica as adulterações de forma independente, podendo identificar diferentes ataques para os canais de áudio e vídeo.

Como descrito no Capítulo 4, tanto para vídeo, quanto para áudio, a marca temporal é extraída e comparada com a original gerada pela chave secreta. Com essa comparação, os quadros (ou trechos) são identificados, e um vetor é produzido com suas posições reais. Após a análise desse vetor, as adulterações são identificadas e classificadas. São três classificações feitas: Remoção de Quadros, Duplicação de Quadros e Troca de Quadros (aqui "quadros" também pode ser entendido como trechos do áudio).

O último passo do algoritmo é marcar visualmente os quadros, explicitando as adulterações encontradas. Alguns resultados obtidos com esse procedimento são mostrados nas figuras a seguir. Na Figura 5.12, é ilustrado que o algoritmo foi capaz de identificar e classificar um ataque por Duplicação de Quadros no vídeo *Pathsong*.

(a) Quadro 263 (b) Quadro 263 duplicado (c) Quadro 264

Figura 5.12: Detecção da duplicação do quadro 263 do vídeo *Pathsong*.

A análise feita para o áudio também pode ser escrita no quadro, fornecendo mais um recurso visual da capacidade de detecção do algoritmo. A Figura 5.13 ilustra o resultado da detecção com o áudio.

(a) Quadro 263 (b) Quadro 263 duplicado (c) Quadro 264

Figura 5.13: Detecção da duplicação de quadro e trecho de áudio do vídeo *Pathsong*.

A Figura 5.14 ilustra o resultado da detecção para o ataque por Remoção de Quadros no vídeo *Magic*. Não havendo o quadro atacado, o algoritmo criou um quadro cinza para evidenciar a análise feita.

(a) Quadro 40 (b) Remoção do quadro 41 (c) Quadro 42

Figura 5.14: Detecção da remoção do quadro 41 do vídeo *Magic*.

Após a análise do áudio, o resultado também é escrito no quadro, mostrando que o algoritmo foi capaz de detectar a adulteração tanto em vídeo, quanto em áudio, como ilustra a Figura 5.15.

(a) Quadro 40 (b) Remoção do quadro 41 (c) Quadro 42

Figura 5.15: Detecção da remoção de quadro e trecho de áudio do vídeo *Magic*.

O terceiro tipo de adulteração, Troca de Quadros, é ilustrado a seguir. Nesse caso, o quadro 29 do vídeo *Music* foi trocado de posição com o quadro 136. O algoritmo foi capaz de identificar os quadros atacados e suas posições corretas. A detecção de ambos os canais já é mostrada na Figura 5.16.

(a) Quadro 28 (b) 136 na posição do 29 (c) Quadro 30

(d) Quadro 135 (e) 29 na posição do 136 (f) Quadro 137

Figura 5.16: Detecção de Troca de Quadros no vídeo *Music*.

Para avaliar a eficiência de detecção do algoritmo no caso de adulterações temporais, foi construída uma tabela com as taxas de detecção para cada vídeo. Mostrada a seguir, cada entrada dessa tabela representa a taxa de detecção obtida após um ataque temporal de 50 quadros do vídeo (e áudio). Para quase todas as ocorrências, a taxa foi de 100%, evidenciando a capacidade do algoritmo. Apenas no caso do vídeo *Guitar*, a taxa de detecção para Remoção de Quadros apresentou 98% de acerto (tanto para áudio quanto para vídeo). Essa diferença de 2%, que, para 50 quadros adulterados, significa 1 quadro não detectado, ocorreu quando o último quadro foi removido. Como a técnica é sem referência (cega), ou seja, não há comparação com o conteúdo do vídeo original, o algoritmo não sabe quando realmente termina o vídeo, e não há como identificar se o último quadro for perdido.

Tabela 5.3: Taxas de detecção de adulterações temporais nas componentes de áudio e vídeo.

VÍDEOS	Duplicação		Remoção		Troca	
	Áudio	Vídeo	Áudio	Vídeo	Áudio	Vídeo
Ballpit	100	100	100	100	100	100
Bells	100	100	100	100	100	100
Cartalk	100	100	100	100	100	100
Diner	100	100	100	100	100	100
Drummer	100	100	100	100	100	100
Flower	100	100	100	100	100	100
Guitar	100	100	98	98	100	100
Magic	100	100	100	100	100	100
Music	100	100	100	100	100	100
Pathsong	100	100	100	100	100	100
Pool	100	100	100	100	100	100
Waterfall	100	100	100	100	100	100
MÉDIA	**100**	**100**	**99,8**	**99,8**	**100**	**100**

O algoritmo também possui a capacidade de detectar múltiplos ataques temporais de diferentes tipos, contanto que o mesmo quadro não seja atingido por mais de um tipo de adulteração.

5.8 Detecção de ataques espaciais e temporais

Como descrito no Capítulo 4, pode-se detectar adulterações espaciais, mesmo se os quadros estiverem fora de posição. Para isso, primeiro é feita uma análise com as marcas temporais, produzindo um vetor com as posições reais dos quadros. Sabendo as posições, o algoritmo pode fazer a correta comparação entre as marcas espaciais extraídas e as equivalentes de referência.

Um dos testes feitos é mostrado nas figuras a seguir. Na simulação, alguns quadros do vídeo *Cartalk* são adulterados espacialmente com ataques do tipo Copiar e Colar. Entre eles, o 10º quadro, que, em seguida, é trocado de posição com o quadro 70. Essas adulterações são mostradas na Figura 5.17.

(a) Quadro 69 (b) 10 na posição do 70 (c) Quadro 71

Figura 5.17: Adulterações espaciais e temporais no vídeo *Cartalk*.

Assim, primeiro foi feita a análise temporal, para produzir o vetor com as posições reais dos quadros. A Figura 5.18 serve para ilustrar que o algoritmo foi capaz de identificar a correta posição do quadro.

(a) Quadro 69 (b) 10 na posição do 70 (c) Quadro 71

Figura 5.18: Identificação da Troca de Quadros em '*Cartalk'*.

Tendo o vetor com as posições, o algoritmo pode analisar as marcas espaciais nas posições corretas e identificar as adulterações espaciais no quadro, como pode ser visto na figura seguinte.

(a) Quadro 69 (b) 10 na posição do 70 (c) Quadro 71

Figura 5.19: Identificação da adulteração espacial em '*Cartalk'*.

Mais um teste foi feito com o intuito de se verificar a eficiência dessa detecção mesmo com quadros duplicados ou deletados, os quais alteram as posições de todos os quadros seguintes. Assim, o vídeo *Diner*, teve primeiramente seu 6º quadro duplicado, deslocando a posição de todos os quadros seguintes. Em seguida, todos os seus quadros foram atacados espacialmente por Borramento. Essas adulterações são mostradas na Figura 5.20.

(a) Quadro 6 (b) Quadro 6 duplicado (c) Quadro 7

Figura 5.20: Adulterações espaciais e temporais no vídeo *Diner*.

Na Figura 5.21, é mostrado que o algoritmo identificou a Duplicação de Quadros, após análise temporal.

(a) Quadro 6 (b) Quadro 6 duplicado (c) Quadro 7

Figura 5.21: Identificação da Duplicação do 6º quadro.

Com o vetor de posições gerado, as adulterações espaciais foram identificadas corretamente, como mostra a Figura 5.22.

(a) Quadro 6 (b) Quadro 6 duplicado (c) Quadro 7

Figura 5.22: Identificação das adulterações espaciais em '*Diner'*.

Uma restrição à aplicação dessa técnica acontece quando adulterações globais comprometem a extração correta da marca temporal. Assim, com adulterações como Inversões de Cores, a qual degrada 100% da marca, não é possível fazer a verificação das marcas temporais e, com isso, não é possível identificar as reais posições dos quadros.

6 CONCLUSÕES

Foi apresentada uma ampliação do algoritmo proposto por Ronaldo Rigoni em sua dissertação de mestrado. O novo algoritmo estende a proteção, já conferida ao canal de vídeo, ao áudio utilizando a mesma técnica de inserção de marca d'água.

Com uma capacidade de inserção menor que a do vídeo, foi escolhida uma marca d'água de menor impacto na qualidade do áudio. Apesar da quantidade de informação ser reduzida, as taxas obtidas para detecção de adulterações em áudio foram de 87,5% a 100%. Demonstrando a validade e eficiência da técnica proposta.

Novos testes foram realizados para diferentes adulterações espaciais, ampliando a gama de ataques passíveis de identificação pelo algoritmo. Os resultados desses testes foram satisfatórios, com taxas médias de detecção acima de 96%. Confirmando, assim, a alta capacidade de detecção do algoritmo para uma grande variedade de adulterações.

Novos testes também foram feitos para adulterações temporais, agora com detecções tanto em áudio quanto em vídeo. As taxas médias de detecção de 100% demonstram a versatilidade e sensibilidade da técnica em conferir proteção para ambos os canais contra diferentes ataques temporais.

A técnica se mostrou robusta ao se demonstrar a possibilidade de se proteger o conteúdo espacial dos quadros, mesmo sofrendo ataques temporais simultâneos.

6.1 Contribuições

As contribuições deste trabalho estão no desenvolvimento de um algoritmo capaz de detectar adulterações espaciais e temporais tanto em vídeo quanto em áudio, aumentando a capacidade do algoritmo proposto por Ronaldo. As funcionalidades acrescidas são:

- Aumento da gama de ataques espaciais em vídeo suportados pelo algoritmo, adicionando detecção aos ataques Copiar e Colar, Borramento e Inversão de Cores;
- Proteção do canal de áudio com detecção de variada gama de ataques;
- Detecção e classificação de ataques temporais que afetam tanto vídeo quanto áudio;
- Detecção de ataques espaciais nos quadros, mesmo os quadros estando fora de posição.

6.2 Trabalhos Futuros

Entre os possíveis trabalhos futuros, está a ampliação ainda maior da quantidade de adulterações, tanto espaciais quanto temporais, suportada pela técnica.

Assim como são identificadas e classificadas as adulterações temporais, um trabalho poderia se concentrar na classificação de ataques espaciais.

Outra possibilidade seria a de gerar e trabalhar com marcas cujos valores tenham relação com o conteúdo das mídias marcadas.

REFERÊNCIAS BIBLIOGRÁFICAS

1. ZHANG, Z. et al. **A Survey on Passive-Blind Image Forgery by Doctor Method Detection**. Proceedings of the Seventh International Conference on Machine Learning and Cybernetics. Kunming: July. 2008. p. 12-15.
2. LIN, E. T. **Video and Image Watermark Synchronization**. Purdue University. West Lafayette. 2005.
3. WANG, W. **Digital Video Forensics**. Dartmouth College. Hanover. 2009.
4. CROSS, D.; MOBASSERI, B. G. **Watermarking for self-authentication of compressed video**. 2002 International Conference on Image Processing. [S.l.]: IEEE. 2002. p. 913-916.
5. HOU, Z.; TANG, X. **Integrity authentication scheme of color video based on the fragile watermarking**. Electronics, Communications and Control (ICECC), International Conference on. Ningbo: IEEE. 2011. p. 4354 - 4358.
6. CHEN, X.; ZHAO, H. **A novel video content authentication algorithm combined semi-fragile watermarking with compressive sensing**. Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on. Sanya: IEEE. 2012. p. 134 - 137.
7. RIGONI, R. **Detecção de Adulterações Espaciais e Temporais em Vídeos Digitais Utilizando o Algoritmo de Marca D'Água por Modulação do Índice de Quantização**. Universidade de Brasília. Brasília. 2013.
8. CHEN, B.; WORNEL, G. W. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. **Information Theory, IEEE Transactions on**, v. 47, n. 4, p. 1423–1443, 2001.
9. KEE, E.; FARID, H. A perceptual metric for photo retouching. **Proceedings of the National Academy of Sciences**, Hanover, v. 108, n. 50, p. 19907–19912, Dec. 2011.
10. GHOBADI, A. et al. **Blind Audio Watermarking for Tamper Detection Based on LSB**. Advanced Communication Technology (ICACT), 2013 15th International Conference on. PyeongChang: IEEE. 2013. p. 1077 - 1082.
11. COX, I. J. et al. Secure Spread Spectrum Watermarking For Multimedia. **Image Processing, IEEE Transactions on**, v. 6, n. 12, p. 1673 –1687, Dec. 1997.
12. LANGELAAR, G. C.; SETYAWAN, I.; LAGENDIJK, R. L. Watermarking Digital Image and Video Data: A State-Of-The-Art Overview. **IEEE Signal Processing Magazine**, p. 20-46, September 2000.
13. ANDERSON, R. J.; PETITCOLAS, F. A. P. On The Limits of Steganography. **Selected Areas in Communications, IEEE Journal on**, v. 16, n. 4, p. 474–481, 1998.
14. COX, I. J. et al. **Digital Watermarking and Steganography**. 2. ed. p. 36–46: Morgan Kaufmann Publishers, 2007.
15. KEJARIWAL, A. Watermarking. **IEEE Potentials Magazine**, p. 37–40, October/November 2003.
16. ABDULLATIF, M. et al. **Properties of Digital Image Watermarking**. 2013 IEEE 9th International Colloquium on Signal Processing and its Applications. Kuala Lumpur, Malaysia: (CSPA). 2013. p. 235 - 240.

17. SEO, Y. et al. QIM Watermarking for Image with Two Adaptive Quantization Stepsizes. **The 9th International Conference on Advanced Communication Technology**, v. 1, p. 797-800, Feb. 2007.
18. THE CONSUMER DIGITAL VIDEO LIBRARY. Disponível em: <http://www.cdvl.org>. Acesso em: Maio 2016.

ANEXOS

Anexo I - Funções em MATLAB

São anexados aqui alguns códigos feitos em MATLAB utilizados neste trabalho.

(a) Função de inserção da marca d'água no vídeo:

```
function [ frames ] = marcavideo( frames, chave )
%MARCAVIDEO Fornece quadros com marcas espacial e temporal

tic
fsize = size(frames);
lframe = fsize(1)*fsize(2); %numel

%Matriz índice para repetição da marca temp. ao longo de 1 frame
quoc = floor(lframe./16);
rest = mod(lframe, 16);
indice = reshape([repmat(1:16,1,quoc) 1:rest],fsize(2),fsize(1))';

%Marcas temporais(16 bits) para todos os quadros
rng(chave); %seed
mt = randperm(65535);
mtbin = uint8(de2bi(mt, 16)); %default MSB-right
p = randi([1 6],fsize(1),fsize(2)); %pos. do bit temporal

fprintf('%04u', 0);

for i = 1:fsize(4) %frames

    frame = double(frames(:,:,:,i));
    qframe = uint8(floor(frame./4).*4);

    mtframe = mtbin(i,indice); %1bit de profundidade
    ms = randi([1 30],fsize(1),fsize(2));%ainda decimal
    msbin = uint8(de2bi(ms, 6)); %5bits espacial +1 espaço p/ temporal

    qframered = qframe(:,:,1);
    qframegre = qframe(:,:,2);
    qframeblu = qframe(:,:,3);

    seis = [6 1 2 3 4 5; 1 6 2 3 4 5; 1 2 6 3 4 5; 1 2 3 6 4 5; 1 2 3 4 6 5; 1 2 3
4 5 6];
    msbin(:,6) = mtframe(:);

    for j = 1:lframe %pixels

        mbin = msbin(j, seis(p(j),:)); %indice_p(j,:));%

        qframered(j) = qframered(j) + mbin(1) + 2*mbin(2);
        qframegre(j) = qframegre(j) + mbin(3) + 2*mbin(4);
        qframeblu(j) = qframeblu(j) + mbin(5) + 2*mbin(6);
```

```
    end

    frames(:,:,1,i) = qframered;
    frames(:,:,2,i) = qframegre;
    frames(:,:,3,i) = qframeblu;

    %Contagem dos frames processados
    fprintf('\b\b\b\b%04u', i); %unsigned zero-padded
end
t = toc;
fprintf(' frames marcados em %.1f s\n', t);
%16 bits = [0 65535], 5 bits = [0 31]
end
```

(b) Função de adulteração espacial do vídeo (Inversão, Borramento e Copiar e Colar):

```
function [ altframes, out ] = adulteraS( qframes, tipo )
rng('shuffle');
fsize = size(qframes);

%Frames a serem adulterados
nframes = round(fsize(4)*0.05); %5% dos frames são adulterados
adulterados = sort(randperm(fsize(4),nframes)); %unique values

percent = zeros(size(adulterados)); %porcentagens
altframes = qframes;
k = 1;
switch tipo

    case {'inversao','neg'}

        for i = adulterados %frames
            altframes(:,:,:,i) = 255 - qframes(:,:,:,i);
            percent(k) = 100;
            k = k + 1;
        end

    case {'borramento','borra','bor'}

        n = 10; %averaging 2n+1
        %Limites das dimenções
        hlimits = round(fsize(1).*[0.20 0.30]);
        wlimits = round(fsize(2).*[0.20 0.30]);

        for i = adulterados %frames

            nadult = randi([1 3],1);%num. de adulterações
            pixels = zeros(fsize(1),fsize(2));

            for j = 1:nadult %adulterações
                h = randi(hlimits,1);
                w = randi(wlimits,1);
                x1 = randi([1 (fsize(1)-h+1)],1);
                y1 = randi([1 (fsize(2)-w+1)],1);

                for x = x1:x1+h-1
                    for y = y1:y1+w-1
```

```
                        xlessn = x-n;
                        ylessn = y-n;
                        xmoren = x+n;
                        ymoren = y+n;
                        if xlessn<1, xlessn = 1; end
                        if ylessn<1, ylessn = 1; end
                        if xmoren>fsize(1), xmoren = fsize(1); end
                        if ymoren>fsize(2), ymoren = fsize(2); end
altframes(x,y,1,i) = sum(sum(qframes(xlessn:xmoren,ylessn:ymoren,1,i)))/(2*n+1)^2;
altframes(x,y,2,i) = sum(sum(qframes(xlessn:xmoren,ylessn:ymoren,2,i)))/(2*n+1)^2;
altframes(x,y,3,i) = sum(sum(qframes(xlessn:xmoren,ylessn:ymoren,3,i)))/(2*n+1)^2;
                        pixels(x,y)=1;
                    end
                end
            end
            percent(k) = 100*nnz(pixels)/(fsize(1)*fsize(2));
            fprintf('\b\b\b\b%04u', k); %unsigned zero-padded
            k = k + 1;
        end

    case {'copiarcolar','cc'}

        %Limites das dimenções
        hlimits = round(fsize(1).*[0.20 0.30]);
        wlimits = round(fsize(2).*[0.20 0.30]);

        for i = adulterados %frames

                dh = randi(hlimits,1)-1; %h = 1+dh
                dw = randi(wlimits,1)-1; %w = 1+dw
                x1 = randi([1 (fsize(1)-dh)],1);
                y1 = randi([1 (fsize(2)-dw)],1);
                copia = qframes(x1:x1+dh, y1:y1+dw, :,i);

                xlessdh = x1-dh;
                ylessdw = y1-dw;
                if xlessdh<1, xlessdh = 1; end
                if ylessdw<1, ylessdw = 1; end

                j = 1:(fsize(1)*fsize(2));
                j = reshape(j,fsize(1),fsize(2));
                j(xlessdh:x1+dh, ylessdw:y1+dw) = 0;
                j(fsize(1)-dh:fsize(1),:) = 0;
                j(:,fsize(2)-dw:fsize(2)) = 0;
                j = j(:)';
                j(j==0)=[];

                j2 = j(randi(length(j))); %j1 = sub2ind(size(),x,y);
                [x2, y2] = ind2sub([fsize(1) fsize(2)], j2);
                altframes(x2:x2+dh, y2:y2+dw, :,i) = copia;

                percent(k) = 100*((1+dh)*(1+dw))/(fsize(1)*fsize(2));
                k = k + 1;
        end
    otherwise
        error('Tipo de adulteração não válido');
end
display(adulterados)
out = [adulterados' percent'];
```

```
end
```

(c) Função de extração e análise da marca espacial no vídeo:

```
function [ qframes, taxas ] = verificaS( qframes, chave )
%VERIFICAS Extrai a ms e a analisa.

%IMPORTANTE: a sequência de criação da marca pseudo-aleatória deve ser
%igual à da função de inserção da marca
tic
fsize = size(qframes);
lframe = fsize(1)*fsize(2); %numel
if ndims(qframes)==3, fsize(4)=1; end

%Gerando as marcas para comparação (igual ao código de inserção)
rng(chave); %seed
mt = randperm(65535); %temporal(16 bits)
mtbin = uint8(de2bi(mt, 16)); % importante pois mantem o rng stream
p = randi([1 6],fsize(1),fsize(2)); %pos. do bit temporal

taxas = zeros(fsize(4),2);
marcadif = zeros(fsize(1),fsize(2),fsize(4),'uint8');

fprintf('%04u', 0);

for i = 1:fsize(4) %frames

    %Extração da marca
    extcor = mod(qframes(:,:,:,i), 4);%marca decimal([0-3] p cada cor)[uint8]
    extframe = extcor(:,:,1) + 4*extcor(:,:,2) + 16*extcor(:,:,3);
    extbin = uint8(de2bi(extframe, 6));

    %mtframe = mtbin(i,indice); %1bit de profundidade
    ms = randi([1 30],fsize(1),fsize(2));%ainda decimal
    %msbin = uint8(de2bi(ms, 5)); %5bits espacial

    framedif = zeros(fsize(1),fsize(2),'uint8');

    for j = 1:lframe %pixels

        nop = (1:6 ~= p(j)); %posição dos bits espaciais
        extmsbin = extbin(j,nop);

        extmsdec = extmsbin(1) + 2*extmsbin(2) + 4*extmsbin(3) ...
            + 8*extmsbin(4) + 16*extmsbin(5);

        %verificaRR - subtraindo marcas extraída e gerada
        if extmsdec >= ms(j) %uint8 não permite negativos
            framedif(j) = extmsdec - ms(j);
        else
            framedif(j) = ms(j) - extmsdec;
        end

        %verificaRR - evidencia adulterações(em vermelho)
        if extmsdec ~= ms(j)
            [r, c] = ind2sub([fsize(1) fsize(2)],j);
            qframes(r,c,1,i) = 255;
```

```
            qframes(r,c,2,i) = 0;
            qframes(r,c,3,i) = 0;
        end
    end

    taxas(i,:) = [i 100*nnz(framedif)/lframe]; %pixels adulterados(%)
    marcadif(:,:,i) = framedif; %output alternativo

    %Contagem dos frames processados
    fprintf('\b\b\b\b%04u', i); %unsigned zero-padded
end

t = toc;
fprintf(' frames verificados em %.1f s\n', t);
taxas((taxas(:,2)==0),:) = []; %elimina linhas nulas
if ~isempty(taxas)
    fprintf('Frames atacados e porcentagens de pixels adulterados\n');
    disp(taxas)
else
    fprintf('Sem adulterações espaciais.\n');
end
end
```

(d) Função de adulteração temporal no vídeo (Remoção, Duplicação e Troca de Quadros):

```
function [ altframes ] = adulteraT( frames, n)
%UNTITLED2 Summary of this function goes here

rng('shuffle');
nframes = size(frames, 4);

if n < 0 %deleção %remoção

    if isscalar(n)
        deletados = randperm(nframes,-n);
    else
        deletados = -n;
    end
    deletados = sort(deletados,'descend');
    altframes = frames;
    altframes(:,:,:,deletados) = [];

    display(sort(deletados));

elseif n > 0 %duplicação

    if isscalar(n)
        duplicados = sort(randi([1 nframes],1,n));
    else
        duplicados = sort(n);
    end
    indice = sort([1:nframes duplicados]);
    altframes = frames(:,:,:,indice);
    display(duplicados);

else %troca
```

```
    ntrocas = 15; %número de trocas
    %Frames a serem trocados entre si (2*ntrocas)
    trocados = randperm(nframes,2*ntrocas);
    trocados = reshape(trocados,ntrocas,2); %duas colunas
    trocados = sortrows(sort(trocados,2)); %organizado
    altframes = frames;
    for i = 1:ntrocas %frames
        altframes(:,:,:,trocados(i,1)) = frames(:,:,:,trocados(i,2));
        altframes(:,:,:,trocados(i,2)) = frames(:,:,:,trocados(i,1));
    end
    display(trocados)

end
end
```

(e) Função de extração da marca temporal e formação do vetor de posições reais:

```
function [ vetoridt ] = verificaT( qframes, chave )
%VERIFICAT Analisa os frames marcados
%   Detailed explanation goes here

%IMPORTANTE: a sequência de criação da marca pseudo-aleatória deve ser
%igual à da função de inserção da marca
tic
fsize = size(qframes);
lframe = fsize(1)*fsize(2); %numel

%Gerando as marcas para comparação (igual ao código de inserção)
rng(chave); %seed
mt = randperm(65535); %temporal(16 bits)
%mtbin = uint8(de2bi(mt, 16)); %default MSB-right
p = randi([1 6],fsize(1),fsize(2)); %pos. do bit temporal

extmt = zeros(1,fsize(4)); %analisar
vetoridt = zeros(1,fsize(4));
mtverif = false(1,fsize(4));

fprintf('%04u', 0);

for i = 1:fsize(4) %frames

    extcor = mod(qframes(:,:,:,i), 4);%marca decimal([0-3] p cada cor)[uint8]
    extframe = extcor(:,:,1) + 4*extcor(:,:,2) + 16*extcor(:,:,3);
    extbin = uint8(de2bi(extframe, 6));

    %mtframe = mtbin(i,indice); %1bit de profundidade
    ms = randi([1 30],fsize(1),fsize(2));%importante pois mantem o rng stream
    %msbin = uint8(de2bi(ms, 5)); %5bits espacial

    extmtframe = zeros(fsize(1),fsize(2),'uint8');

    for j = 1:lframe %pixels
        extmtframe(j) = extbin(j,p(j));
    end

    extmtframe = extmtframe';
    extmtbin = zeros(1,16); %precisa ser double
```

```
    for t = 1:16
        extmtbin(t) = mode(extmtframe(t:16:end));
    end
    extmt(i) = bi2de(extmtbin);

    mtverif(i) = (extmt(i) == mt(i));

    if (extmt(i) == mt(i))
        vetoridt(i) = i; %identificação temporal
    elseif find(mt==extmt(i))
        vetoridt(i) = find(mt==extmt(i));
    else
        vetoridt(i)= -1;
    end

    %Contagem dos frames processados
    fprintf('\b\b\b\b%04u', i); %unsigned zero-padded
end

t = toc;
fprintf(' frames verificados em %.1f s\n', t);

if all(mtverif)
    disp('Sem alterações temporais')
end
end
```

(f) Função de análise do vetor de posições e classificação das adulterações temporais no vídeo.

```
function [ qframes, vetortemp ] = analiseVT( vetortemp, qframes )

%Pré-evidência
%valor para todos insertText, onde a pos. [x y] = [n m]
position = round([size(qframes,2)/2 size(qframes,1)*2/5]);

%%% Trocados
%(Pode haver outras adulterações temporais, contanto que não sejam com
% os mesmos quadros que foram trocados)

%Vetor desordenados
vetorsort = sort(vetortemp);
desordenados = zeros(length(vetortemp),3);
for i = 1:length(vetortemp)

    if (vetortemp(i) ~= vetorsort(i))
        desordenados(i,:) = [vetortemp(i) vetorsort(i) i];
        %[id do desordenado / id do qual devia estar ali / posição]
    end
end
desordenados( ~any(desordenados,2), : ) = []; %retira linhas nulas

%Vetor trocados
trocados = zeros(size(desordenados));
for j = 1:size(desordenados,1)

    x = find(desordenados(:,2)==desordenados(j,1));
```

```
    if any(x) && (desordenados(x,1)==desordenados(j,2))

        trocados(j,:) = desordenados(j,:);
    end
end
trocados( ~any(trocados,2), : ) = []; %retira linhas nulas
trocas = unique(sort(trocados(:,1:2),2),'rows');

if trocas
    fprintf('%i troca(s):\n', size(trocas,1));
    disp(trocas);

    %Evidência - trocados
    for t = 1:size(trocados,1);
        troc = trocados(t,3); %posição
        errortext = sprintf('\n\n\n  (quadro %i no lugar do quadro %i)  \n',...
            trocados(t,1),trocados(t,2));
        qframes(:,:,:,troc) = insertText(qframes(:,:,:,troc),position, ...
            errortext,'AnchorPoint','Center','FontSize',15, ...
            'BoxColor','red','BoxOpacity',0.3,'TextColor',[255 255 80]);
        errortext = sprintf('ERRO  TEMPORAL\n\n');
        qframes(:,:,:,troc) = insertText(qframes(:,:,:,troc),position, ...
            errortext,'AnchorPoint','Center','FontSize',17, ...
            'BoxColor','red','BoxOpacity',0,'TextColor',[255 255 80]);
    end
end

%%% Duplicados e Deletados

freq = zeros(1,max(vetortemp));
for i = 1:max(vetortemp)
    freq(i) = nnz(vetortemp==i); %number of nonzeros
end
framesid = 1:max(vetortemp);
deletados = framesid(freq==0); %ids dos deletados
duplicados = framesid(freq>1); %ids dos duplicados
duplicados = [duplicados; freq(duplicados)-1]'; %freq(freq>1)-1

if any(duplicados)
    nduplic = length(vetortemp) - length(unique(vetortemp));
    fprintf(['total de %i duplicações dos seguintes frames (com o \n' ...
    'respectivo número de duplicações encontradas):\n'], nduplic);
    disp(duplicados);

    %Evidência - duplicados
    for dup = 1:size(duplicados,1)
        posdup = find(vetortemp==duplicados(dup,1),1);
        text_dup = sprintf(['\n       ERRO  TEMPORAL\n\n' ...
            '  (quadro %i duplicado)  \n'],duplicados(dup,1));
        for d = 1:duplicados(dup,2) %frequência
            copia = posdup + d;
            qframes(:,:,:,copia) =  insertText(qframes(:,:,:,copia), ...
               position,text_dup,'AnchorPoint','Center','FontSize',17, ...
               'BoxColor','red','BoxOpacity',0.3,'TextColor',[255 255 80]);
        end
    end
end
```

```
if any(deletados)
    fprintf('%i deletado(s):\n', length(deletados));
    disp(deletados);

    %Evidência - deletados (por último, pois altera as posições ao inserir frames)
    vetorcomdel = sort([vetortemp deletados]);
    framecinza = 128*ones(size(qframes(:,:,:,1)),'uint8');
    for del = deletados
        posdel = find(vetorcomdel==del);
        text_dup = sprintf(['\n      ERRO TEMPORAL\n\n' ...
            '  (quadro %i deletado)  \n'],del);
        framedup =  insertText(framecinza,position,text_dup, ...
            'AnchorPoint','Center','FontSize',17,'BoxColor','red', ...
            'BoxOpacity',0.3,'TextColor',[255 255 80]);
        belowd = (1:length(vetortemp) < posdel);
        qframes = cat(4, qframes(:,:,:,belowd),framedup, ...
            qframes(:,:,:,~belowd));
        vetortemp = [vetortemp(belowd) 0 vetortemp(~belowd)];
        %^importante a auto iteração
    end
end
end
```