**UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC**
**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**FLAVIO HENRIQUE CIMOLIN**

# ANÁLISE COMPARATIVA ENTRE ALGORITMOS DE ESTIMAÇÃO DE MOVIMENTO POR *BLOCK-MATCHING* PARA COMPRESSÃO DE VÍDEO EM RESOLUÇÃO FULL HD

**CRICIÚMA**
**2018**

**FLAVIO HENRIQUE CIMOLIN**

# ANÁLISE COMPARATIVA ENTRE ALGORITMOS DE ESTIMAÇÃO DE MOVIMENTO POR *BLOCK-MATCHING* PARA COMPRESSÃO DE VÍDEO EM RESOLUÇÃO FULL HD

Projeto de Pesquisa do Trabalho de Conclusão de Curso em Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Giácomo Antônio Althoff Bolan

**CRICIÚMA**

**2018**

**FLAVIO HENRIQUE CIMOLIN**

# ANÁLISE COMPARATIVA ENTRE ALGORITMOS DE ESTIMATIVA DE MOVIMENTO POR *BLOCK-MATCHING* PARA COMPRESSÃO DE VÍDEO NA RESOLUÇÃO FULL HD

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharelado, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Processamento de Imagens.

Criciúma, 25 de Junho de 2018.

**BANCA EXAMINADORA**

Prof. Giácomo Antônio Althoff Bolan - Esp. - UNESC - Orientador

Prof. Evânio Ramos Nicoleit - Me. - UNESC

Prof. Marcel Campos Inocêncio - Esp. - UNESC

**À minha mãe, meu irmão, minha namorada e a Deus, que estiveram sempre ao meu lado, estivesse eu disposto ou não a compreender.**

## AGRADECIMENTOS

Devo prestar meus agradecimentos primeiramente à minha querida mãe, que sempre me deu todo o apoio, onde quer que eu esteja, ao meu irmão, que sempre se mostrou presente no processo de formação de meu caráter, e minha namorada, por em todos os momentos me dar toda a sua atenção e carinho. Agradeço também a todos os professores com os quais tive convívio. Sem os mesmos me faltaria a inspiração que tenho para seguir em frente rumo aos meus sonhos.

Presto também agradecimento a meu orientador, Giácomo, por ser sempre disponível, prestativo, encorajador e tão carismático, além de genial. Agradeço a meus colegas de sala que me acompanharam ao longo do curso: Fabrício e Guilherme, por compartilharem sempre palavras de apoio.

.

**“Vigie seus pensamentos, eles tornam-se palavras.**
**Vigie suas palavras, elas tornam-se ações.**
**Vigie suas ações, elas tornam-se hábitos.**
**Vigie seus hábitos, eles formam seu caráter.**
**Vigie seu caráter, ele se torna seu destino.”**

**Frank Outlaw**

## RESUMO

Em virtude do recente crescimento das resoluções e qualidades de imagem em vídeos digitais, o aumento gradativo da demanda de largura de banda para a transmissão se torna inviável, despertando a necessidade da compressão destes vídeos. A principal técnica utilizada para a compressão de vídeos consiste em eliminar a redundância de informações entre quadros consecutivos, substituindo-as por vetores de movimento que indicam o deslocamento de determinados *pixels* do quadro. Esta técnica é chamada estimação de movimento, método empregado para identificar o movimento do quadro atual em comparação com o quadro anterior, sintetizar estes vetores de movimento e utilizá-los, enfim, para a reconstrução o quadro desejado. Essa reconstrução ocorre somente com o quadro anterior e os vetores, permitindo que o quadro atual seja descartado. Tal prática reduz significativamente o peso do vídeo, mas eleva a complexidade computacional e o tempo de compressão. A fim de amenizar esta contrapartida, foi introduzida na literatura a técnica de *block-matching*, responsável por segmentar o quadro em blocos de determinado tamanho, compará-los com outros blocos e atribuir o mesmo vetor de movimento para todos os *pixels* deste bloco. Esta técnica é amplamente difundida pela área tecnológica em vista de sua simplicidade de implementação e baixa complexidade computacional, enquanto que produzindo resultados satisfatórios, perante o olho humano. Ao longo dos anos, vários algoritmos de *block-matching* foram elaborados, refinando cada vez mais seus critérios de seleção de blocos. Contudo, estes algoritmos podem tornar-se obsoletos muito rapidamente, por conta do desenfreado crescimento das resoluções de imagem. Portanto, este trabalho visa apresentar e comparar três algoritmos de *block-matching* desenvolvidos nos últimos três anos, supostamente adaptados para a resolução *full HD* (1920x1080), que são os algoritmos PAL (Pal, 2015), WUARPS (Wu; Huang, 2016) e PRO (Ziwei et al, 2017). Os resultados apresentados apontam um equilíbrio entre os três no que diz respeito a qualidade do quadro reconstruído, complexidade computacional e tempo empregado na execução. Em destaque, o algoritmo PRO apresentou, de todos, a melhor qualidade de imagem, enquanto que o WUARPS obteve o menor tempo de execução. Já o algoritmo PAL possui melhor desempenho em sequências de baixo grau de movimentação. É interessante que futuras pesquisas possam abordar a utilização de tamanhos variados de blocos, no lugar de tamanhos de bloco fixos.



**Palavras-chave**: Processamento de imagens, Seleção de blocos, Vetores, Compensação de movimento.

## ABSTRACT

*Due to the recent growth in resolution and image quality in digital videos, the excessive bandwidth need for its transmission turns out to be out of hand, paving the way for video compression technologies. The fundamental technique used for video compression is based on eliminating frames that are highly redundant, swapping them for motion vectors that point the offset pertaining to a frame's pixel. This method is called motion estimation and its main goal is to compare the displacement between the current and previous frame, generate motion vectors and use them, at last, to rebuild the target frame. The frame reassembly is comprised of the previous frame and the motion vectors, allowing the compressor to discard redundant information. Such approach significantly decreases the video's original size, but on the other hand, the computational complexity and compression time usage are notably increased. In order to attenuate this, the concept of block-matching was introduced, bringing forth the idea of segmenting the frame in blocks of a predetermined size, comparing them to other blocks and assigning the same motion vector to all of the block's pixels. This process is widely adopted by the academy on the grounds that it is simple to deploy and it is easy on the computational complexity, while rendering results that, for the human eye, are acceptable. Various block-matching algorithms have been developed over time, with each new entry refining its criteria for choosing the vector generating block. Yet, these algorithms tend to become obsolete very quickly, due to ever growing video resolutions. Therefore, this paper seeks to introduce and compare three algorithms developed in the past three years, supposedly suited for the full HD resolution. These algorithms are PAL (Pal, 2015), WUARPS (Wu; Huang, 2016) and PRO (Ziwei et al, 2017). Results following their executions show a balance between the three when it comes to frame quality, computational complexity and elapsed time for execution. The highlights are that the PRO algorithm showed the best image quality, whereas WUARPS took up the least time. The PAL algorithm happened to render the best results when the sequence had low amount of movement. An interesting subject to broach, in the future, would be to utilize dynamic block sizes, instead of fixed block sizes.*



***Keywords**: Image processing, Block selection, Vectors, Motion compensation.*

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ARPS | Adaptive Rood Pattern Search |
| CAVLC | Context-adaptive variable length coding |
| CODEC | Coder/Decoder |
| CD | Compact Disc |
| CS | Cross Search |
| DCS | Dual Cross Search |
| DS | Diamond Search |
| ES | Exhaustive Search |
| FS | Full Search |
| FSS | Four-Step Search |
| GOP | Group of Pictures |
| H.264 | Padrão de compressão de vídeo |
| HDTV | High-Definition Television |
| ICEEI | International Conference on Electrical Engineering and Informatics |
| IPTV | Internet Protocol Television |
| MAD | Mean Absolute Difference |
| MATLAB | Matrix Laboratory |
| MB | Megabyte |
| MPEG | Moving Pictures Experts Group |
| MSE | Mean Squared Error |
| NTSS | New Three-Step Search |
| OTS | One at Time Search |
| PSNR | Peak Signal-to-Noise Ratio |
| ROM | Read Only Memory |
| SES | Simple and Efficient Search |
| TSS | Three-Step Search |

**SUMÁRIO**

# 1 INTRODUÇÃO

Desde meados dos anos 1990, com a introdução do conceito da transmissão de vídeo em tempo real pela internet, a tecnologia utilizada para mover os dados de imagens e áudio de um ponto A da rede até o ponto B prosperou a passos tão largos que evoluiu a técnica de *streaming* de uma mera novidade a um dos recursos mais difundidos da sociedade moderna (CONKLIN et al., 2001, tradução nossa).

A transmissão em tempo real de vídeos de alta qualidade vem sendo um dos maiores focos da academia nas últimas duas décadas, em vista de seu elevado potencial de geração de renda. Aplicações voltadas ao *streaming* online tornaram-se, portanto, cada vez mais voltadas para a otimização de desempenho e qualidade.

Alguns exemplos destas aplicações poderiam compreender, por exemplo, conferências ao vivo num ambiente corporativo, em que a qualidade do vídeo, se comprometida, poderia causar uma impressão negativa no cliente; Serviços de televisão digital sob demanda (IPTV), em que o cliente pode assistir a basicamente qualquer filme ou série em alta definição de onde estiver, sem necessidade de realizar o download de um arquivo excessivamente pesado; Acesso livre de vídeos na web em alta qualidade, sem requerer da audiência muito tempo de espera para o carregamento (*buffering*) do transporte do arquivo; Seminários educativos, como *webinars* e palestras.

O envio desta carga de dados com alta fidelidade não é tão simples, no entanto. Vídeos digitais em alta definição possuem alta demanda de largura de banda, costumam apresentar *delay* na apresentação, podem sofrer interrupções e severa perda de qualidade em regiões que dispõem de má qualidade de conexão. Em virtude destes motivos, o desenvolvimento de mecanismos e arquiteturas com foco no aproveitamento de banda se faz cada vez mais necessário para amenizar estas adversidades (WU et al., 2001, tradução nossa).

A conciliação entre qualidade do vídeo transferido e a eficiência do uso de rede não teria o mesmo sucesso sem os contínuos esforços da comunidade acadêmica e de organizações para desenvolver novas técnicas, como a criação de servidores de vídeo dedicados, protocolos de envio, e em especial, a compressão de vídeo. (CONKLIN et al., 2001, tradução nossa).

A compressão de vídeo tem como base o princípio de que os quadros dentro de um vídeo apresentam muitas informações redundantes, cuja ausência não é perceptível pelo olho humano. O objetivo da compressão é eliminar estas redundâncias, conciliando desempenho e qualidade de vídeo (ROSA et al, 2007). Os esforços para otimizar a compressão no modelo temporal são importantes, dado que 80% do processamento da compressão ocorre na eliminação de informações redundantes entre quadros sucessivos (KUHN, 1999, tradução nossa).

Atendendo a este problema de processamento excessivo, uma técnica eficaz e amplamente empregada nos CODECs de vídeo, desde os mais rudimentares até os mais robustos no mercado é o *block-matching.* Esta técnica é utilizada em padrões reconhecidos mundialmente, como a série H.264 e MPEG, por ser simples, rápida e efetiva (ZHU; MA, 1997, tradução nossa).

O objetivo do *block-matching* consiste em dividir um quadro em blocos e compará-los com blocos da imagem anterior, com o objetivo de gerar vetores de movimentação, responsáveis por mover apenas algumas partes da imagem original para formar a imagem sucessora, sem uso de informações da mesma. A realização desta técnica evita a necessidade de enviar informações redundantes pela rede, como cenários e objetos parados. O *block-matching* efetua a divisão do quadro atual em blocos 16x16 ou 8x8. Isso ocorre, pois, para a técnica, a movimentação dos pixels dentro de um bloco tende a ser de caráter idêntico (KUHN, 1999, tradução nossa; KINTSCHNER et al, 2011; CHOUDHURY, SAIKIA, 2014, tradução nossa).

Existem muitos algoritmos de *block-matching* na literatura*,* empregados em diversos CODECs, equilibrando complexidade, dificuldade de implementação, desempenho, escalabilidade e qualidade da imagem produzida (AL-NAJDAWI et al., 2016, tradução nossa). Alguns exemplos difundidos pela literatura são: Exhaustive Search (ES); Three Step Search (TSS); New Three Step Search (NTSS); Simple and Efficient Search (SES) e Adaptive Rood Pattern Search (ARPS).

No entanto, estes algoritmos datam de uma época em que não existia a definição *full* HD (1920x1080), que requer que muitos pontos de pesquisa sejam avaliados para a realização da estimação de movimento (RAO, MURALIDHAR, RAO, 2014, tradução nossa), que pode comprometer tanto a qualidade da imagem quanto o uso de processamento, por considerar o mínimo local como sendo a solução ótima (TURAGA; ALKANHAL, 1998, tradução nossa)

Em vista deste argumento, a proposta do presente trabalho de conclusão de curso possui como finalidade realizar a análise comparativa dos resultados das aplicações de três algoritmos desenvolvidos nos últimos três anos com foco na resolução *full* HD para compressão de vídeo por *block-matching,* com base em métricas avaliativas validadas pela academia.

## 1.1 OBJETIVO GERAL

Realizar uma análise do dispêndio de recursos e qualidade de imagem resultante da aplicação dos algoritmos de compressão de vídeo por meio da estimação de movimento por *block-matching* propostos nos últimos três anos, especificamente para a definição *full* HD.

## 1.2 OBJETIVOS ESPECÍFICOS

A. Documentar a base sobre compressão de vídeo;
B. Descrever o método de estimação e compensação de movimento por meio da técnica de *block-matching*;
C. Levantar algoritmos propostos nos últimos três anos que sugerem otimizações e melhorias em técnicas já conhecidas pela literatura para a definição *full* HD;
D. Submeter os algoritmos avaliados a métricas para análise do desempenho computacional.
E. Analisar e documentar os resultados dos testes.

## 1.3 JUSTIFICATIVA

As técnicas para compressão de vídeo focam na redução de redundâncias estatísticas nos dados e na exploração das limitações do sistema visual do ser humano. Posto que as informações de luz e cor em vídeos possuem maior grau de similaridade no sentido de movimento, a remoção de redundância é dependente da estimação de movimento (DUFAUX; KONRAD, 2000, tradução nossa).

Os algoritmos de estimação de movimento por *block-matching* estão entre os mais difundidos e acolhidos pelo mercado, uma vez que são fáceis de entender, de simples implementação e em virtude de sua aplicação em padrões difundidos, como a série H.261 até H.264 (HAMID et al, 2014, tradução nossa).

O conceito por trás da tecnologia de *block-matching* se baseia na divisão do quadro atual em uma matriz de blocos, que são comparados com seus blocos correspondentes e adjacentes do quadro anterior. Esta comparação, em seguida, gera um vetor que estipula o movimento deste bloco de um local para o outro, criando a relação entre quadro anterior e atual (BARJATYA, 2004, tradução nossa).

Os algoritmos atualmente difundidos pela academia não consideram que a resolução *full* HD demanda muitas comparações para a geração deste vetor de movimento, gerando a necessidade de acolhimento de novos algoritmos, robustos e econômicos (RAO, MURALIDHAR, RAO, 2014, tradução nossa).

A comparação entre blocos utiliza uma função de custo como critério para reconhecer se determinado bloco é mais adepto a servir de modelo para a realização do cálculo do vetor. Ou seja, seria selecionado o bloco com menor custo proveniente da métrica utilizada (BARJATYA, 2004, tradução nossa).

A métrica adotada neste trabalho para indicar a similaridade entre blocos comparados será o cálculo da Diferença Média Absoluta (MAD), por ser o método de menor complexidade computacional e o mais disseminado entre trabalhos que avaliam o desempenho em amostragem de vídeos sujeitos a técnicas de estimação de movimento (VASSILIADIS et al., 1998, tradução nossa).

## 1.4 ESTRUTURA DO TRABALHO

Este trabalho divide-se em seis capítulos. O primeiro capítulo apresenta justificativas para a comparação a ser realizada, além de breves descrições do funcionamento das técnicas descritas. O segundo capítulo fornece uma descrição sobre o modelo de eliminação da redundância temporal na compressão de vídeo seguindo o fluxo de compressão de padrões de compressão comuns entre a série H.261 até H.264. No terceiro capítulo, a técnica de *block-matching* é apresentada em destaque a outros métodos de eliminação de redundância temporal, apresentando motivos para a escolha do mesmo e o conceito básico por trás da

seleção de blocos e geração dos dados para a compressão dos quadros. Em sequência, serão apresentados alguns trabalhos visando essencialmente à mesma proposta de comparar algoritmos de compressão por *block-matching*. No quinto capítulo, serão levantados os algoritmos a serem submetidos às métricas avaliativas e uma breve descrição sobre seu conceito e funcionalidade. Por fim, nos capítulos seis e sete serão apresentados os modos como o cenário de testes foi implantado e os resultados obtidos. No capítulo dos resultados foram abordadas as métricas em relação aos dados retirados da execução da aplicação e foi realizada uma análise com base em gráficos e tabelas, relacionando desempenho e qualidade.

# 2 FUNDAMENTOS DA COMPRESSÃO DE VÍDEO

Vídeos costumam demandar uma grande quantidade de dados para transferência. Kintschner et al. (2011) contextualiza este problema, evidenciando que uma sequência de vídeo em *full* HD (1920x1080 *pixels*), utilizando um formato 4:2:0 a 30 fps, ocupa aproximadamente 746 Mbits por segundo de banda. O transporte desta carga, utilizando um serviço de provedor comum de 1MB/s é irrealizável. Para tanto, a etapa de compressão (codificação) de um vídeo antes de seu envio é fundamental.

Neste processo, um arquivo de vídeo que originalmente demandaria um largo espaço de banda é reduzido a uma sequência de dados codificados de tamanho razoável, que em seguida é reconstruído no nó de destino, seja ele transmitido via web ou até mesmo arquivamento local (RICHARDSON, 2003, tradução nossa).

Para Richardson (2003, tradução nossa), os métodos de compressão e descompressão, se aplicados com perícia, podem servir como determinante de liderança do produto no mercado, por oferecer melhor qualidade de imagem, maior confiança na entrega dos dados e maior flexibilidade de transmissão do que outras soluções disponíveis. A intensidade de pesquisa na área é acirradamente disputada entre o setor de entretenimento, comunicação, desenvolvedores de hardware aplicado e criadores de algoritmos patenteados potencialmente lucrativos.

Diversos grupos de pesquisa aplicada no ramo de telecomunicação elaboraram, com o passar dos anos, padrões de compressão cada vez mais avançados. Um dos mais rudimentares a serem introduzidos na indústria foi o padrão MPEG-1, criado pela Moving Pictures Experts Group (MPEG) (AHMAD; HE; LIOU, 2001, tradução nossa). Este padrão buscava a solução do desafio de armazenar vídeo em um CD-ROM de apenas 1.4Mbps (WOOTTON, 2005, tradução nossa). Mais tarde, foram publicados padrões utilizando tecnologias mais robustas, como o H.264/AVC, e mais recentemente o H.265/HEVC.

Para que se possam analisar as diversas técnicas de compressão existentes, é necessário observar uma importante característica presente nos vídeos, que é o fato destes apresentarem muitas informações redundantes, sejam elas espaciais (um quadro composto por informações de cunho redundante, como o

céu azul, por exemplo) ou temporais (pouca movimentação entre seus quadros, como um objeto parado) (ROSA et al., 2007).

A grande maioria das técnicas de codificação explora essas características, suprimindo informações redundantes cuja ausência não compromete totalmente a qualidade do vídeo, pelo fato de não serem perceptíveis pelo olho humano.

## 2.1 REDUNDÂNCIA TEMPORAL

Dentro de um intervalo de tempo, um vídeo pode conter muitos quadros praticamente iguais uns aos outros. Um meio utilizado para reduzir o espaço ocupado por estes quadros é identificar a redundância temporal e, por meio de cálculos de estimação de movimento, render vetores de movimentação (KUHN, 1999, tradução nossa). Estes servem para mover apenas algumas partes pontuais da imagem de referência necessárias para formar a imagem seguinte, e possuem um peso correspondente a apenas uma fração do quadro original.

### 2.1.1 Tipos de quadro

O padrão MPEG-1 define três tipos de quadros (WOOTTON, 2005, tradução nossa), que se encontram listados a seguir:

a) *I-Frame:* São os quadros de referência para servir de base para a predição de movimento. Este tipo de quadro guarda os dados brutos da imagem. Por este motivo, este tipo de quadro é o mais pesado, porém necessário em caso de introdução de novos objetos em cena ou mudança total dela. Empregar *I-frames* frequentemente em uma sequência do vídeo é importante também em caso de perda de pacotes, ao transmiti-lo via web (KONDOZ, 2009, tradução nossa). Um vídeo não comprimido é composto somente por *I-frames* (ou seja, somente imagens);
b) *P-Frame:* Este tipo de quadro faz proveito de informações do quadro que o antecede. É criado pela técnica de estimação de movimento, e reduz drasticamente a redundância temporal. *P-frames*, por conterem

somente informações de movimento relativo (não contém nenhum dado de imagem), podem pesar até três vezes menos do que seu quadro de referência (CARNE, 2011, tradução nossa);

c) *B-Frame:* São quadros similares aos *P-frames* no sentido de utilizarem somente vetores de movimento relativos ao quadro de referência (seja este um *I-frame* ou um *P-frame*). O diferencial se dá em sua característica bidirecional, bem como a letra 'B' indica, ou seja: não somente carrega informações referentes a quadros anteriores, mas posteriores também. Este tipo de quadro pode pesar até seis vezes menos do que um *I-frame* (CARNE, 2011). Em contrapartida, o custo de memória despendido para o processamento deste tipo de quadro pode se tornar inviável em aplicações que requerem baixo custo de banda, como videoconferência (KONDOZ, 2009, tradução nossa).

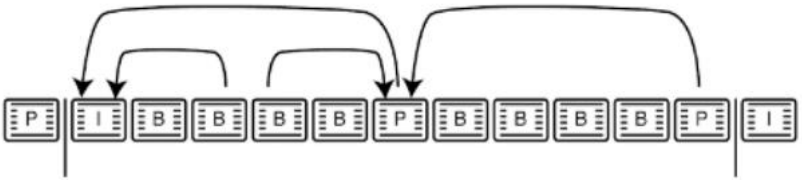
### 2.1.1 Grupos de imagens

Em virtude da exigência de que sejam inseridos *I-frames* periodicamente, se faz necessário que o vídeo a ser comprimido seja segmentado em grupos chamados de Grupos de Imagens (do inglês *Group of Pictures, GOP*). O mesmo faz parte da definição do padrão MPEG-1 e é a primeira etapa da compressão de vídeo (WOOTTON, 2005, tradução nossa).

Um GOP geralmente é formado por uma sequência de 10 a 30 quadros. O primeiro quadro presente em um destes grupos será sempre um *I-frame*, dado que é imperativa a existência de um quadro de referência para construção dos vetores de movimento. Posteriormente, serão posicionados os quadros contendo informações de movimento como *B-frames* e *P-frames*, seguindo a lógica de que no meio e no fim de um GOP deverá conter um *P-quadro* (WOOTTON, 2005, tradução nossa). A figura 1 ilustra, graficamente, como é formado um GOP.

Este perfil de organização de quadros ameniza o problema de perda de pacotes de transmissão e atenua a propagação de erros entre *P-frames* e *B-frames* sucessivos.

Figura 1 - Exemplo de estrutura de um GOP

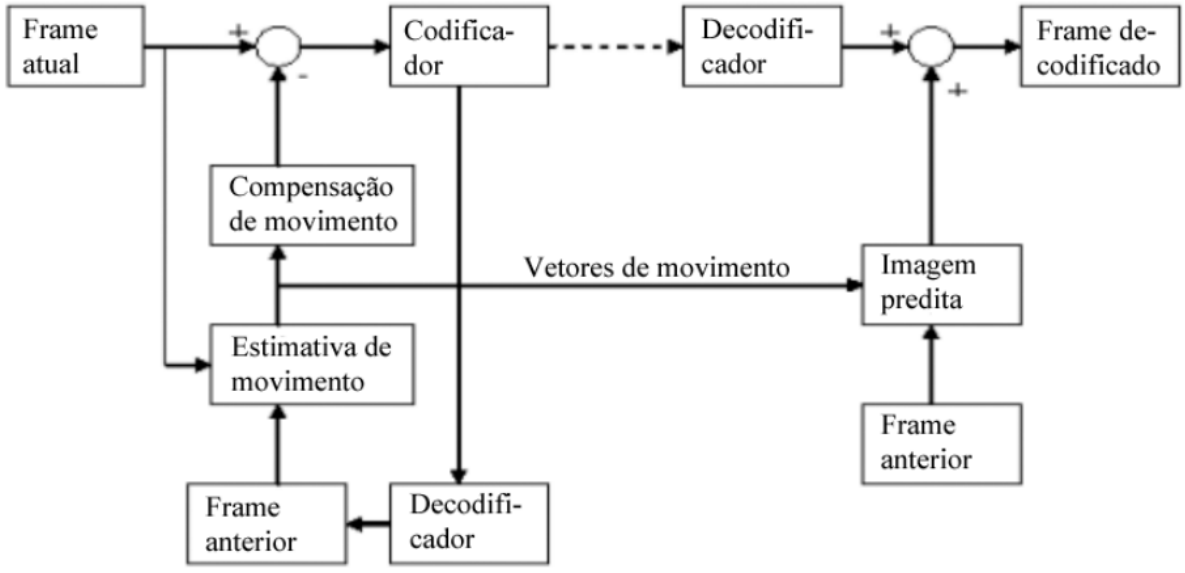



Fonte: Wootton (2005, adaptado pelo autor).

## 2.2 FLUXO DA COMPRESSÃO

Um vídeo, ao ser submetido à técnica de compressão, deve passar por um fluxo contendo várias etapas antes de seu envio ao destino. O fluxo contemplado neste trabalho é comum aos padrões MPEG-1, MPEG-2, MPEG-4, H.261, H.263 e H.264 (AHMAD; HE; LIOU, 2001, tradução nossa) e está disposto na figura 2. A mesma ilustra a existência de um fluxo específico para a etapa de codificação e um para a de decodificação.

Figura 2 - Estrutura genérica do fluxo de compressão de padrões MPEG




Fonte: Kulkarni, Bormane e Nalbalwar (2015, adaptado pelo autor).

### 2.2.1 Estimação e compensação de movimento

A primeira etapa a ser executada no lado do codificador é a de estimação de movimento, cuja responsabilidade é identificar as diferenças entre o quadro de

referência e o quadro seguinte em divisões de área chamadas macro blocos e gerar os vetores de movimento, que serão P ou B *frames* subsequentes.

A próxima etapa, contemplada tanto pelo codificador como o decodificador é a de compensação de movimento, em que os vetores de movimento, gerados pela etapa anterior, são recebidos e a imagem original é reconstruída a partir de blocos do quadro de referência (I-*frame*) movidos por estes vetores.

Esta reconstrução pode não ser completamente fiel à imagem original, no entanto. Por mais que a compressão tenha como objetivo a reprodução perfeita do quadro, ele raramente irá reproduzir com exatidão o quadro original. Por isso, o quadro gerado pelos vetores de movimento é comparado com seu quadro de referência para determinar o coeficiente de erro de predição (AHMAD; HE; LIOU, 2001, tradução nossa). A partir deste coeficiente, serão codificadas as distinções para que se possam efetuar correções pontuais no quadro reconstruído (LE GALL, 1992, tradução nossa).

A figura 3 ilustra o processo de correção da predição. A imagem à esquerda apresenta o quadro original da sequência "*Flower Garden*", enquanto que a figura à direita retrata o quadro contendo somente as diferenças entre o quadro original (de referência) e o gerado pela etapa de compensação de movimento.

Figura 3 - Quadro de diferença gerado pela compensação de movimento


Fonte: Tham et al (1998)

### 2.2.2 Envio ao destinatário

Finalmente, os vetores de movimento, as informações de cor e os *I*-quadros são combinados numa sequência de bits, que por sua vez é comprimida

utilizando quantização *lossless* em Context-adaptive variable length coding (CAVLC), algoritmo em que os dados para transmissão são reduzidos em espaço sem perder informações (LAO; CHOU; CHUNG, 2001, tradução nossa).

Neste fluxo, a sequência recebida é decodificada e a imagem original é reconstruída com base nos vetores de movimento alimentados pelo fluxo inicial. A distinção entre os fluxos supracitados é que o decodificador não realiza a etapa de estimação de movimento, somente a de compensação de movimento (AHMAD; HE; LIOU, 2001, tradução nossa).

## 2.3 ESTIMAÇÃO DE MOVIMENTO

Para compreender o conceito básico da estimação de movimento, é interessante atentar para uma definição básica sobre vetores, estabelecida pela física: vetores são utilizados para identificar a movimentação de um determinado objeto de um ponto A ao B, descrevendo informações de direção e magnitude do movimento (PHILIP et al, 2014, tradução nossa).

Basicamente, a estimação de movimento pode ser descrita como um processo cuja entrada é um quadro de referência e que a saída são vetores responsáveis por ilustrar o movimento de blocos de um quadro ao próximo (CARNE, 2011, tradução nossa).

### 2.3.1 Métodos de estimação de movimento

Os métodos de estimação de movimento elaborados desde sua concepção inicial podem ser classificados entre diretos e indiretos. O método direto é orientado por pixels, como *Block-matching, pixel recursive, Phase-correlation* e fluxo ótico, caracterizados por realizar comparações entre quadros subsequentes e gerar vetores de movimento. Já os métodos indiretos empregam técnicas como análise de tonalidade, informações presentes em extremidades e cálculos estatísticos (PHILIP, 2014, tradução nossa).

Duas técnicas fundamentais que podem ser destacadas dentre a variedade de métodos diretos de estimação de movimento são a de *pixel recursive e block-matching*. O contraste entre as duas técnicas se dá no modo como as duas avaliam os pixels à sua volta para a geração de *P e B*-frames.

A técnica de *pixel recursive* compara a movimentação de todos os pixels do quadro individualmente utilizando uma técnica de gradiente, enquanto que a de *block-matching* divide o quadro atual em macro blocos e assume que todos os pixels dentro deste bloco possuem a mesma informação de movimento.

Pelo fato da técnica de *pixel recursive* analisar todos os pixels separadamente, ela se torna mais custosa em termos de processamento, (KULKARNI; BORMANE; NALBALWAR, 2015, tradução nossa), desviando, portanto, atenção especial do mercado para a técnica de *Block-matching*. A preferência pelo *Block-matching* também é justificável em virtude de sua simplicidade de implementação e aplicação em padrões difundidos, como a série H.261 até H.264 (HAMID et al, 2014, tradução nossa).

## 2.4 A TÉCNICA DE *BLOCK-MATCHING*

O conceito principal por trás da técnica de *block-matching* consiste em dividir o quadro atual numa matriz de blocos chamados macro blocos. Esta divisão parte do princípio de que todos os pixels dentro deste bloco sofrerão o mesmo grau de deslocamento, e são, portanto, passíveis de serem tratados como uma só entidade (CHOUDHURY, SAIKIA, 2014, tradução nossa).

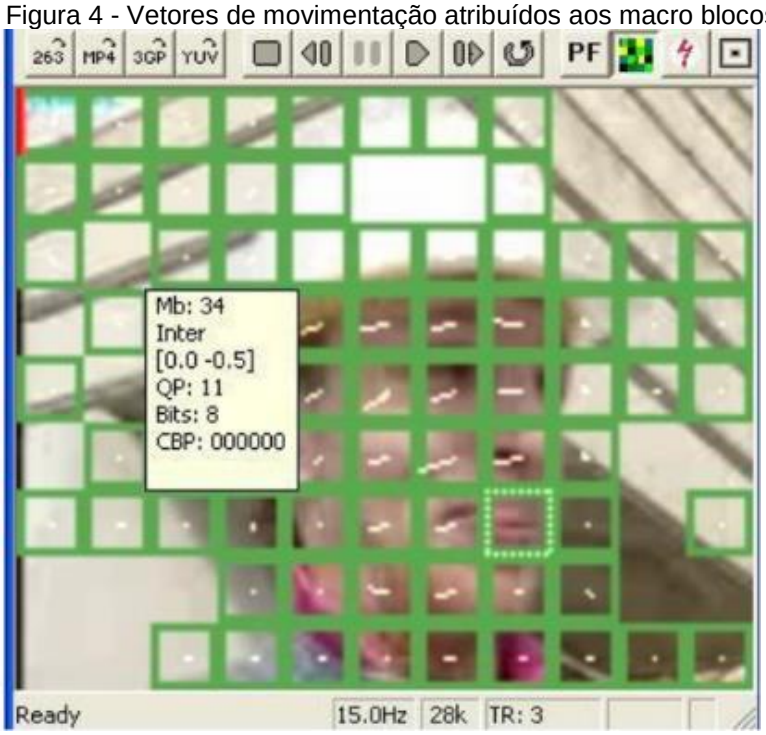
Estes macro blocos são comparados com blocos vizinhos adjacentes do quadro anterior, processo que deverá ter como produto um vetor de movimento, responsável por definir de onde serão “emprestados” os *pixels* pertencentes ao bloco no momento da reconstrução da imagem (PHILIP, 2014, tradução nossa).

Segundo definição dos padrões MPEG-1, MPEG-2, MPEG-4, H.263 e H.264 (KINTSCHNER et al, 2011), a comparação entre os blocos ocorre no que é chamada uma janela de pesquisa, em que os mesmos, divididos em 16x16 pixels recebem uma variável *p*, que designa o alcance de comparação entre o bloco atual e os adjacentes, a fim de encontrar o bloco de maior similaridade. Quanto maior for o deslocamento entre o bloco atual e o bloco ideal, maior deverá ser a variável *p,* com o custo de maior complexidade de processamento (PHILIP, 2014, tradução nossa).

Todos os blocos da imagem recebem uma janela de pesquisa e são comparados com blocos adjacentes candidatos. O bloco candidato de maior similaridade, ou seja, com menor nível de distorção em relação ao bloco atual é

utilizado para a geração do vetor de movimento (KULKARNI; BORMANE; NALBALWAR, 2015, tradução nossa), conforme pode ser visto na figura 4.

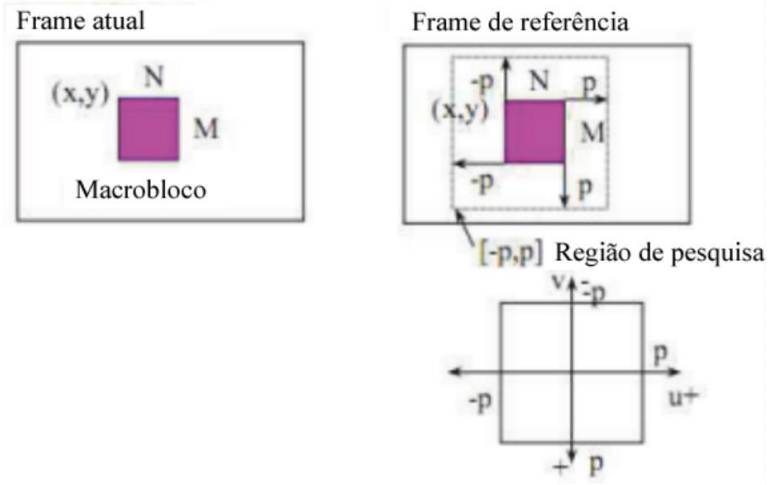
Figura 4 - Vetores de movimentação atribuídos aos macro blocos




Fonte: Shahid (2010)

A figura 5 retrata as características básicas inerentes do procedimento de geração dos vetores. No primeiro quadro da primeira coluna pode-se observar um macro bloco de tamanho N x M *pixels*, em uma determinada posição [x, y]. A segunda coluna ilustra a janela de pesquisa com alcance *p*, partindo da posição do macro bloco, se estendendo em *p* para as quatro direções.

Figura 5 - Macroblocos e vetores de movimento



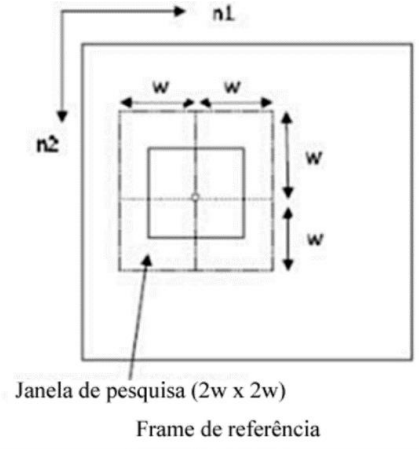

Fonte: Kintschner et al (2011, adaptado pelo autor)

### 2.4.1 Janela de pesquisa

Em um algoritmo comum de *block-matching,* a janela de pesquisa em que ocorrem as comparações entre macro blocos possui lado de tamanho *2p + 1* (CUEVAS et al., 2013, tradução nossa) e o alcance de pesquisa em uma janela segue o padrão *p = 7,* conforme definição proposta por Takaya (2006, tradução nossa). Esta distância é projetada partindo do bloco central, ou, o bloco que está sendo comparado no momento, para as quatro direções.

Seguindo estes parâmetros, pode-se concluir que uma janela de pesquisa tem em potencial 225 blocos de comparação (LI; ZENG; LIOU, 1994, tradução nossa). Um esboço de como seria construída uma janela de pesquisa pode ser visto na figura 6.

Figura 6 - Dimensões de uma janela de pesquisa




Fonte: Selvakumar e Manikandan (2014, adaptado pelo autor)

### 2.4.2 Métricas avaliativas de comparação entre blocos

Para determinar qual dos blocos na janela de pesquisa é o ideal para a síntese dos vetores de movimento deve-se empregar uma função de custo de cálculo estatístico, cujo objetivo é observar o grau de distorção, isto é, a quantidade de distinções entre os pixels dos blocos em foco de comparação (PHILIP, 2014, tradução nossa).

Em meio às funções de custo disponíveis na literatura, duas das mais difundidas são o *Mean Absolute Difference* (MAD) e *Mean Squared Error* (MSE) (METKAR; TALBAR, 2013), a primeira sendo a mais popular e de menor complexidade computacional, por não envolver cálculos de potenciação e raiz, em contraste com a MSE (PHILIP, 2014, tradução nossa; LUO et al., 1997, tradução nossa). Pelo presente motivo, o cálculo da MAD será utilizado neste trabalho no lugar dos demais, disponíveis na literatura.

#### 2.4.2.1 Mean Absolute Difference

O cálculo da MAD, ou Diferença Média Absoluta, em português, itera por todos os pixels do macro bloco, analisando as informações de intensidade do pixel (sem levar em consideração a cor, ou seja, em escala de cinza) e as comparando com seu correspondente do quadro de referência, a fim de trazer como resultado a diferença entre os dois blocos (TAKAYA, 2006, tradução nossa).

O MAD pode ser expresso pela equação (1), conforme definição retirada de Lu e Liou (1997, tradução nossa):

$$MAD(x,y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |F_c(k+i, l+j) - F_p(k+x+i, l+y+j)| \qquad (1)$$

Em que $F_c$e $F_p$ representam a intensidade do pixel no quadro original e atual codificado, respectivamente, $k$ e $l$ são as coordenadas do canto esquerdo superior do bloco atual e $x$ e $y$ representam o deslocamento em pixels relativo à posição do bloco atual.

Após checar cada pixel de cada bloco na área de pesquisa, o vetor de movimento é determinado como o $x$ e $y$ na posição do bloco em que o MAD possuir

o menor valor, ou seja, o menor índice de erro. *N* define o tamanho da janela de pesquisa.

2.4.2.2 Peak Signal-to-Noise Ratio

Outra função de custo utilizada é chamada *Peak Signal-to-Noise Ratio* (PSNR), traduzido como razão sinal-ruído de pico, função aplicada na imagem recriada pelos vetores de movimento no processo de compensação de movimento (PHILIP et al, 2014, tradução nossa).

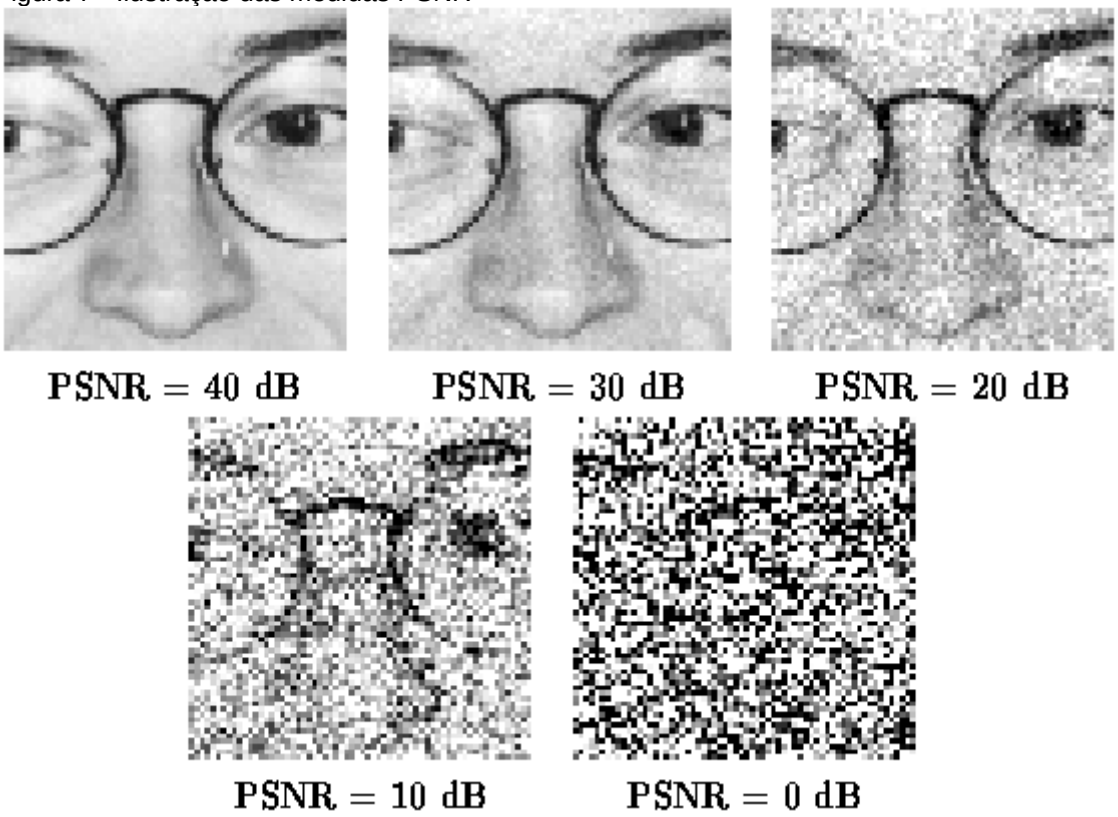
Esta função designa a qualidade da imagem compressa. Quanto maior for o resultado do PSNR, melhor deverá ser a qualidade da imagem recriada em comparação com o quadro original, no caso, não comprimido (HAMID et al, 2014, tradução nossa).

O cálculo do PSNR pode ser expresso pela equação (2), dado que $N$ e $M$ sejam as dimensões do bloco e $x$ e $x'$ sejam a intensidade do pixel em foco (HAMID et al, 2014, tradução nossa):

$$PSNR = 10log_{10}\left(\frac{255^2}{\frac{1}{NM}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|x_{ij}-x'_{ij}|^2}\right) \quad (2)$$

O PSNR destaca a diferença entre o *pixel* recriado e o original. A diferença entre os dois dá origem ao divisor, que será inversamente proporcional ao valor máximo do pixel em questão. No caso da equação (2), serão 8 bits, ou seja, 255. O PSNR é medido em decibéis (dB) e sua utilização para a avaliação da qualidade da imagem pode ser vista na figura 7 (VELDHUIZEN, 1998, tradução nossa).

Figura 7 - Ilustração das medidas PSNR




Fonte: Veldhuizen (1998)

# 3 ALGORITMOS DE BLOCK-MATCHING DA LITERATURA

O algoritmo mais primitivo de comparação entre blocos dentro de uma janela de pesquisa se chama Busca Exaustiva, do inglês *Exhaustive Search* (ES). Também conhecido como *Full Search* (FS), sua função é percorrer todos os blocos da janela à procura do bloco de maior similaridade (LUO et al., 1997, tradução nossa).

Este método é o ideal para localizar o melhor bloco para a geração do vetor pelo fato de gerar o maior valor PSNR, em relação a outros métodos. No entanto, este se torna inviável em aplicações que dispõem de banda limitada de dados, pois sua complexidade computacional é demasiadamente alta, chegando a ser $O(n)$ (PHILIP, 2014, tradução nossa).

Considerando o elevado uso de memória desta técnica, vários algoritmos de *Block-matching* que o sucederam sugeriram mudanças no modo como são tratadas as comparações entre blocos. Serão apresentados neste capítulo alguns algoritmos que datam de dez a vinte anos atrás.

## 3.1 THREE STEP SEARCH

Uma das primeiras e mais conhecidas técnicas apresentadas para a solução de tal problema é o Three Step Search (TSS), proposto por Koga et al (1981 apud KAMBLE; THAKUR; BAJAJ, 2017, tradução nossa). O mesmo reduz radicalmente o custo computacional nos cálculos de MAD e limita a procura de pontos na janela de pesquisa em comparação, potencialmente tornando a complexidade do algoritmo $O(log_n)$ (PHILIP, 2014, tradução nossa).

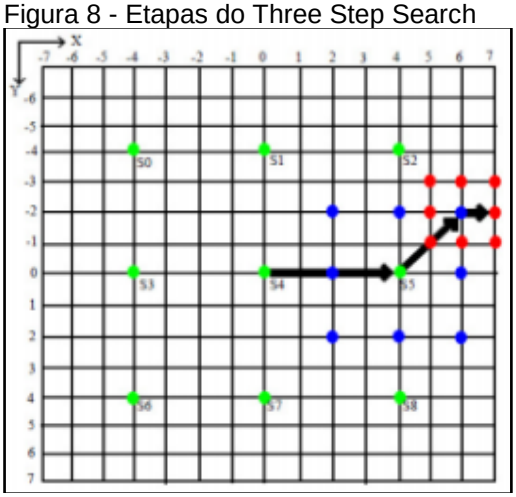
O algoritmo inicia sua primeira etapa estabelecendo uma janela de pesquisa no centro do bloco a ser comparado. Em seguida, o seguinte fluxo, detalhado por Kamble, Thakur e Bajaj (2017, tradução nossa) é executado:

a) dispersar oito pontos ao redor do ponto de origem, respeitando uma distância regular entre cada um. Em seguida, realizar o cálculo de custo MAD sobre cada um destes pontos com o intuito de observar seu grau de distorção;

b) deslocar o ponto de origem para o ponto de menor distorção entre os oito primeiros e dividir pela metade a distância entre os próximos pontos a serem espalhados;

c) repetir os passos 1 e 2 sucessivamente até a distância entre os pontos chegar a 1. Neste momento, o bloco de menor distorção dentro da sub janela será utilizado para a geração do vetor de movimento. A figura 6 ilustra graficamente o funcionamento de cada etapa do algoritmo, dado que a distância inicial entre os pontos seja de 4 blocos:

Figura 8 - Etapas do Three Step Search




Fonte: Kulkarni, Bormane e Nalbalwar (2015).

Enquanto o algoritmo de busca exaustiva realiza 225 comparações entre blocos, este método restringe o número de comparações para apenas 25, reduzindo o impacto no processamento da etapa de estimação de movimento (KAMBLE; THAKUR; BAJAJ, 2017, tradução nossa).

Todavia, existe um ponto negativo intrínseco de algoritmos que limitam a quantidade de comparações, como o TSS. Pelo fato do primeiro passo efetuar comparações com somente oito pontos da janela, a possibilidade de o algoritmo localizar apenas o mínimo local como grau de distorção é muito forte. Encontrar somente o mínimo local significa tomar como escolha final um bloco que somente parece que é, entre todos, o mais similar, sendo que em outro ponto da janela existe um bloco mais similar ao comparado, chamado de mínimo global.

Isto significa que o algoritmo é ineficaz para detecção de movimentos sutis, tais como de altas resoluções. Adicionar mais pontos de pesquisa na primeira etapa aumenta as chances de o algoritmo encontrar o mínimo global. Porém, como

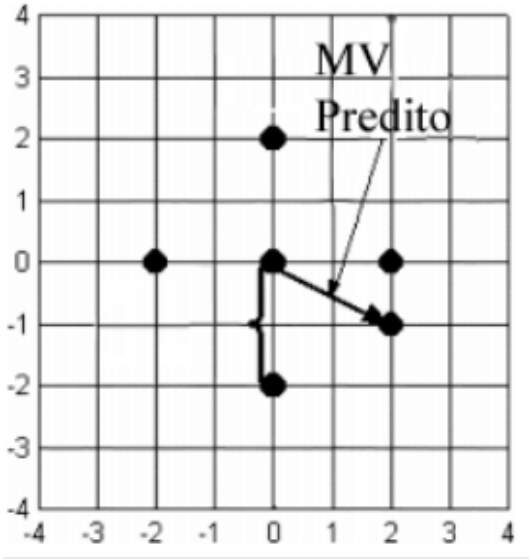
consequência, essa medida eleva a complexidade computacional. (LAKAMSANI; ZENG; LIOU, 1996, tradução nossa; PANDIAN; BALA; ANITHA, 2011, tradução nossa).

## 3.2 ADAPTIVE ROOD PATTERN SEARCH

Para amenizar o problema descrito no TSS, o algoritmo Adaptive Rood Pattern Search (ARPS) foi introduzido. Desenvolvido pro Nie e Ma (2002), o mesmo possui duas etapas: a primeira, responsável por realizar uma pesquisa abrangente, com o objetivo de se aproximar do mínimo global, chamado Large Diamond Search Pattern (LDSP) e a segunda, incumbida de engajar-se numa etapa de refinamento, utilizando tamanhos de pesquisa curtos até que o ponto de menor MAD seja o ponto central da pesquisa, chamado Short Diamond Search Pattern (SDSP).

Na primeira etapa do ARPS, são espalhados seis pontos na janela de pesquisa: cinco destes ilustrando o formato de uma cruz (incluindo o ponto central), e o sexto sendo o bloco encontrado pelo vetor de movimento emprestado do bloco imediatamente à esquerda do bloco comparado. Esta configuração pode ser ilustrada conforme a figura 9:

Figura 9 - Primeira etapa do algoritmo ARPS




Fonte: Nie e Ma (2002)

Isso se baseia no princípio de que a maioria dos blocos possui um vetor de movimento similar aos que estão ao seu redor (NIE, MA, 2002, tradução nossa). O funcionamento inicial do ARPS pode ser visto na figura 13.

O cálculo do MAD é realizado em todos estes seis pontos, revelando, ao final dos cálculos, qual destes é o mais similar, que servirá como ponto central para a etapa de refinamento. A etapa de refinamento distribui pontos ao redor do novo centro estabelecido pelo primeiro passo, também em formato de cruz. Nesta etapa, deve-se encontrar o bloco de menor distorção que melhor deverá se aproximar do mínimo global.

## 3.3 DEMAIS ALGORITMOS

Ainda se atendo ao escopo da literatura que permeia o fim dos anos 1990 e início dos anos 2000, é possível enumerar outros algoritmos de *block-matching* além do TSS e do NTSS, como Four Step Search (FSS) (PO; MA, 1996) e Diamond Search (DS) (THAM et al., 1998).

A tabela 1, retirada do trabalho proposto por Florêncio (2015), relaciona cada um dos algoritmos citados acima e define suas vantagens e desvantagens. Já a tabela 2 fornece uma relação entre os algoritmos e a quantidade de pontos de procura necessários para a localização do bloco de maior similaridade.

Tabela 1 - Vantagens e desvantagens dos algoritmos de *block-matching* da literatura

| Algoritmo | Vantagens | Desvantagens |
|---|---|---|
| TSS | Simples e robusto, e próximo do desempenho ótimo; Melhores vetores de movimento para um comportamento padrão; | Pouca eficiência para estimar movimentos curtos; |
| NTSS | Melhor estimação de movimento comparado aos algoritmos TSS e DS devido à boa estimação de movimentos curtos; | Tempo de processamento maior do que os algoritmos TSS e DS; |
| FSS | Mais robusto, comparado aos algoritmos TSS e NTSS; | Maior tempo de processa-mento, comparado aos outros algoritmos; |
| DS | Menor tempo de processamento. | Não é eficiente em implementação em hardware. |

Fonte: Florêncio (2015)

Tabela 2 - Quantidade de pontos necessários para a localização do bloco de maior similaridade entre os algoritmos de block-matching da literatura

| Algoritmo | Min. Pontos de comparação | Máx. Blocos comparados |
|---|---|---|
| FS | 1 | 225 |
| TSS | 25 | 25 |
| NTSS | 17 | 33 |
| FSS | 17 | 27 |
| DS | 9 | 24 |

Fonte: Choudhury e Saikia (2014); Selvakumar e Manikandan (2014)

# 4 ALGORITMOS COMPARADOS

No decorrer da década de 1990, muitos algoritmos de *block-matching* foram elaborados com a finalidade de reduzir a carga de processamento na detecção do bloco ideal para geração dos vetores de movimento.

A extensa variedade de algoritmos de *block-matching* disponíveis no mercado torna a decisão de qual deles utilizar uma questão de equilíbrio entre complexidade, dificuldade de implementação, desempenho, escalabilidade e qualidade da imagem produzida (AL-NAJDAWI et al., 2016, tradução nossa).

Os algoritmos citados no capítulo anterior são difundidos amplamente pela academia, mas datam de uma época em que não existiam *displays* em *High Definition* (do inglês alta definição, HD).

Mesmo versáteis e ágeis para aplicações de baixa disponibilidade de banda, estes algoritmos não são adequados para os movimentos complexos e minuciosos da definição *full* HD, que requerem que a janela de pesquisa seja extensa demais para a realização da estimação de movimento (RAO, MURALIDHAR, RAO, 2014, tradução nossa). A utilização destes algoritmos tende a convergir os cálculos de similaridade para o mínimo local, comprometendo tanto a qualidade da imagem quanto o uso de processamento do mesmo (TURAGA; ALKANHAL, 1998, tradução nossa).

Em virtude deste detalhe, neste trabalho serão avaliados algoritmos de compressão de vídeo por *block-matching* encorajados pela busca de um modo de melhorar a qualidade da compressão para vídeos em *full* HD e reduzir a complexidade computacional.

## 4.1 AN OPTIMIZED BLOCK MATCHING ALGORITHM FOR MOTION ESTIMATION USING LOGICAL IMAGE

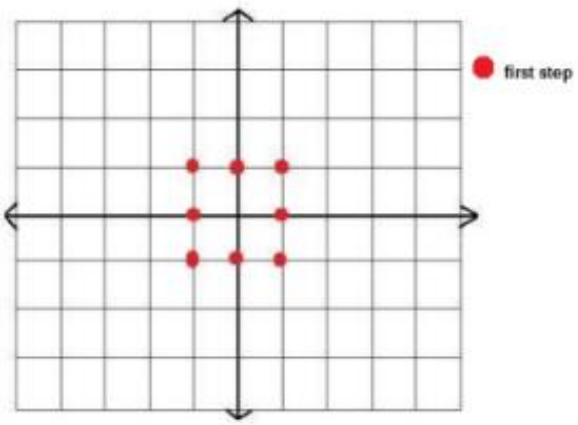
Este trabalho, proposto por Pal (2015), explora o problema que a maioria dos algoritmos da literatura possui de permanecerem presos ao mínimo local, não sendo capazes de identificar o mínimo global, ou seja, o bloco de maior similaridade ao comparado entre todos na janela de pesquisa (PANDIAN; BALA; ANITHA, 2011, tradução nossa). Neste trabalho, o algoritmo receberá a alcunha PAL.

A proposta envolve elaborar um algoritmo que possa localizar tanto o mínimo local quanto o global com um menor número de cálculos realizados. Deste modo, procura-se otimizar o processamento utilizado para encontrar o bloco ideal para geração do vetor de movimento.

O algoritmo é dividido em quatro etapas e é inspirado pelo conceito aplicado nos algoritmos TSS e DS. Na primeira etapa, são levados em consideração dois tipos de quadro: O quadro de diferença gerado pelo contraste entre o quadro anterior e o atual e o quadro em forma de 0's e 1's, chamado de quadro lógico. Em seguida, estes quadros são divididos em macro blocos.

Nesta etapa, são levantadas as somas de todos os bits do quadro de diferença e o quadro lógico. Se a diferença entre os dois quadros for zero na primeira instância, pode-se concluir que o movimento é zero também. A primeira etapa, com a primeira janela de pesquisa é ilustrada na figura 9. Pode-se observar que este início é similar ao do TSS no modo como são espalhados os primeiros pontos de procura.

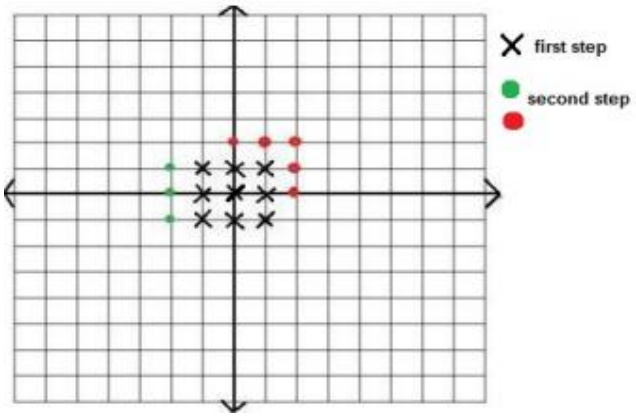
Figura 10 - Primeira etapa do algoritmo "Optimized Block Matching Using Logical Image"




Fonte: Pal (2015)

Na segunda etapa, são propostos dois modos diferentes de abordar a janela de pesquisa, em função da necessidade de precisão no momento de realizar a procura, conforme pode ser visto na figura 11.

Figura 11 - Segunda etapa do algoritmo “Optimized Block Matching Using Logical Image” estendendo a área de pesquisa

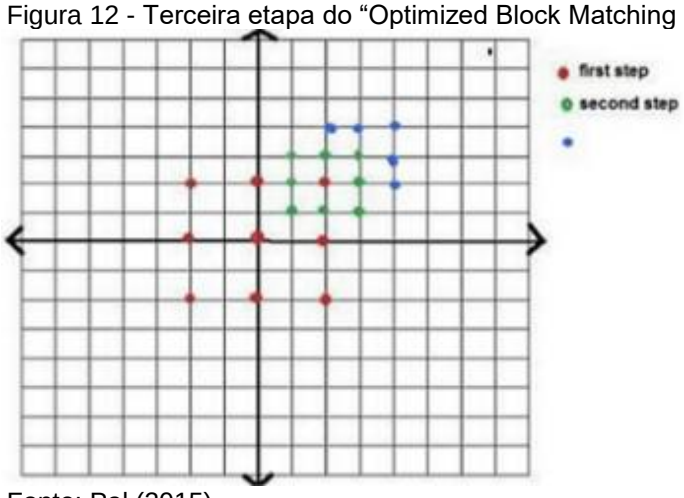



Fonte: Pal (2015)

Se a soma de todos os bits do quadro lógico é igual a zero ou 256 (o máximo, considerando que o bloco possui tamanho 16x16), o algoritmo tende a realizar uma procura mais reclusa, limitando-se a pesquisar no centro do quadro. Caso contrário, a pesquisa estende-se de modo mais amplo. Esta medida foi implementada com o intuito de localizar o mínimo global com maior exatidão.

Na terceira etapa, a mesma lógica é utilizada, alternando entre uma abordagem precisa e ampla, como feito na segunda etapa, desta vez crescendo o tamanho do passo para dois macro blocos de distância entre os pontos. A figura 12 ilustra a terceira etapa.

Figura 12 - Terceira etapa do “Optimized Block Matching Using Logical Image”

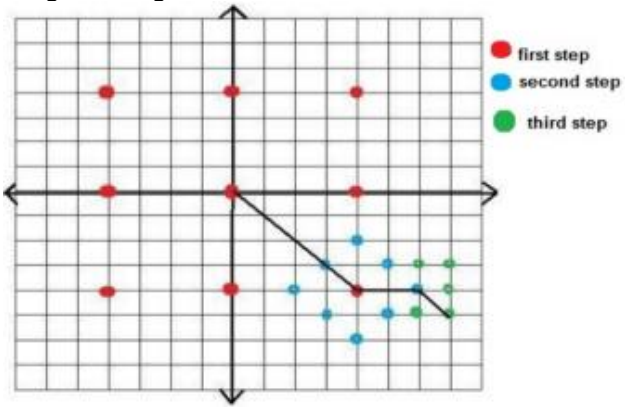



Fonte: Pal (2015)

Finalmente, na quarta etapa, se a soma dos bits do quadro de diferença e do quadro lógico excederem um limite pré-estabelecido no início da execução, o algoritmo realiza uma pesquisa ampla, conforme observado na figura 13. No fim da

execução das quatro etapas, o bloco de maior similaridade será o de melhor resultado oriundo da aplicação de todas as quatro etapas.

Figura 13 - Quarta etapa do algoritmo “Optimized Block Matching Using Logical Image”



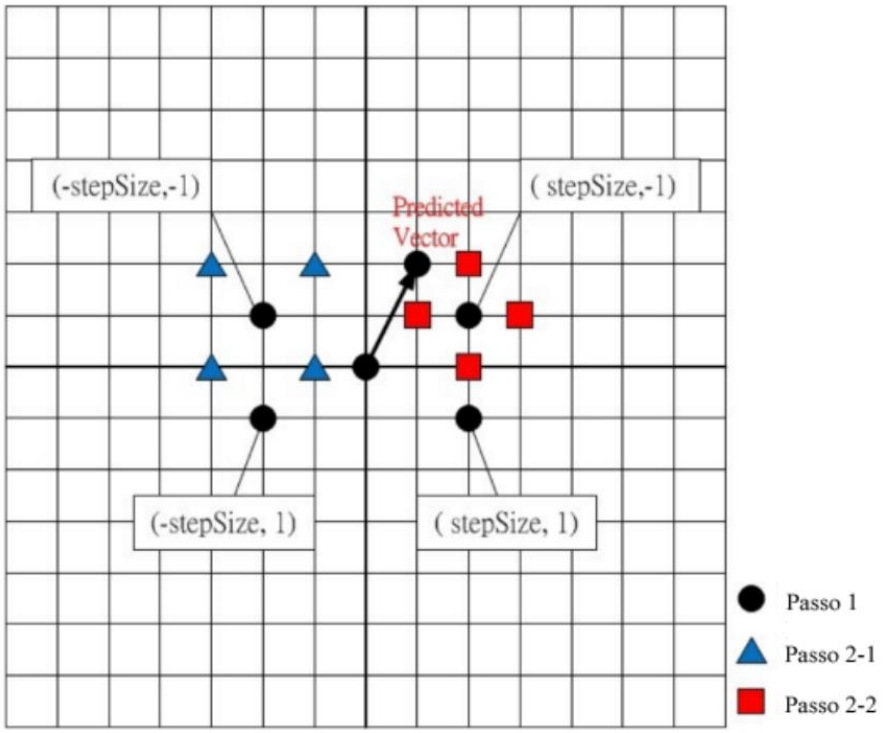

Fonte: Pal (2015)

## 4.2 A NEW BLOCK MATCHING ALGORITHM FOR MOTION ESTIMATION

Este algoritmo, concebido por Wu e Huang (2016), possui como bagagem os conceitos elaborados no algoritmo ARPS.

Wu e Huang propõem melhorias para adaptar o conceito do ARPS a vídeos em alta resolução compostos por movimentos bruscos. O mesmo considera que a maior parte da movimentação de um vídeo ocorre no plano horizontal, portanto distribui os pontos iniciais numa configuração distinta do ARPS, conforme pode ser observado na figura 14.

Figura 14 - Adaptação do algoritmo ARPS para vídeos de alta definição




Fonte: Wu e Huang (2016)

Os pontos iniciais distribuídos pela janela de pesquisa localizam-se nas coordenadas (*stepSize*, -1); (-*stepSize*, 1); (-*stepSize*, -1) e (0,0). Se o ponto de maior similaridade encontrado entre estes for o ponto em (-*stepSize*, -1); (0,0) ou (*stepSize*, 1), os pontos ilustrados em azul da figura 14 são escolhidos para dar continuidade à procura. Caso contrário, os pontos vermelhos são utilizados para refinar a pesquisa. No final das duas etapas, o bloco de maior similaridade deverá ser encontrado. Neste trabalho, o algoritmo receberá a denominação WUARPS.

## 4.3 ALGORITMO PRO

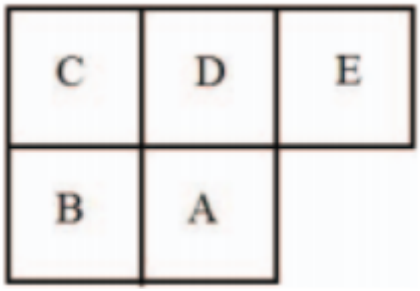
Este algoritmo, em comum com o algoritmo apresentado anteriormente, possui como sua base o algoritmo ARPS. Seu funcionamento segue basicamente as mesmas etapas, mas tem como objetivo encontrar vetores de movimento mais precisos, intrínsecos de sequências de vídeo em alta resolução, sob o custo de maior tempo de processamento e mais cálculos de bloco.

O trabalho, apresentado por Ziwei et al (2017) defende que o ARPS deixa de encontrar o melhor bloco de comparação quando empresta da iteração anterior

somente o vetor de movimento do bloco à sua esquerda para a montagem do MV predito.

Para os autores, deve-se pesquisar pelos vetores presentes em blocos vizinhos acima do mesmo, além de somente o bloco da esquerda, conforme se pode observar na figura 15, dado que A é o bloco do centro da pesquisa atual.

Figura 15 - Blocos fornecedores do MV predito no algoritmo PRO




Fonte: Ziwei et al (2017)

Seguindo este preceito, pode-se encontrar um bloco de maior similaridade que o ARPS não havia sido capaz de encontrar, por se ater ao mínimo local, em vez do global, nas etapas de refinamento (SDSP).

Destes quatro blocos candidatos, dois dos que possuírem o menor valor MAD em relação do bloco central serão escolhidos para fornecerem seu vetor de movimento, já calculado em iterações anteriores.

É feita, então, uma média ponderada destes dois vetores, a fim de encontrar um vetor que representa a média entre eles. Esta média é definida por (3), em que $w_1$ e $w_2$ somados resultem em 1:

$$MV_{predict} = w_1 * MV_1 + w_2 * MV_2 \quad (3)$$

A partir deste resultado, prosseguem-se as etapas do ARPS como o algoritmo é originalmente, utilizando como vetor predito o vetor construído nos passos relatados acima.

# 5 TRABALHOS CORRELATOS

Neste capítulo foram enumerados alguns trabalhos que compartilham da intenção de comparar algoritmos de *block-matching*, com o objetivo de evidenciar pontos positivos e negativos em cada um, além de compará-los por meio do emprego de testes padronizados, como avaliação dos resultados das funções de custo MAD, PSNR, entre outros. Por meio deste, será possível observar a motivação por trás das análises dos trabalhos citados, além de suas abordagens, material este que serviu de inspiração e norte para o presente trabalho.

## 5.1 A COMPARISON OF DIFFERENT BLOCK MATCHING ALGORITHMS FOR MOTION ESTIMATION

No artigo científico redigido por Yaakob et al (2013, tradução nossa), publicado na International Conference on Electrical Engineering and Informatics (ICEEI), o autor teve como norte avaliar o desempenho e qualidade da produção da imagem entre diferentes algoritmos de compressão de vídeo por meio da técnica de *block-matching*.

Em primeiro momento, o trabalho apresenta o conceito básico por trás da técnica de *block-matching* e como funcionam as comparações entre macro blocos e janelas de pesquisa. Em seguida, faz-se a apresentação de algumas funções de custo, como MAD e PSNR, partindo, posteriormente, para a explicação superficial de como funcionam os algoritmos a serem analisados.

Neste trabalho, o autor comparou os seguintes algoritmos: Exaustive Search (ES); New Three Step Search (NTSS); Simple and Efficient Search (SES) e Adaptive Rood Pattern Search (ARPS).

Por fim, apresentou-se, por meio de gráficos e tabelas, os resultados das comparações, utilizando como critério o resultado das funções de custo MAD e PSNR em três sequências distintas de vídeo, aplicando-as no *software* MATLAB.

## 5.2 AVALIAÇÃO ALGORÍTMICA PARA A ESTIMAÇÃO DE MOVIMENTO NA COMPRESSÃO DE VÍDEOS DIGITAIS

O trabalho apresentado por Rosa et al. (2007), submetido no periódico Hífen, da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), procurou realizar uma análise comparativa entre algoritmos de estimação de movimento com o uso do *block-matching*.

Em primeiro momento, fez-se um levantamento geral do conceito da estimação de movimento e seleção de blocos candidatos dentro de uma janela de pesquisa.

No terceiro capítulo, os autores apresentaram os algoritmos a serem submetidos à avaliação, conforme listam: Full Search (FS); One at Time Search (OTS); Diamond Search (DS); Dual Cross Search (DCS); Three Step Search (TSS) e Hexagon-based Search (HS). Dentre estes, a técnica mais recente data de 2005. Além dos algoritmos, foram implementadas duas técnicas de subamostragem, com o objetivo de avaliar os efeitos de suas aplicações.

Em seguida, foram realizados os testes em software, com todos os algoritmos implementados na linguagem C, aplicados em dez sequências de vídeo em SDTV 720x480 não comprimidas. Além dos gráficos e tabelas destacando os resultados, é realizada uma resenha sobre todos os pontos analisados no decorrer dos testes, envolvendo a análise aprofundada da lógica por trás dos algoritmos.

## 5.3 SURVEY ON BLOCK MATCHING ALGORITHMS FOR MOTION ESTIMATION

Apresentado no jornal International Conference on Communication and Signal Processing, os autores Choudhury e Saikia (2014, tradução nossa) realizaram uma avaliação dos algoritmos basais da literatura acerca da técnica de *block-matching*.

Após um breve levantamento sobre a definição de estimação e compensação de movimento, foram apresentados os cálculos das métricas avaliativas (MAD, PSNR), e em seguida os algoritmos a serem avaliados, que

foram: Full Search, New Three Step Search, Cross Search (CS), Three Step Search, Four Step Search e Adaptive Rood Pattern Search (ARPS).

O trabalho extraiu como conclusão o destaque do algoritmo ARPS como a melhor conciliação entre complexidade computacional e qualidade da imagem gerada na etapa de compensação de movimento.

## 5.4 A NEW SURVEY ON BLOCK MATCHING ALGORITHMS IN VIDEO CODING

Com o objetivo de fornecer a pesquisadores um levantamento construtivo acerca dos diversos algoritmos de *block-matching* disponíveis na literatura, este trabalho, proposto por Selvakumar e Manikandan (2014, tradução nossa) procurou, assim como os demais citados acima, realizar uma análise comparativa do desempenho das técnicas de *block-matching* mais conhecidas pelo mercado, desde os anos 1980 até 2002.

Um diferencial elaborado na realização do trabalho foi uma tabela que define a complexidade computacional para cada um dos algoritmos, assim como a quantidade de comparações efetuadas em cada um deles. Outro detalhe que vale a pena destacar neste artigo é o foco em minuciar cada etapa de cada algoritmo, servindo até mesmo como guia de implementação.

# 6 ANÁLISE COMPARATIVA ENTRE ALGORITMOS DE ESTIMAÇÃO DE MOVIMENTO POR BLOCK-MATCHING PARA COMPRESSÃO DE VÍDEO NA RESOLUÇÃO FULL HD

No presente capítulo serão abordadas as tecnologias utilizadas para a elaboração do cenário de testes, assim como a aplicação dos algoritmos e as métricas utilizadas na extração dos resultados dos testes comparativos.

## 6.1 MATLAB

O MATLAB é uma linguagem de programação multi-paradigma que encapsula em pacotes determinadas funções como plotagem de gráficos e cálculos de matrizes, formato utilizado para o armazenamento e manipulação das imagens.

Além disso, o ambiente oferece ferramentas de *benchmarking* confiáveis e amplamente utilizadas pela academia (MATHWORKS, 2011, tradução nossa).

Neste projeto foi utilizado o pacote externo de processamento de imagens (*Image Processing Toolbox*). O *software* MATLAB R2017b, utilizado na implementação e extração dos dados demanda o pagamento de uma mensalidade, mas possui licença temporária de trinta dias.

Os algoritmos foram implementados seguindo orientações fornecidas por meio de pseudocódigo presente nos artigos levantados na metodologia juntamente com um código base para execução dos algoritmos implementados, disponibilizado por Barjatya (2004) e executável pelo MATLAB.

## 6.2 FFMPEG

Para a construção do cenário utilizado para as comparações, foi utilizado o *ffmpeg*, uma biblioteca gratuita que oferece ferramentas para manipulação e análise de sequências de vídeo, operável via linha de comando.

Por meio deste, foi possível extrair os quadros em formato *bmp* sem compressão de uma amostra de vídeo em *full HD*. Estes quadros foram submetidos ao processo de estimação e compensação de movimento.

Para a extração, foi utilizado o comando *ffmpeg -i foreman.bmp $foreman%03d.bmp.* O resultado da execução do mesmo foi a criação de arquivos em *bitmap* de 24 *bits* contendo, cada um, um quadro do vídeo sem compressão.

## 6.3 PARÂMETROS DE INICIALIZAÇÃO

Ainda que os algoritmos possuam distinções essenciais em seu funcionamento, todos eles recebem os mesmos parâmetros de entrada, a fim de padronizar os testes.

Os parâmetros se encontram na tabela *3*, em que estão relacionados seus nomes com suas responsabilidades e valores inseridos.

Tabela 3 - Parâmetros de entrada da solução aplicada

| Parâmetro | Descrição | Valor inserido |
|---|---|---|
| *imageName* | Nome da sequência de vídeo | '$foreman' |
| *videoWidth* | Largura da sequência de vídeo | 1920 |
| *videoHeight* | Altura da sequência de vídeo | 1080 |
| *mbSize* | Tamanho do macro bloco | 16 |
| *p* | Largura e altura da janela de pesquisa, em *pixels* | 7 |

Fonte: Do autor

Estes parâmetros serão alimentados em cada um dos algoritmos, que executarão sua etapa de estimação de movimento, gerando, consequentemente, os vetores de movimento e demais saídas que serão relatadas posteriormente.

## 6.4 EXECUÇÃO DA SOLUÇÃO

Assim que a aplicação recebe os parâmetros de entrada, define-se qual algoritmo de estimação de movimento será utilizado. Esta informação é definida em tempo de compilação.

Ao executar a solução, é realizada uma iteração por todos os quadros da sequência de vídeo escolhida, processando, em todos eles, a etapa de estimação e compensação de movimento.

### 6.4.1 Estimação de movimento

A etapa de estimação de movimento é encarregada de comparar o conteúdo do quadro vigente com o quadro duas posições à frente. Nesta implementação, todos os quadros vigentes foram *I*-frames, o que não ocorre na prática. Num processo de codificação real, muitas vezes o quadro atual será um *P-frame* ou *B-frame*. Todavia, foi desenvolvido desta maneira para fins de isolar os resultados de maneira mais controlável e analítica.

Neste momento, o vídeo é dividido em macro blocos de tamanho definido pelo parâmetro *mbSize,* e em cada um destes blocos é projetada uma janela de pesquisa de tamanho definido pelo parâmetro *p,* que se estende em *p pixels* para as quatro direções.

Nesta janela de pesquisa é que será realizada a busca pelo bloco de maior similaridade, utilizando como critério para definição de tal o resultado dos cálculos de desvio médio padrão (MAD) que tiver o menor valor. Para o algoritmo mais básico da implementação, o *Exhaustive Search*, percorre-se toda a janela, comparando todos os 225 blocos possíveis da janela com o atual. Tal prática consome um tempo inviável de codificação. Os algoritmos deste trabalho tentam simular os resultados ótimos do ES sem realizar tantas comparações.

Ao encontrar o bloco ideal, o algoritmo encerra a busca, guarda o vetor de movimento correspondente às coordenadas do bloco ideal encontrado numa matriz chamada *vectors* e o valor MAD entre este bloco e o atual.

Por fim, a solução inicia uma nova busca no bloco adjacente à direita do atual, em que o mesmo processo descrito acima será realizado novamente.

Ao se deparar com o limite à direita do quadro, pula-se para a “linha” de baixo na posição *X* zero (bloco adjacente abaixo do primeiro bloco). Quando o programa chega ao limite extremo à direita e abaixo, termina-se a execução da estimação de movimento deste quadro e avança-se para o próximo.

### 6.4.2 Compensação de movimento

Antes de passar para o próximo quadro, o programa executa a etapa de compensação de movimento, em que o quadro de referência é manipulado conforme os vetores de movimento retirados da etapa de estimação de movimento.

Este processo ocorre bloco por bloco. A imagem é montada retirando blocos do quadro de referência deslocados de acordo com os vetores e posicionando-os no local da iteração atual. A figura 16 ilustra o processo de compensação de movimento da solução aplicada neste trabalho.

Figura 16 - Processo de reconstrução do quadro por meio da compensação de movimento


Fonte: Do autor

O programa encerra as atividades quando alcança o último quadro do vídeo.

## 6.5 VARIÁVEIS DE SAÍDA

A execução da solução culmina na saída das mídias relacionadas a seguir, utilizadas, neste trabalho, para a análise e comparação entre os algoritmos aplicados.

Estes resultados são determinantes para contrastar as soluções em termos de qualidade da imagem reconstruída, tempo de execução e complexidade computacional do algoritmo.

a) razão sinal-ruído de pico - *Peak signal to noise ratio* (PSNR): indica, em dB, a qualidade da imagem reconstruída, comparando-a com a imagem original. Para extrair o PSNR, deve-se comparar o conteúdo do quadro atual com o quadro gerado pela compensação de movimento;
b) tempo de execução: indica, em segundos, o tempo utilizado para a geração dos vetores e a reconstrução do quadro, medido por intermédio da função *tic; toc;* do MATLAB. Esta função oferece os melhores resultados em termos de precisão na medição do tempo, utilizando métodos robustos que atenuam os atrasos provocados por agentes externos (MATHWORKS, 2011, tradução nossa);
c) média dos diferenças médias absolutas - *Mean absolute difference* (MAD): a média entre todos os desvios médios absolutos retornados, que indicam a similaridade entre o bloco atual e o bloco considerado como ideal. Como pode ser conferido no capítulo da metodologia, quanto menor esta variável, mais fiel deverá ser a reconstrução da imagem, mensurada pelo PSNR;
d) número de cálculos - Para cada iteração, os algoritmos possuem um determinado número de blocos candidatos em que se realiza a procura pelo bloco mais similar. Para cada bloco em que se mede o MAD, incrementa-se o contador em um. Ao final da execução, somam-se os valores. Para efeito de contextualização, o *Exhaustive Search* sempre resultará em 225 cálculos, e o TSS sempre 25.
e) quadro de diferença: Este quadro é gerado comparando os *pixels* da imagem reconstruída contra os *pixels* correspondentes do quadro original. Isso destaca todas as imperfeições pontuais da compensação de movimento, que pode servir para comparar a eficiência qualitativa dos algoritmos. Para este fim foi utilizado o comando *imshow* do MATLAB.

## 6.6 SEQUÊNCIAS DE TESTE

Foram utilizadas três sequências de vídeo na realização deste trabalho. Todas possuem resolução full HD (1920x1080) e são representadas em escala de cinza. Uma destas sequências, chamada *flow,* é uma amostra de vídeo que ilustra um dançarino sendo alvejado por partículas de água, sequência que possui um alto teor de detalhamento na imagem, além de um momento na metade do vídeo em que existe alto grau de movimentação. A mesma possui 167 quadros e apresenta baixa movimentação entre eles.

Outra sequência, chamada *forest*, mostra um veículo atravessando um percurso rodeado por árvores. O desafio de compressão deste vídeo reside no fato de que o painel do carro não se move, mas as árvores ao redor possuem alto grau de movimentação. Esta sequência, assim como a anterior, possui 167 quadros.

A terceira sequência, chamada *foreman*, que é basicamente a "reencenação" de uma sequência também amplamente utilizada. Isso se deve ao fato do vídeo possuir diversos tipos de movimento, que põem à prova todos os algoritmos em vários quesitos diferentes. Estes quesitos incluem movimentos sutis, movimentos bruscos, deslocamento geral de todos os pixels da tela e blocos com zero movimentação. Esta sequência possui 180 quadros.

A figura 17 apresenta um quadro de exemplo de cada uma das sequências. À esquerda, a *flow*, e à direita, a *forest*. Abaixo de todas encontra-se a *foreman* Todas as amostras possuem uma taxa de 30 quadros por segundo e não possuem compressão realizada de antemão, ou seja, são sequências de vídeo descomprimidas, ou *raw*.

Figura 17 - Sequências de teste utilizadas

Fonte: Do autor (2018)

# 7 RESULTADOS OBTIDOS

Neste capítulo serão apresentados os resultados levantados mediante a execução dos algoritmos. Os mesmos serão relacionados em formato de tabelas, comparando-se dados como PSNR (em dB), tempo (em segundos) e quantidade de cálculos.

Além dos algoritmos implementados neste projeto, serão impressos nos gráficos e tabelas os algoritmos ES; TSS, precursor do PAL; ARPS, precursor dos algoritmos WUARPS e PRO. Isso será realizado com o objetivo de nivelar o progresso dos algoritmos em relação a seus precursores. Estes algoritmos foram implementados na solução de exemplo fornecida por Barjatya (2004).

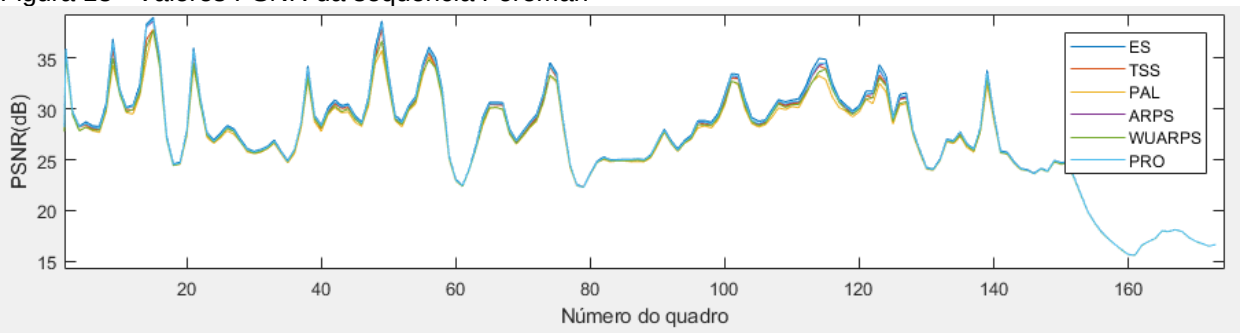
Além disso, será fornecido, como meio ilustrativo, os quadros de diferença entre o quadro gerado pela compensação de movimento e o original.

## 7.1 QUALIDADE DO QUADRO REPRODUZIDO

Por meio da etapa de compensação de movimento, podemos reconstruir o quadro desejado a partir do *I-frame* e seus vetores de movimento. Inerentemente, sua qualidade não será totalmente igual à original, por conta de diferenças pontuais entre blocos e também em função da potencial ineficiência do algoritmo em produzir vetores de movimento precisos.

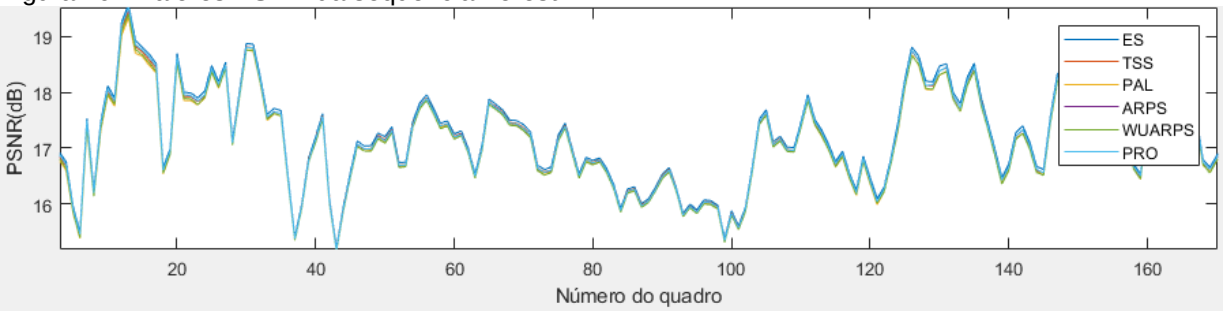
Essas diferenças podem ser mensuradas por meio do valor PSNR. As figuras 18, 19 e 20 ilustram graficamente os valores PSNR produzidos pela execução da solução, quadro a quadro. O eixo vertical define o valor PSNR (medido em dB) e o eixo horizontal define o número do quadro de que foi retirado o valor.

Figura 18 - Valores PSNR da sequência Foreman



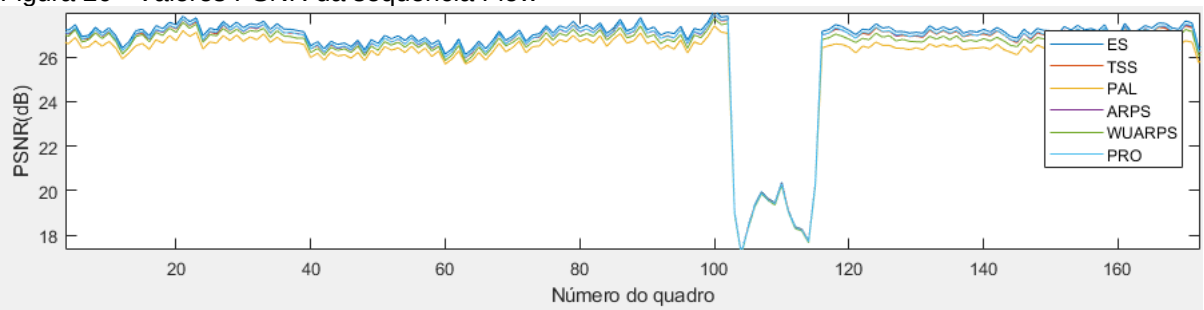

Fonte: Do autor

Figura 19 - Valores PSNR da sequência Forest.




Fonte: Do autor

Figura 20 - Valores PSNR da sequência Flow




Fonte: Do autor

Tomando como base os resultados exibidos nas figuras, podemos concluir que, como o esperado, o algoritmo *Exhaustive Search* rendeu os melhores resultados em termos de qualidade da imagem reconstruída, nos três contextos aplicados.

Dos três algoritmos apresentados neste trabalho, pode-se atribuir destaque ao algoritmo PRO, que no decorrer dos três cenários apresentou consistente qualidade que supera, na maioria das vezes, os do ARPS, algoritmo que

lhe serviu de inspiração. Este algoritmo foi o que apresentou o grau de qualidade mais similar ao ES.

Na sequência *Foreman*, em full HD, podemos observar que o algoritmo PAL possui resultados PSNR abaixo do seu algoritmo pai, o TSS, em 1.2% no segmento A, e 0.01% no segmento B. Isso indica menor eficiência na geração dos vetores de movimento em ocasiões de baixa movimentação e eficiência equiparada em alta movimentação.

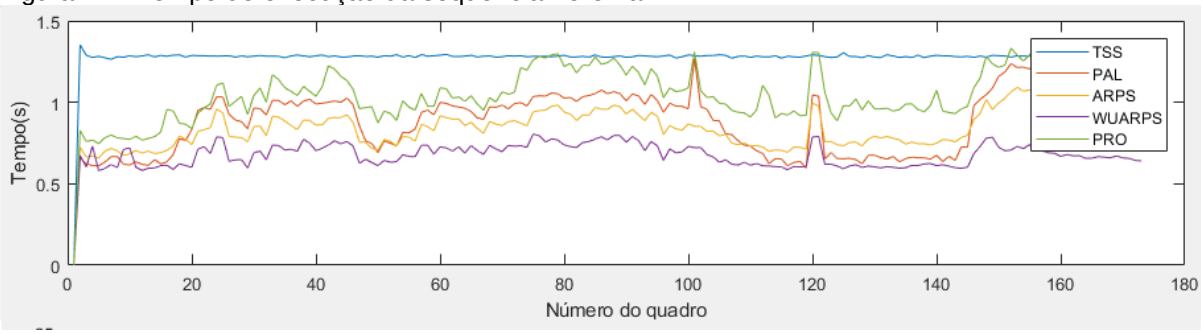
Em seguida, o algoritmo WUARPS exibe resultados abaixo de seu inspirador, o ARPS, salvo em momentos de alta movimentação (sequência *forest*), cenário no qual o WUARPS se assemelha à qualidade do ARPS.

## 7.2 TEMPO DE EXECUÇÃO

Os algoritmos apresentados neste trabalho, mesmo que apresentando qualidade ligeiramente inferior de reconstrução da imagem, possuem um ponto positivo que é o aproveitamento de banda e economia de recursos, virtude que se torna valiosa, dependendo das condições de tráfego da rede.

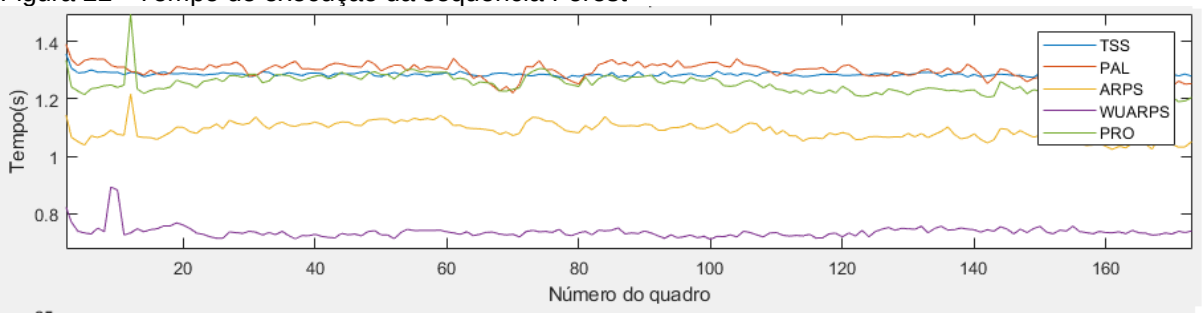
Nas figuras 21, 22 e 23 está ilustrado graficamente, quadro a quadro, o tempo utilizado para codificar e decodificar os quadros para cada algoritmo comparado. O eixo vertical marca o tempo em segundos.

Figura 21 - Tempo de execução da sequência Foreman



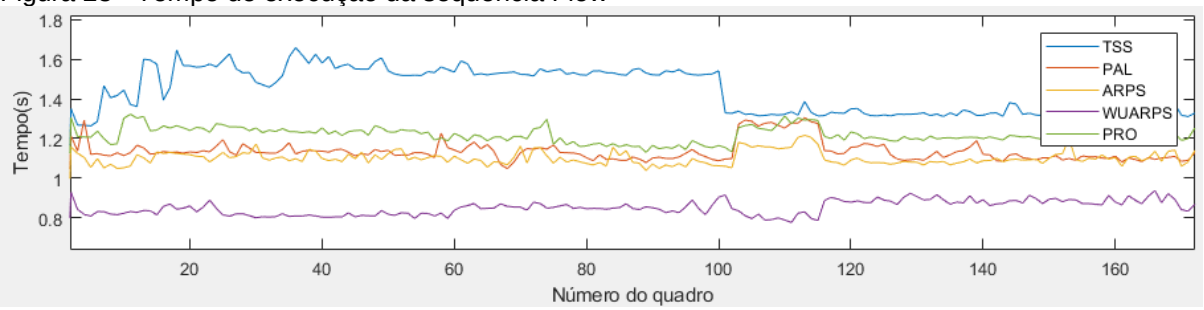

Fonte: Do autor

Figura 22 - Tempo de execução da sequência Forest




Fonte: Do autor

Figura 23 - Tempo de execução da sequência Flow




Fonte: Do autor

Os tempos de execução do ES, por possuírem uma disparidade muito grande entre os demais, foram deixados de fora, mas possui, em média, um tempo de execução sete vezes maior que os outros algoritmos.

Pode-se detectar, por meio destes resultados, que o algoritmo WUARPS possui um tempo de codificação menor, em relação a todos os outros, em 23.4% na sequência de baixa e média movimentação (*foreman e flow*), e de 37.8% na sequência de alta movimentação (forest). Ou seja, possui melhor desempenho quando lida com sequências de alto grau de movimentação.

Observa-se, também, que a sequência PRO possui um tempo de execução superior ao do ARPS em aproximadamente 14%, que seria devido a seus pontos extra de pesquisa, no processo de montagem do vetor predito (sexto vetor).

Podemos, por fim, destacar que o algoritmo PAL possui um tempo relativamente errático, devido a seus métodos de encerramento da execução variados, que poupam o algoritmo de realizar pesquisas que julga desnecessárias.

Mesmo assim, o PAL possui uma melhora de 41.8% e 5.6% nas sequências de baixa e alta movimentação, respectivamente, em comparação com o seu precursor, o TSS. Isso indica que o algoritmo tem a capacidade de produzir

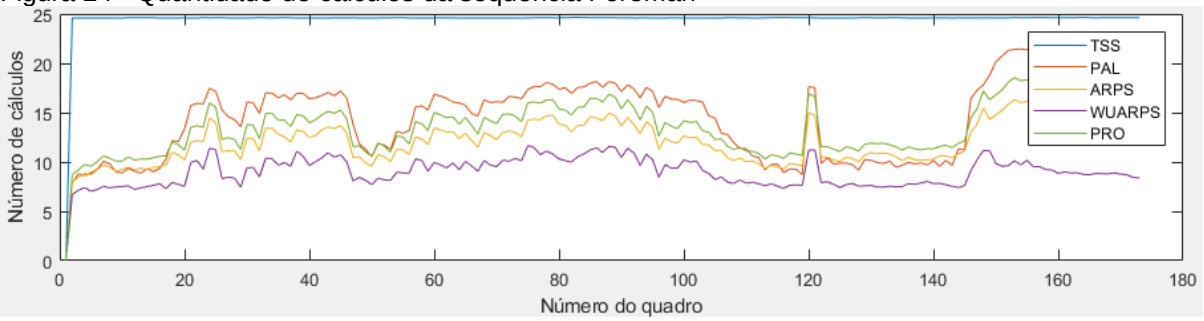
vetores próximos do TSS, com a vantagem da elevada economia de recursos. Ainda assim, o PAL leva mais tempo para codificar, em comparação com o WUARPS e o PRO.

## 7.3 QUANTIDADE DE CÁLCULOS POR QUADRO

Aliado aos resultados de qualidade do quadro reconstruído e tempo para a codificação e decodificação, podemos também requerer a quantidade de cálculos efetuados, que significa a quantidade de comparações entre blocos no processo de codificação. Neste caso, o processo de decodificação não influencia os resultados em nenhuma maneira. Deste modo, pode-se isolar de maneira mais eficaz o dispêndio de recursos de forma crua, para cada algoritmo.

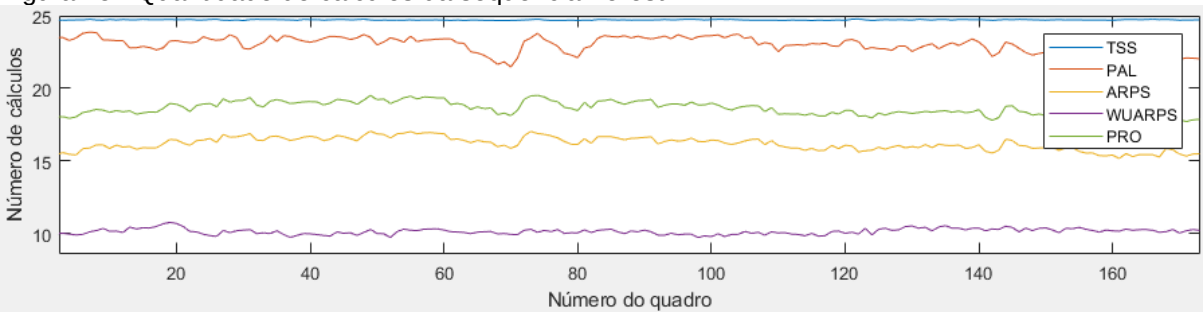
As figuras 24, 25 e 26 ilustram as quantidades de cálculos para cada algoritmo, quadro a quadro. O eixo vertical marca a quantidade de cálculos.

Figura 24 - Quantidade de cálculos da sequência Foreman



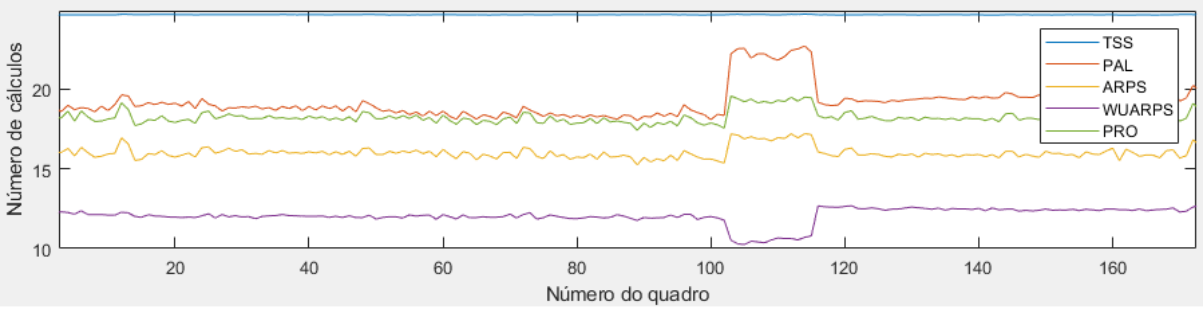

Fonte: Do autor

Figura 25 - Quantidade de cálculos da sequência Forest




Fonte: Do autor

Figura 26 - Quantidade de cálculos da sequência Flow




Fonte: Do autor

Novamente, por motivos de grande disparidade gráfica entre o ES e os demais, o mesmo foi deixado de fora dos resultados gráficos. No entanto, sabe-se que seu número de cálculos é 225 de forma constante.

Pode-se observar que o número de cálculos do algoritmo WUARPS é o menor de todos, o que justifica seu tempo reduzido de execução, significando que o algoritmo encontra o mínimo local em menos iterações.

Observa-se que o algoritmo PRO possui um número de cálculos sempre superior ao do ARPS, devido ao maior número de pontos de pesquisa para a montagem do vetor de predição, emprestado de blocos vizinhos. Essa disparidade aumenta à medida que também aumenta a movimentação.

Podemos observar também que o algoritmo PAL possui um número de cálculos por vezes inferior ao do ARPS e PRO no segmento de baixa movimentação. No entanto, este cenário muda no momento em que os movimentos se tornam mais bruscos, ou seja, se eleva a movimentação. Isso reforça a teoria de que o algoritmo PAL é mais bem empregado em sequências de vídeo em que se faz uso de informações fortemente redundantes, sem muitos movimentos bruscos.

Os resultados são semelhantes no contexto da baixa resolução.

## 7.1 RELATÓRIO GERAL

As tabelas 4, 5 e 6 apresentam uma média dos resultados obtidos com a execução da solução. Nela contém as variáveis de saída relacionadas no capítulo anterior. As tabelas correspondem, respectivamente, às sequências *Foreman, Forest* e *Flow*.

Tabela 4 - Resultados gerais da sequência Foreman

| Algoritmo | PSNR (dB) | Tempo (s) | MAD médio | Nº de cálculos |
|---|---|---|---|---|
| ES | 27,504 | 1560,89 | 26,127 | 225,00 |
| TSS | 27,237 | 220,71 | 26,147 | 25,00 |
| ARPS | 27,333 | 145,98 | 26,162 | 12,18 |
| PAL | 26,954 | 155,36 | 26,228 | 14,81 |
| WUARPS | 27,096 | 116,77 | 26,205 | 9,04 |
| PRO | 27,371 | 182,90 | 26,150 | 13,63 |

Tabela 5 - Resultados gerais da sequência Forest

| Algoritmo | PSNR (dB) | Tempo (s) | MAD médio | Nº de cálculos |
|---|---|---|---|---|
| ES | 17,283 | 1555,44 | 18,554 | 225,00 |
| TSS | 17,228 | 221,18 | 18,753 | 25,00 |
| ARPS | 17,228 | 187,28 | 18,719 | 16,18 |
| PAL | 17,183 | 223,27 | 19,136 | 23,01 |
| WUARPS | 17,180 | 126,73 | 19,041 | 10,13 |
| PRO | 17,231 | 214,91 | 18,714 | 18,59 |

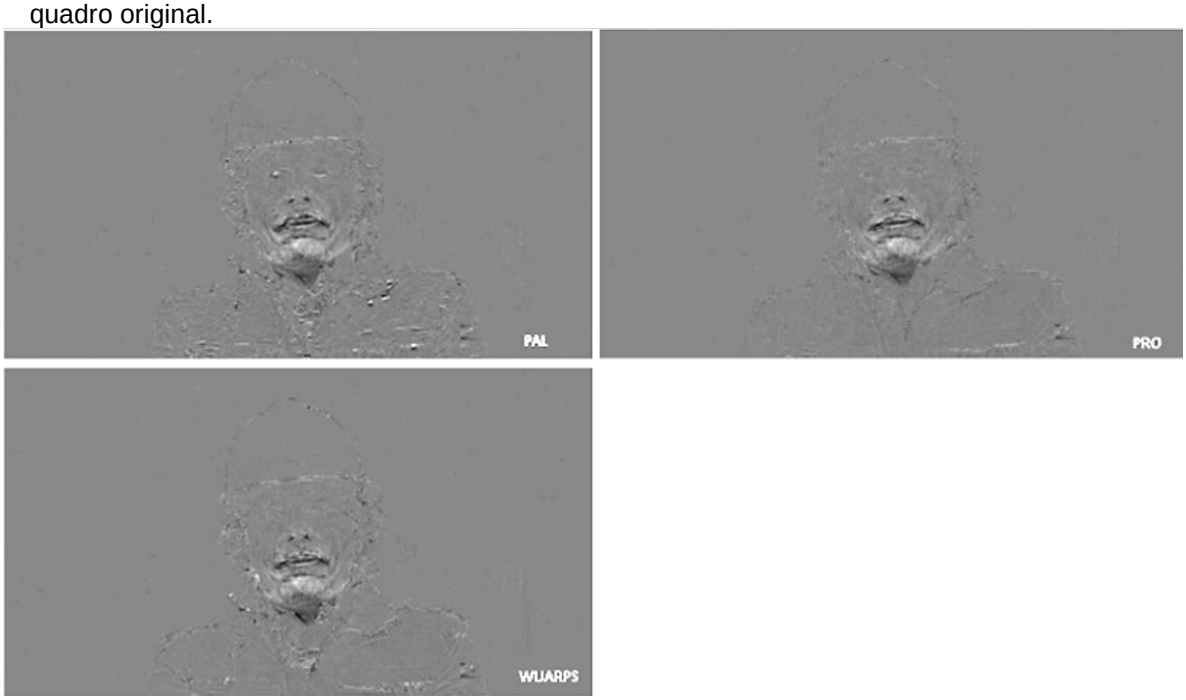
Tabela 6 - Resultados gerais da sequência Flow

| Algoritmo | PSNR (dB) | Tempo (s) | MAD médio | Nº de cálculos |
|---|---|---|---|---|
| ES | 26,520 | 1597,72 | 6,367 | 225,00 |
| TSS | 26,365 | 247,91 | 6,488 | 25,00 |
| ARPS | 26,378 | 189,54 | 6,492 | 16,01 |
| PAL | 25,916 | 195,38 | 7,091 | 19,21 |
| WUARPS | 26,206 | 146,28 | 6,690 | 12,05 |
| PRO | 26,387 | 209,45 | 6,488 | 18,24 |

## 7.5 QUADROS DE DIFERENÇA

Nesta etapa serão ilustradas as diferenças entre os quadros reconstruídos pela compensação de movimento e os quadros originais, aos quais os quadros reconstruídos melhor deveriam se assemelhar. As diferenças podem ser melhor vistas na sequência *Foreman*, do qual foi retirado um dos quadros para a elaboração deste quadro de diferença.

Figura 27 - Quadros de diferença entre os algoritmos PAL, PRO e WUARPS em relação ao quadro original.




Fonte: Do autor

Utilizando como base as imagens demonstradas na figura 27, podemos observar que quanto mais detalhes forem aparentes além do tom constante cinza, maior será a diferença entre o quadro reconstruído e o quadro original.

Destaca-se o algoritmo PAL, que possui, entre todos, o pior valor PSNR. Isso acarreta em mais pontos aparentes na figura 27. Podemos observar que até mesmo a silhueta dos olhos possui detalhes aparentes, ausentes nos outros algoritmos comparados. Isso pode dificultar o processo de correção da predição de erro.

# 8 CONCLUSÃO

O presente trabalho apresentou e comparou três métodos de compressão de vídeo por meio da técnica de *block-matching* propostos nos últimos três anos, destacando suas qualidades e dificuldades em meio a diversos cenários.

No início do projeto, foram levantados os aspectos gerais do funcionamento da compressão de vídeos digitais e eliminação da redundância entre quadros por meio da técnica de *block-matching, composto pela* divisão do quadro em macro blocos e a sucessiva geração de vetores de movimento.

O levantamento de recentes algoritmos foi motivado pela necessidade da busca de novos métodos que fossem capazes de detectar movimentos minuciosos da resolução *full HD* sem comprometer a velocidade da compressão, tampouco a complexidade computacional.

Os resultados obtidos com a implementação revelam que, entre os três algoritmos, o que melhor se sobressai em termos de qualidade da imagem produzida foi o algoritmo PRO, proposto em 2017. Este algoritmo, mesmo possuindo melhor qualidade de compressão, não elevou sua complexidade computacional a níveis inaceitáveis, mantendo seu tempo de execução e quantidade de cálculos sutilmente acima aos outros algoritmos comparados.

Outra conclusão tirada é que o algoritmo PAL possui os menores tempos de execução, em ambas as resoluções, dado que a quantidade de movimentos seja limitada. Ou seja, é um algoritmo qualificado para ser utilizado quando se leva em consideração banda limitada de rede e necessidade de compressão em *full HD* sem uso de movimentos bruscos, como vídeo aulas ou palestras.

Por fim, o algoritmo WUARPS fornece um tempo de execução que chega a até 40% de economia de tempo e quantidade de cálculos em sequências de alto grau de movimentação, sem comprometer a qualidade da compressão. Em quadros de baixa movimentação, sua qualidade deixa a desejar, mas possui um tempo de execução equiparável aos outros algoritmos.

Em trabalhos futuros, pode-se abordar outros algoritmos, pertinentes às novas tecnologias do padrão HEVC, em que se faz uso de tamanhos de bloco variados, em vez de uma grade de blocos de tamanhos fixos. Deste modo, movimentos extremamente pequenos podem ser isolados em maior detalhe e

trabalhados minuciosamente, enquanto que seções estáticas do quadro recebem um tamanho maior de bloco, a fim de evitar cálculos de comparação desnecessários.

# REFERÊNCIAS

AHMAD, Ishfaq; HE, Yong; LIOU, Ming. **Video compression with parallel processing.** Parallel Computing. Arlington, p. 1039-1079. dez. 2001. Disponível em: <http://dl.acm.org/citation.cfm?id=611434>. Acesso em: 24 ago. 2017.

AL-NAJDAWI, Nijad et al. A Frequency Based Hierarchical Fast Search Block Matching Algorithm for Fast Video Communication. **International Journal Of Advanced Computer Science And Applications**. Ames, p. 1-9. maio 2016. Disponível em: <https://www.researchgate.net/publication/301542602>. Acesso em: 23 out. 2017.

BARJATYA, Aroh. **Block matching algorithms for motion estimation.** 2004. 6 f. TCC (Graduação) - Curso de Ciência da Computação, Utah State University, Logan, 2004. Disponível em: <https://goo.gl/kz72A1>. Acesso em: 13 maio 2017.

CARNE, E. Bryan. **Connections for the digital age**: Multimedia communications for mobile, nomadic and fixed devices. Nova Jersey: Wiley, 2011. 280 p.

CHOUDHURY, Hussain Ahmed; SAIKIA, Monjul. Survey on Block Matching Algorithms for Motion Estimation. **International Conference On Communication And Signal Processing**. Melmaruvathur, p. 36-40. abr. 2014. Disponível em: <ieeexplore.ieee.org/document/6949794/>. Acesso em: 26 out. 2017.

CONKLIN, Gregory J. et al. Video Coding for Streaming Media Delivery on the Internet. **IEEE Transactions On Circuits And Systems For Video Technology.** Seattle, p. 269-280. mar. 2001.

CUEVAS, Erik et al. Block matching algorithm based on Differential Evolution for motion estimation. **Engineering Applications Of Artificial Intelligence**. Guadalajara, p. 488-498. jan. 2013. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0952197612002047>. Acesso em: 25 set. 2017.

DUFAUX, Frederic; KONRAD, Janusz. **Efficient, robust, and fast global motion estimation for video coding.** 3. ed. Charlottesville: Ieee, 2000. Disponível em: <http://ieeexplore.ieee.org/abstract/document/826785/>. Acesso em: 09 maio 2017.

FLORÊNCIO, Ayrton Galindo Bernardino. **Sistema de estabilização de vídeo baseado em acelerômetro, filtragem robusta e algoritmo de busca de três etapas.** 2015. 106 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade de São Paulo, São Carlos, 2015. Disponível em: <http://www.teses.usp.br/teses/disponiveis/18/18153/tde-19102015-085456/pt-br.php>. Acesso em: 24 out. 2017.

GOLEMATI, Spyretta et al. Carotid artery wall motion estimated from b-mode ultrasound using region tracking and block matching. **Ultrasound In Medicine & Biology**. Amsterdam, p. 387-399. mar. 2003. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0301562902007603>. Acesso em: 21 nov. 2017.

HAMID, na et al. A New Orthogonal: Diamond Search Algorithm for Motion Estimation. **IEEE International Conference On Computer, Communication, And Control Technology**. Langkawi, p. 467-471. set. 2014. Disponível em: <http://ieeexplore.ieee.org/document/6914228/>. Acesso em: 04 set. 2017.

KAMBLE, Shailesh; THAKUR, Nileshsingh; BAJAJ, Preeti. Modified Three-Step Search Block Matching Motion Estimation and Weighted Finite Automata based Fractal Video Compression. **International Journal Of Interactive Multimedia And Artificial Intelligence**. Madrid, p. 27-39. jun. 2017. Disponível em: <http://www.ijimai.org/journal/node/1512>. Acesso em: 02 out. 2017.

KINTSCHNER, Ricardo et al. **Implementação Prática de Algoritmo de estimação de Movimento H.264 Paralelo em um Processador Cell.** Florianópolis: Sociedade Brasileira de Computação, 2011. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/webmedia/2011/025.pdf>. Acesso em: 12 maio 2017.

KOGA, Takanori et al. Motion-compensated interframe coding for video conferencing. **National Telesystems Conference Proceedings**. New Orleans, p. G5.3.1–G5.3.5. dez. 1981.

KONDOZ, Ahmet. **Visual Media Coding and Transmission**. Chichester: Wiley, 2009. 556 p.

KUHN, Peter. **Algorithms, complexity analysis and VLSI architectures for MPEG-4 motion estimation.** Dordrecht: Kluwer, 1999. 420 p.

KULKARNI, Satish M; BORMANE, Dattatraya; NALBALWAR, Sanjay. Coding of Video Sequences Using Three Step Search Algorithm. **Procedia Computer Science**. Kakinada, p. 42-49. jan. 2015. Disponível em: <https://goo.gl/Q8mWRT>. Acesso em: 28 ago. 2017.

LAI, Yeong-Kang, CHOU, Chih-Chung, CHUNG, Yu-Chieh. A simple and cost effective video encoder with memory-reducing CAVLC. **IEEE Transactions On Circuits And Systems For Video Technology.** Montreal, p. 432-435. maio 2001.

LAKAMSANI, Prasad; ZENG, Bing; LIOU, Ming. An Enhanced Three Step Search Motion Estimation Method and its VLSl Architecture. **IEEE International Symposium On Circuits And Systems**. Clear Water Bay, p. 754-757. maio 1996.

Disponível em: <http://ieeexplore.ieee.org/abstract/document/541835/>. Acesso em: 20 nov. 2017.

LI, Renxiang; ZENG, Bing; LIOU, Ming. A New Three-Step Search Algorithm for Block Motion Estimation. **IEEE Transactions On Circuits And Systems For Video Technology**. Chicago, p. 438-442. ago. 1994. Disponível em: <ieeexplore.ieee.org/iel4/76/7587/00313138.pdf>. Acesso em: 25 set. 2017.

LU, Jianhua; LIOU, Ming. A Simple and Efficient Search Algorithm For Block-Matching Motion Estimation. **IEEE Transactions On Circuits And Systems For Video Technology**. Chicago, p. 429-433. abr. 1997. Disponível em: <ieeexplore.ieee.org/document/564122>. Acesso em: 27 set. 2017.

ZIWEI, Lu et al. An improved block matching motion estimation algorithm based on adaptive rood pattern search. **IEEE Control And Decision Conference**. Chongqing, p. 5199-5203. maio 2017. Disponível em: <https://ieeexplore.ieee.org/document/7979419/>. Acesso em: 20 jan. 2018.

LUO, Lijun et al. A New Prediction Search Algorithm for Block Motion Estimation in Video Coding. **IEEE Transactions On Consumer Electronics**. Nanjing, p. 56-61. fev. 1997. Disponível em: <ieeexplore.ieee.org/document/580385/>. Acesso em: 25 set. 2017.

LE GALL, Didier. The MPEG video compression algorithm. **Signal Processing: Image Communication 4.** Amsterdam, p. 129-140. abr. 1992.

MATHWORKS. **Image Processing and Computer Vision**. Disponível em: <https://www.mathworks.com/products/matlab.html>. Acesso em: 21 nov. 2017

MATHWORKS. **Improvements to tic and toc Functions for Measuring Absolute Elapsed Time Performance in MATLAB.** Disponível em: <https://www.mathworks.com/company/newsletters/articles/improvements-to-tic-and-toc-functions-for-measuring-absolute-elapsed-time-performance-in-matlab.html>. Acesso em: 22 maio 2018

METKAR, Shilpa, TALBAR, Sanjay, **Motion Estimation Techniques for Digital Video Coding**, Nova Delhi: Springer, 2013. 64 p.

NIE, Yao; MA, Kai-Kuang. Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation, **IEEE Transactions on Image Processing**, Maryland, p. 1442-1448, dez, 2002.
Disponível em: <ieeexplore.ieee.org/document/1176932>. Acesso em: 02 out. 2017.

PAL, Manisha. An Optimized Block Matching Algorithm for Motion Estimation using Logical Image. **International Conference On Computing, Communication And**

**Automation**. Delhi, p. 1138-1142. maio 2015. Disponível em: <ieeexplore.ieee.org/iel7/7126877/7148334/07148546.pdf>. Acesso em: 07 nov. 2017.

PANDIAN, S Immanuel Alex; BALA, G Josemin; ANITHA, J. Enhanced modified orthogonal search for motion estimation. **Recent Advances In Intelligent Computational Systems (raics), 2011 Ieee.** Trivandrum, p. 907-910. nov. 2011. Disponível em: <http://ieeexplore.ieee.org/document/6069440/>. Acesso em: 23 out. 2017.

PHILIP, Jobin T. et al. A Comparative Study of Block Matching and Optical Flow Motion Estimation Algorithms. **International Conference on Magnetics, Machines & Drives**. Florida. jul. 2014. Disponível em: <ieeexplore.ieee.orgiel7689522069081540690820 4.pdf>. Acesso em: 01 set. 2017.

RICHARDSON, Iain. **H.264 and MPEG-4 Video Compression:** Video Coding for Next-generation Multimedia. Chichester: John Wiley & Sons, 2003.

ROSA, Leandro et al. **Avaliação Algorítmica para a estimação de Movimento na Compressão de Vídeos Digitais.** Uruguaiana: Pucrs, 2007. Disponível em: <http://revistaseletronicas.pucrs.br/ojs/index.php/hifen/article/view/3868/2942>. Acesso em: 10 maio 2017.

SELVAKUMAR, Rk; MANIKANDAN, Lc. A New Survey on Block Matching Algorithms in Video Coding. **International Journal Of Engineering Research**. Marthandam, p. 121-125. fev. 2014. Disponível em: <www.ijer.in/ijer/publication/v3s2/IJER_2014_218.pdf>. Acesso em: 26 out. 2017.

SELVAKUMAR, Rk; MANIKANDAN, Lc. A Study on Block Matching Algorithms for Motion Estimation in Video Coding. **International Journal Of Scientific & Engineering Research.** Delhi, p. 25-30. jul. 2014. Disponível em: <https://www.ijser.org/paper/A-Study-on-Block-Matching-Algorithms-for-Motion-Estimation-in-Video-Coding.html>. Acesso em: 25 set. 2017.

SHAHID, Muhammad. **On Computational Complexity ofMotion Estimation Algorithms inMPEG-4 Encoder.** 2010. 57 f. Dissertação (Mestrado) - Curso de Mestrado em Engenharia Elétrica, Blekinge Institute Of Technology, Karlskrona, 2010.

SIMERIA, Luc. Accelerating MATLAB using MEX-files. 2008. Disponível em: <https:www.embedded.com.print4016874>. Acesso em: 21 nov. 2017.

TAKAYA, Kunio. Detection of Moving Object in Video Scene: MPEG like Motion Vector vs. Optical Flow. **The First International Workshop On Video Processing**

**For Security**. Quebec, p. 1-8. jun. 2006. Disponível em: <http://www.computer-vision.org/4security/pdf/saskatchewan.pdf>. Acesso em: 25 set. 2017.

TURAGA, Deepak; ALKANHAL, Mohamed. **Search Algorithms for Block-Matching in Motion Estimation**. 1998. Disponível em: <https://www.ece.cmu.edu/~ee899/project/deepak_mid.htm>. Acesso em: 07 nov. 2017.

VELDHUIZEN, Todd. **Measures of image quality.** 1998. Disponível em: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html>. Acesso em: 06 nov. 2017.

WOOTTON, Cliff. **A Practical Guide to Video and Audio Compression:** From Sprockets and Rasters to Macro Blocks. Burlington: Elsevier, 2005. 787 p.

WU, Dapeng et al. Streaming Video over the Internet: Approaches and Directions. **IEEE Transactions On Circuits And Systems For Video Technology.** Pittsburgh, p. 282-300. mar. 2001.

WU, Chia-ming; HUANG, Jen-yi. A New Block Matching Algorithm for Motion Estimation. **Applied Mechanics And Materials.** Zürich, p. 179-183. out. 2016. Disponível em: <ieeexplore.ieee.org/document/743234/>. Acesso em: 20 nov. 2017.

YAAKOB, Razali et al. A Comparison of Different Block Matching Algorithms for Motion Estimation. **International Conference On Electrical Engineering And Informatics**. Bangkok, p. 199-205. abr. 2013. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2212017313003356>. Acesso em: 24 out. 2017.

ZHU, Shan; MA, Kai-Kuang. A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. **IEEE Transactions on Image Processing**. Maryland, p. 287-290, fev, 2000.
Disponível em: <ieeexplore.ieee.org/document/821744>. Acesso em: 29 set. 2017.

## APÊNDICE A – ARTIGO CIENTÍFICO

# ANÁLISE COMPARATIVA ENTRE ALGORITMOS DE ESTIMAÇÃO DE MOVIMENTO POR BLOCK-MATCHING PARA COMPRESSÃO DE VÍDEO EM RESOLUÇÃO FULL HD

**Flavio H. Cimolin**

Curso de Ciência da Computação

Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brazil

fhcimolin@gmail.com

***Abstract.*** *Due to the recent growth in video resolutions, there has been a growth in bandwidth and storage use as well. In order to decrease the general size of these vídeos, it is crucial that new vídeo coding algorithms be developed, able to eliminate redundancies and generate the highest quality results without the spiking use of resources. This paper presents three algorithms using the block-matching motion estimation technique, introduced in the past three years which aim to further improve methods already known by the literature, enhancing the quality in the video's reassembling and reducing coding time consumption.*

***Resumo.*** *Em virtude do recente aumento das resoluções de vídeos digitais, existe paralelamente um crescimento no uso de armazenamento e banda para o envio desta carga de dados. Para que se possa reduzir o peso desses vídeos, é crucial que haja também a elaboração de algoritmos de codificação de vídeo cada vez melhores, a fim de eliminar redundâncias, gerar resultados de maior qualidade sem o dispêndio elevado de recursos. Este trabalho apresenta três algoritmos de estimação de movimento por block-matching dos últimos três anos que visam aprimorar técnicas já conhecidas pela literatura, melhorando a qualidade de reconstrução do quadro e reduzindo o tempo de compressão.*

## 1. Introdução

A transmissão em tempo real de vídeos de alta qualidade vem sendo um dos maiores focos da academia nas últimas duas décadas, em vista de seu elevado potencial de geração de renda. Aplicações voltadas ao streaming online tornaram-se, portanto, cada vez mais voltadas para a otimização de desempenho e qualidade.

Alguns exemplos destas aplicações poderiam compreender, por exemplo, conferências ao vivo num ambiente corporativo, em que a qualidade do vídeo, se comprometida, poderia causar uma impressão negativa no cliente; Serviços de televisão digital sob demanda (IPTV), em que o cliente pode assistir a basicamente qualquer filme ou série em alta definição de onde estiver, sem necessidade de realizar o download de um arquivo excessivamente pesado; Seminários educativos, como webinars e palestras.

O envio desta carga de dados com alta fidelidade não é tão simples, no entanto. Vídeos digitais em alta definição possuem alta demanda de largura de banda, costumam apresentar delay na apresentação, podem sofrer interrupções e severa perda de qualidade em

regiões que dispõem de má qualidade de conexão. Em virtude destes motivos, o desenvolvimento de mecanismos de compressão de vídeo se fazem cada vez mais necessário para amenizar estas adversidades (WU et al., 2001, tradução nossa; CONKLIN et al., 2001, tradução nossa).

A compressão de vídeo tem como base o princípio de que os quadros dentro de um vídeo apresentam muitas informações redundantes, cuja ausência não é perceptível pelo olho humano. O objetivo da compressão é eliminar estas redundâncias, conciliando desempenho e qualidade de vídeo (ROSA et al, 2007). Uma técnica de compressão eficaz e amplamente empregada nos CODECs de video é o *block-matching*. Esta técnica é utilizada em padrões reconhecidos mundialmente, como a série H.264 e MPEG, por ser simples, rápida e efetiva (ZHU; MA, 1997, tradução nossa).

O objetivo do block-matching consiste em dividir um quadro em blocos e compará-los com blocos da imagem anterior, com o objetivo de gerar vetores de movimentação, responsáveis por mover apenas algumas partes da imagem original para formar a imagem sucessora, sem uso de informações da mesma. (AL-NAJDAWI et al., 2016, tradução nossa).

Os algoritmos disponíveis na literatura datam de uma época em que não existia a definição *full HD* (1920x1080), que requer que muitos pontos de pesquisa sejam avaliados para a realização da estimação de movimento (RAO, MURALIDHAR, RAO, 2014, tradução nossa), que pode comprometer tanto a qualidade da imagem quanto o uso de processamento, por considerar o mínimo local como sendo a solução ótima (TURAGA; ALKANHAL, 1998, tradução nossa)

Em vista deste argumento, a proposta do presente trabalho de conclusão de curso possui como finalidade realizar a análise comparativa dos resultados das aplicações de três algoritmos desenvolvidos nos últimos três anos com foco na resolução *full HD* para compressão de vídeo por block-matching, com base em métricas avaliativas validadas pela academia.

## 2. Fundamentos da compressão de video

Vídeos costumam demandar uma grande quantidade de dados para transferência. Kintschner et al. (2011) contextualiza este problema, evidenciando que uma sequência de vídeo em full HD (1920x1080 pixels), utilizando um formato 4:2:0 a 30 fps, ocupa aproximadamente 746 Mbits por segundo de banda. O transporte desta carga, utilizando um serviço de provedor comum de 1MB/s é irrealizável. Para tanto, a etapa de compressão (codificação) de um vídeo antes de seu envio é fundamental.

Neste processo, um arquivo de vídeo que originalmente demandaria um largo espaço de banda é reduzido a uma sequência de dados codificados de tamanho razoável, que em seguida é reconstruído no nó de destino, seja ele transmitido via web ou até mesmo arquivamento local.

Dentro de um intervalo de tempo, um vídeo pode conter muitos quadros praticamente iguais uns aos outros. Um meio utilizado para reduzir o espaço ocupado por estes quadros é identificar a redundância temporal e, por meio de cálculos de estimação de movimento, render vetores de movimentação (KUHN, 1999, tradução nossa). Estes servem para mover apenas algumas partes pontuais da imagem de referência necessárias para formar a imagem seguinte, e possuem um peso correspondente a apenas uma fração do quadro original. Estes quadros são chamados de *P-frames* ou *B-frames* (WOOTTON, 2005, tradução nossa).

Estes tipos de quadros fazem proveito de informações do quadro que o antecede. É criado pela técnica de estimação de movimento e reduz drasticamente a redundância temporal. *P-frames*, por conterem somente informações de movimento relativo (não contém nenhum dado de imagem), podem pesar até três vezes menos do que seu quadro de referência (CARNE, 2011, tradução nossa).

A estimação de movimento pode ser descrita como um processo cuja entrada é um quadro de referência e que a saída são vetores responsáveis por ilustrar o movimento de informações de um quadro ao próximo (CARNE, 2011, tradução nossa).

## 3. A técnica de *block-matching*

O conceito principal por trás da técnica de *block-matching* consiste em dividir o quadro atual numa matriz de blocos chamados macro blocos. Esta divisão parte do princípio de que todos os pixels dentro deste bloco sofrerão o mesmo grau de deslocamento, e são, portanto, passíveis de serem tratados como uma só entidade (CHOUDHURY, SAIKIA, 2014, tradução nossa).

Estes macro blocos são comparados com blocos vizinhos adjacentes do quadro anterior, processo que deverá ter como produto um vetor de movimento, responsável por definir de onde serão "emprestados" os pixels pertencentes ao bloco no momento da reconstrução da imagem (PHILIP, 2014, tradução nossa).

A comparação entre os blocos ocorre no que é chamada uma janela de pesquisa, em que os mesmos, divididos em 16x16 pixels recebem uma variável *p* (em *pixels*), que designa o alcance de comparação entre o bloco atual e os adjacentes, a fim de encontrar o bloco de maior similaridade (KINTSCHNER et al, 2011).

Todos os blocos da imagem recebem uma janela de pesquisa e são comparados com blocos adjacentes candidatos. O bloco candidato de maior similaridade, ou seja, com menor nível de distorção em relação ao bloco atual é utilizado para a geração do vetor de movimento (KULKARNI; BORMANE; NALBALWAR, 2015, tradução nossa), conforme pode ser visto na figura 1.

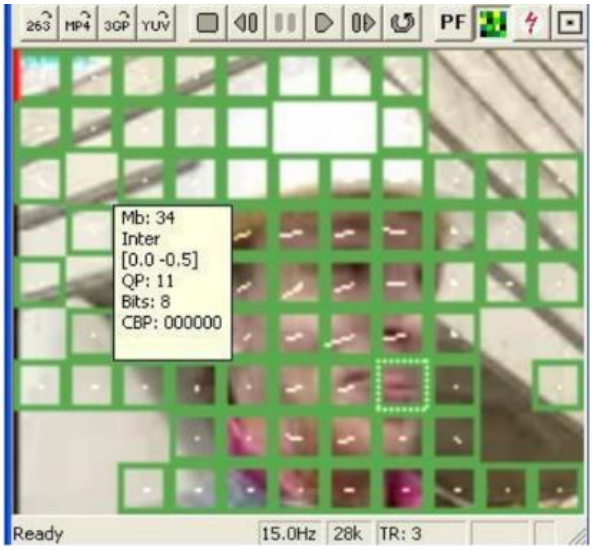



**Figura 1 - Vetores de movimentação atribuídos aos macro blocos**

Os *I-frames* e os vetores de movimento são enviados ao destinatário. Ao chegar, a imagem original é reconstruída com base nos vetores de movimento que deslocam informações do *I-frame*, gerando as partes do quadro que foram descartadas.

## 4. Algoritmos comparados

No decorrer da década de 1990, muitos algoritmos de *block-matching* foram elaborados com a finalidade de reduzir a carga de processamento na detecção do bloco ideal para geração dos vetores de movimento.

A extensa variedade de algoritmos de block-matching disponíveis no mercado torna a decisão de qual deles utilizar uma questão de equilíbrio entre complexidade, dificuldade de implementação, desempenho, escalabilidade e qualidade da imagem produzida (AL-NAJDAWI et al., 2016, tradução nossa).

Os algoritmos disponíveis na literatura datam de uma época em que não existiam displays em *High Definition* (do inglês alta definição, HD). Mesmo versáteis e ágeis para aplicações de baixa disponibilidade de banda, estes algoritmos não são adequados para os movimentos complexos e minuciosos da definição *full HD*, que requerem que a janela de pesquisa seja extensa demais para a realização da estimação de movimento (RAO, MURALIDHAR, RAO, 2014, tradução nossa).

Muitos algoritmos realizam buscas muito amplas, ou muito isoladas, o que favorece as chances dos algoritmos localizarem apenas o bloco de resultado mínimo local como grau de distorção. Encontrar somente o mínimo local significa tomar como escolha final um bloco que somente parece que é, entre todos, o mais similar, sendo que em outro ponto da janela existe um bloco mais similar ao comparado, chamado de mínimo global. Os algoritmos comparados neste trabalho propõem novas maneiras de se encontrar este mínimo global, com foco na resolução full HD.

O primeiro algoritmo a ser comparado neste trabalho foi proposto por Pal (2015). Chamado de PAL, possui como conceito avaliar a necessidade de abrangência do alcance da procura dentro da janela de pesquisa. São levados em consideração dois tipos de quadro: O quadro de diferença, gerado pelo contraste entre o quadro anterior e o atual e o quadro em forma de 0’s e 1’s, chamado de quadro lógico. Em seguida, estes quadros são divididos em macro blocos.

O algoritmo levanta as somas de todos os bits do quadro de diferença e o quadro lógico. Esta soma determina a quantidade de movimento que foi detectado em todo o quadro. Dependendo desta soma, o algoritmo pode decidir entre fazer uma pesquisa de longo alcance ou de curto alcance.

O segundo algoritmo, elaborado por Wu e Huang (2016) é chamado WUARPS e possui como conceito um fundamento do algoritmo ARPS (Adptive Rood Pattern Search, de Nie e Ma (2002)), que diz que a tendência é que o vetor de movimento do bloco atual é muito similar ao vetor do bloco à sua esquerda. Portanto, um bloco conveniente para realizar a comparação, além dos 4 blocos iniciais, é o bloco correspondente ao do vetor de movimento do bloco à esquerda do bloco atual.

No entanto, Wu e Huang defendem que existe uma configuração destes quatro pontos iniciais que melhor pode representar movimentos bruscos, que possui uma disposição horizontal, em vez de vertical, como ilustado na figura 2:

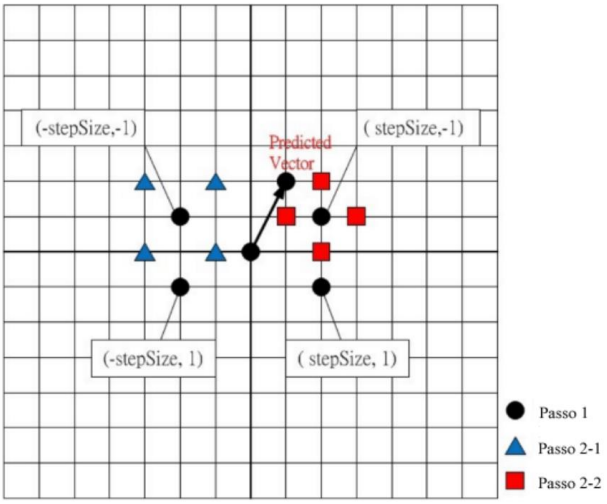



**Figura 2: Adaptação do algoritmo ARPS para vídeos de alta definição**

O terceiro algoritmo, chamado de PRO, proposto por Ziwei et al (2017), também possui como bagagem os conceitos do algoritmo ARPS, e defende que deve-se pesquisar pelos vetores presentes em blocos vizinhos acima do mesmo, além de somente o bloco da esquerda, conforme se pode observar na figura 15, dado que A é o bloco do centro da pesquisa atual.

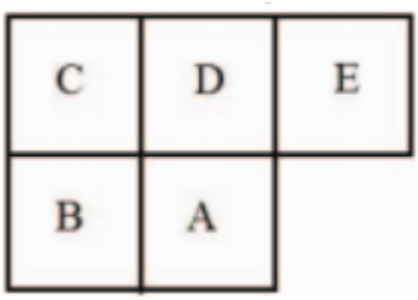



**Figura 3 - Blocos fornecedores do MV predito no algoritmo PRO**

Destes quatro blocos candidatos, dois dos que possuírem o menor valor MAD em comparação do bloco central serão escolhidos para fornecerem seu vetor de movimento, já calculado em iterações anteriores.

## 5. Ferramentas utilizadas

Os algoritmos apresentados neste trabalho foram implementados seguindo orientações fornecidas por meio de pseudocódigo presente nos artigos levantados na metodologia juntamente com um código base para execução dos algoritmos implementados, disponibilizado por Barjatya (2004) e executável pelo MATLAB.

Ainda que os algoritmos possuam distinções essenciais em seu funcionamento, todos eles recebem os mesmos parâmetros de entrada, a fim de padronizar os testes. Os parâmetros se encontram na tabela 1, em que estão relacionados seus nomes com suas responsabilidades e valores inseridos.

| Parâmetro | Descrição | Valor inserido |
|---|---|---|
| *imageName* | Nome da sequência de vídeo | '$foreman' |
| *videoWidth* | Largura da sequência de vídeo | 1920 |
| *videoHeight* | Altura da sequência de vídeo | 1080 |
| *mbSize* | Tamanho do macro bloco | 16 |
| *P* | Largura e altura da janela de pesquisa, em *pixels* | 7 |

**Tabela 1 - Parâmetros de entrada da solução aplicada**

A execução da solução culmina na saída das mídias relacionadas a seguir, utilizadas, neste trabalho, para a análise e comparação entre os algoritmos aplicados.

Estes resultados são determinantes para contrastar as soluções em termos de qualidade da imagem reconstruída, tempo de execução e complexidade computacional do algoritmo.

a) razão sinal-ruído de pico - Peak signal to noise ratio (PSNR): indica, em dB, a qualidade da imagem reconstruída, comparando-a com a imagem original. Para extrair o PSNR, deve-se comparar o conteúdo do quadro atual com o quadro gerado pela compensação de movimento;

b) tempo de execução: indica, em segundos, o tempo utilizado para a geração dos vetores e a reconstrução do quadro, medido por intermédio da função tic; toc; do MATLAB. Esta função oferece os melhores resultados em termos de precisão na medição do tempo, utilizando métodos robustos que atenuam os atrasos provocados por agentes externos (MATHWORKS, 2011, tradução nossa);

c) média dos diferenças médias absolutas - Mean absolute difference (MAD): a média entre todos os desvios médios absolutos retornados, que indicam a similaridade entre o bloco atual e o bloco considerado como ideal. Como pode ser conferido no capítulo da metodologia, quanto menor esta variável, mais fiel deverá ser a reconstrução da imagem, mensurada pelo PSNR;

d) número de cálculos - Para cada iteração, os algoritmos possuem um determinado número de blocos candidatos em que se realiza a procura pelo bloco mais similar. Para cada bloco em que se mede o MAD, incrementa-se o contador em um. Ao final da execução, somam-se os valores. Para efeito de contextualização, o Exhaustive Search sempre resultará em 225 cálculos, e o TSS sempre 25.

Foram utilizadas três sequências de vídeo na realização deste trabalho. Todas possuem resolução full HD (1920x1080) e são pré-codificadas em escala de cinza. Uma destas sequências, chamada *flow*, é uma amostra de vídeo que ilustra um dançarino sendo alvejado por partículas de água, sequência que possui um alto teor de detalhamento na imagem, além de um momento na metade do vídeo em que existe alto grau de movimentação.

Outra sequência, chamada *forest*, mostra um veículo percorrendo um percurso rodeado por árvores. O desafio de compressão deste vídeo reside no fato de que o painel do carro não se move, mas as árvores ao redor possuem alto grau de movimentação. A terceira sequência, chamada *foreman* é basicamente a "reencenação" de uma sequência também amplamente utilizada, devido ao fato do vídeo possuir diversos tipos de movimento, que põem à prova todos os algoritmos em vários quesitos diferentes. Estes quesitos incluem movimentos sutis, movimentos bruscos, deslocamento geral de todos os pixels da tela e blocos com zero movimentação.

## 6. Resultados

Neste capítulo serão apresentados os resultados levantados mediante a execução dos algoritmos. Os mesmos serão relacionados em formato de tabelas, comparando-se dados como PSNR (em dB), tempo (em segundos) e quantidade de cálculos.

Além dos algoritmos implementados neste projeto, serão impressos nos gráficos e tabelas os algoritmos ES; TSS, precursor do PAL; ARPS, precursor dos algoritmos WUARPS e PRO. Isso será realizado com o objetivo de nivelar o progresso dos algoritmos em relação a seus precursores. Estes algoritmos foram implementados na solução de exemplo fornecida por Barjatya (2004).

| Algoritmo | PSNR (dB) | Tempo (s) | MAD médio | Nº de Cálculos |
|---|---|---|---|---|
| ES | 27,504 | 1560,89 | 26,127 | 225,00 |
| TSS | 27,237 | 220,71 | 26,147 | 25,00 |
| ARPS | 27,333 | 145,98 | 26,162 | 12,18 |
| PAL | 26,954 | 155,36 | 26,228 | 14,81 |
| WUARPS | 27,096 | 116,77 | 26,205 | 9,04 |
| PRO | 27,371 | 182,90 | 26,150 | 13,63 |

**Tabela 2 - Resultados gerais da sequência Foreman**

| Algoritmo | PSNR (dB) | Tempo (s) | MAD médio | Nº de Cálculos |
|---|---|---|---|---|
| ES | 17,283 | 1555,44 | 18,554 | 225,00 |
| TSS | 17,228 | 221,18 | 18,753 | 25,00 |
| ARPS | 17,228 | 187,28 | 18,719 | 16,18 |
| PAL | 17,183 | 223,27 | 19,136 | 23,01 |
| WUARPS | 17,180 | 126,73 | 19,041 | 10,13 |
| PRO | 17,231 | 214,91 | 18,714 | 18,59 |

**Tabela 3 - Resultados gerais da sequência Forest**

| Algoritmo | PSNR (dB) | Tempo (s) | MAD médio | Nº de Cálculos |
|---|---|---|---|---|
| ES | 26,520 | 1597,72 | 6,367 | 225,00 |
| TSS | 26,365 | 247,91 | 6,488 | 25,00 |
| ARPS | 26,378 | 189,54 | 6,492 | 16,01 |
| PAL | 25,916 | 195,38 | 7,091 | 19,21 |
| WUARPS | 26,206 | 146,28 | 6,690 | 12,05 |
| PRO | 26,387 | 209,45 | 6,488 | 18,24 |

**Tabela 4 - Resultados gerais da sequência Flow**

Dos três algoritmos apresentados neste trabalho, pode-se atribuir destaque ao algoritmo PRO, que no decorrer dos três cenários apresentou consistente qualidade que supera, na maioria das vezes, os do ARPS, algoritmo que lhe serviu de inspiração. Este algoritmo foi o que apresentou o grau de qualidade mais similar ao *Exhaustive Search*, que procura em todos os blocos da janela de pesquisa.

Podemos também observar que o algoritmo PAL exibe menor eficiência na geração dos vetores de movimento em ocasiões de baixa movimentação e eficiência equiparada em alta movimentação.

Em seguida, o algoritmo WUARPS exibe resultados abaixo de seu inspirador, o ARPS, salvo em momentos de alta movimentação (sequência Foreman), cenário no qual o WUARPS se assemelha à qualidade do ARPS.

Pode-se detector que o algoritmo WUARPS possui um tempo de codificação menor, em relação a todos os outros, em 23.4% em sequências de baixa movimentação (flow e foreman), e de 37.8% no segmento de alta movimentação (forest). Ou seja, possui melhor desempenho quando lida com sequências de alto grau de movimentação.

Observa-se, também, que a sequência PRO possui um tempo de execução superior ao do ARPS em aproximadamente 14%, que seria devido a seus pontos extra de pesquisa, no processo de montagem do vetor predito.

Podemos, por fim, destacar que o algoritmo PAL possui um tempo relativamente errático, devido a seus métodos de encerramento da execução variados, que poupam o algoritmo de realizar pesquisas que julga desnecessárias.

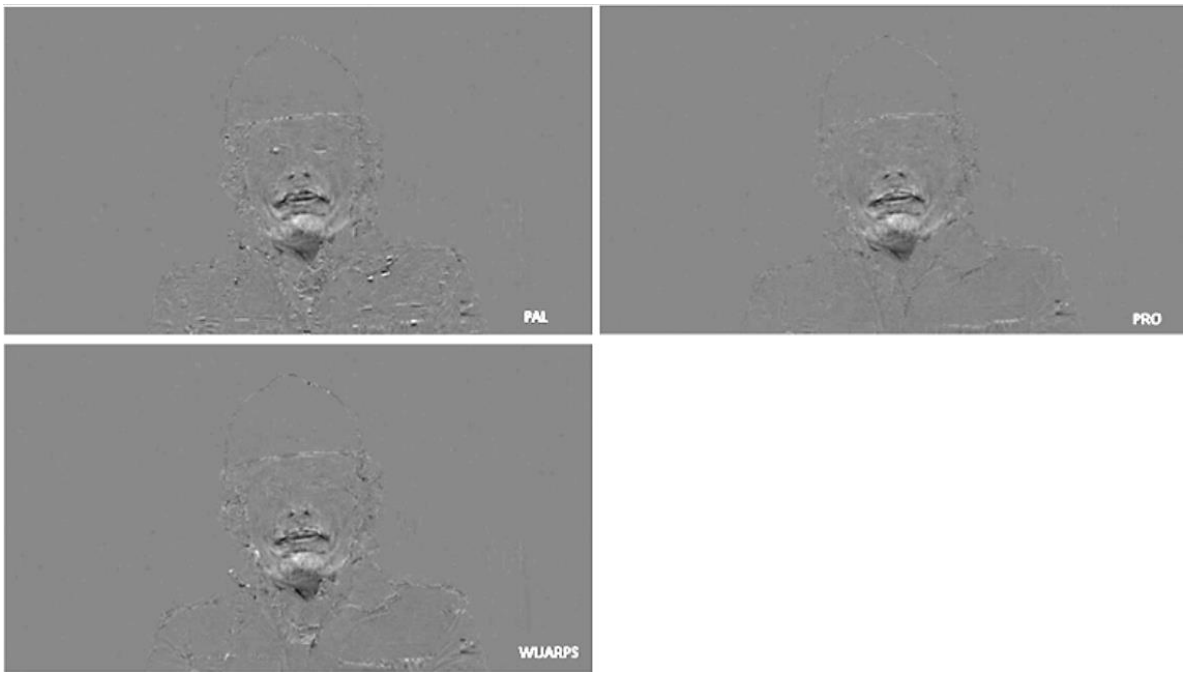



**Figura 4 – Quadro de diferença entre após aplicação dos algoritmos**

Utilizando como base as imagens demonstradas na figura 27, podemos observar que quanto mais detalhes forem aparentes além do tom constante cinza, maior será a diferença entre o quadro reconstruído e o quadro original.

Destaca-se o algoritmo PAL, que possui, entre todos, o pior valor PSNR. Isso acarreta em mais pontos aparentes na figura 27. Podemos observar que até mesmo a silhueta dos olhos possui detalhes aparentes, ausentes nos outros algoritmos comparados. Isso pode dificultar o processo de correção da predição de erro.

## 7. Conclusão

O presente trabalho apresentou e comparou três métodos de compressão de vídeo por meio da técnica de block-matching propostos nos últimos três anos, destacando suas qualidades e dificuldades em meio a diversos cenários.

No início do projeto, foram levantados os aspectos gerais do funcionamento da compressão de vídeos digitais e eliminação da redundância entre quadros por meio da técnica de block-matching, composto pela divisão do quadro em macro blocos e a sucessiva geração de vetores de movimento.

O levantamento de recentes algoritmos foi motivado pela necessidade da busca de novos métodos que fossem capazes de detectar movimentos minuciosos da resolução full HD sem comprometer a velocidade da compressão, tampouco a complexidade computacional.

Os resultados obtidos com a implementação revelam que, entre os três algoritmos, o que melhor se sobressai em termos de qualidade da imagem produzida foi o algoritmo PRO, proposto em 2017. Este algoritmo, mesmo possuindo melhor qualidade de compressão, não elevou sua complexidade computacional a níveis inaceitáveis, mantendo seu tempo de execução e quantidade de cálculos sutilmente acima aos outros algoritmos comparados.

Outra conclusão tirada é que o algoritmo PAL possui os menores tempos de execução, em ambas as resoluções, dado que a quantidade de movimentos seja limitada. Ou seja, é um algoritmo qualificado para ser utilizado quando se leva em consideração banda limitada de rede e necessidade de compressão em full HD sem uso de movimentos bruscos, como vídeo aulas ou palestras.

Por fim, o algoritmo WUARPS fornece um tempo de execução que chega a até 40% de economia de tempo e número de cálculos em sequências de alto grau de movimentação, sem comprometer a qualidade da compressão. Em quadros de baixa entropia, sua qualidade deixa a desejar, mas possui um tempo de execução equiparável aos outros algoritmos.

Em trabalhos futuros, pode-se abordar outros algoritmos, pertinentes às novas tecnologias do padrão HEVC, em que se faz uso de tamanhos de bloco variados, em vez de uma grade de blocos de tamanhos fixos. Deste modo, movimentos extremamente pequenos podem ser isolados em maior detalhe e trabalhados minuciosamente, enquanto que seções estáticas do quadro recebem um tamanho maior de bloco, a fim de evitar cálculos de comparação desnecessários.

## REFERÊNCIAS

AL-NAJDAWI, Nijad et al. A Frequency Based Hierarchical Fast Search Block Matching Algorithm for Fast Video Communication. International Journal Of Advanced Computer Science And Applications. Ames, p. 1-9. maio 2016.

CARNE, E. Bryan. Connections for the digital age: Multimedia communications for mobile, nomadic and fixed devices. Nova Jersey: Wiley, 2011. 280 p.

CONKLIN, Gregory J. et al. Video Coding for Streaming Media Delivery on the Internet. IEEE Transactions On Circuits And Systems For Video Technology. Seattle, p. 269-280. mar. 2001.

KINTSCHNER, Ricardo et al. Implementação Prática de Algoritmo de ESTIMAÇÃO de Movimento H.264 Paralelo em um Processador Cell. Florianópolis: Sociedade Brasileira de Computação, 2011.

TURAGA, Deepak; ALKANHAL, Mohamed. Search Algorithms for Block-Matching in Motion Estimation. 1998.

WOOTTON, Cliff. A Practical Guide to Video and Audio Compression: From Sprockets and Rasters to Macro Blocks. Burlington: Elsevier, 2005. 787 p.

WU, Dapeng et al. Streaming Video over the Internet: Approaches and Directions. IEEE Transactions On Circuits And Systems For Video Technology. Pittsburgh, p. 282-300. mar. 2001.

ZHU, Shan; MA, Kai-Kuang. A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. IEEE Transactions on Image Processing. Maryland, p. 287-290, fev, 2000.