UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO SUPERIOR DE ENGENHARIA DE COMPUTAÇÃO

BRUNNO ALBERTO WISTUBA BRAGA
CAIO NOGARA ANDREATTA

SASQV2: AMBIENTE DE ADIÇÃO CONTROLADA DE ARTEFATOS E AVALIAÇÃO DE VÍDEOS DIGITAIS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2012

BRUNNO ALBERTO WISTUBA BRAGA
CAIO NOGARA ANDREATTA

SASQV2: AMBIENTE DE ADIÇÃO CONTROLADA DE ARTEFATOS E AVALIAÇÃO DE VÍDEOS DIGITAIS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso II, do Curso Superior de Engenharia de Computação do Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de Engenheiro.

Orientadora: Prof[a]. Dr[a]. Keiko Veronica Ono Fonseca

CURITIBA

2012

AGRADECIMENTOS

Primeiramente, agradecemos aos alunos Wyllian e Emmerson do LCD (Laboratório de Comunicação de Dados) da UTFPR pela paciência em responder às nossas dúvidas, pelos materiais de auxílio e pelas opiniões sobre o que foi desenvolvido, mesmo sem nos conhecer e sem ter a obrigação de fazê-lo.

À Marina e à Hannah pelo carinho e dedicação na correção do texto e, principalmente, pela compreensão durante este processo que exigiu grande parte do nosso tempo.

Ao professor Dario por ter persistido nesta greve, permitindo que nós utilizássemos o laboratório para continuar desenvolvendo o projeto; pelas dicas de apresentação e pelas conversas matinais motivacionais.

À professora e orientadora Keiko pela oportunidade de dar continuidade a este projeto, pelo incansável esforço em estar sempre presente orientando, auxiliando e nos conduzindo pelo melhor caminho.

RESUMO

ANDREATTA, Caio N.; BRAGA, Brunno A. W.. SASQV2: AMBIENTE DE ADIÇÃO CONTROLADA DE ARTEFATOS E AVALIAÇÃO DE VÍDEOS DIGITAIS. 86 f. Trabalho de Conclusão de Curso – Curso Superior de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

Este projeto teve como objetivo aperfeiçoar o *software* SASQV de avaliação subjetiva e objetiva da qualidade de vídeos digitais, assim como desenvolver e integrar, em uma plataforma unificada de avaliação de qualidade, ferramentas para a adição controlada de artefatos em vídeos. A manipulação destes artefatos por meio de parametrização permite a adição de efeitos de borramento e blocagem, assim como a simulação de artefatos decorrentes da perda de pacotes em uma transmissão de vídeo por *streaming*.

Palavras-chave: vídeo digital, geração artificial de artefatos, avaliação de qualidade de vídeo, avaliação subjetiva, avaliação objetiva

ABSTRACT

ANDREATTA, Caio N.; BRAGA, Brunno A. W.. SASQV2: A CONTROLLED ARTIFACT ADDITION AND DIGITAL VIDEO ASSESSMENT ENVIRONMENT. 86 f. Trabalho de Conclusão de Curso – Curso Superior de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

This project aimed to improve the SASQV software of subjective and objective video quality assessment, as well as to develop and integrate, into an unified platform for assessing video quality, tools for adding artifacts in a controlled manner. The manipulation of these artifacts by means of parameterization allows the addition of blocking and blurring effects and the simulation of artifacts resulting from packet loss in a transmission of streaming video.

Keywords: digital video, artificial artifact generation, video quality assessment, subjective assessment, objective assessment

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AC	Arithmetic Coding
A/D	Analógico/Digital
BPP	bits por pixel
D/A	Digital/Analógico
DC	Differential Coding
DCT	Discrete Cosine Transform
DSCQS	Double Stimiulus Continuous Quality Scale
DSIS	Double Stimulus Impairment Scale
DVD	Digital Video Disk
EFR	Effective Frame Rate
ER	Entidade-Relacionamento
FPS	Frames Per Second
FRF	Frame Repetition Factor
Full-HD	Resolução Full-HD de 1080 linhas contendo 1980 *pixels* cada.
GCC	GNU Compiler Collection
GNU	Projeto de desenvolvimento de software livre
GOP	Group Of Pictures
GPL	General Public License
GUI	Graphical User Interface
H.262	Padrão de compressão de vídeo digital também conhecido MPEG-2 Part 2
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JRE	Java Runtime Environment
JVM	Java Virtual Machine
LGPL	Lesser General Public License
MOR	Mapeamento Objeto Relacional
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
MSSIM	Mean Structural SIMilarity Index
MSU	Moscow State University
PSNR	Peak Signal to Noise Ratio
SD	Standard Definition
SDSCE	Simultaneous Double Stimulus for Continuous Evaluation
SDTV	Standard Definition Television
SGBD	Sistema Gerenciador de Banco de Dados
SMPTE	Society of Motion Picture and Television Engineers

SSCQE	Single Stimulus Continuous Quality Evaluation
SSIM	Structural SIMilarity Index
SVH	Sistema Visual Humano
TKN	Telecommunication Networks Group
TS	Transport Stream
UNIX	Sistema Operacional

SUMÁRIO

1 INTRODUÇÃO

Há algumas décadas o formato digital de vídeo tornou-se viável comercialmente, mais precisamente em 1986 com o formato D-1 da Sony, que armazenava imagens não-comprimidas em definição padrão (SD - Standard Definition). Por ser muito caro, o D-1 foi utilizado apenas por grandes emissoras de televisão. Posteriormente, este formato foi substituído por outros que utilizavam a compressão de vídeo, tornando o equipamento mais barato e acessível (WIKIPEDIA, 2012e).

Na década de 1990 surgiram algumas padronizações, como o MPEG-1 e o MPEG-2, que estabeleceram padrões de codificação com perdas tanto para vídeo quanto para áudio, baseando-se na capacidade de armazenamento e na banda de transmissão disponíveis (CORPORATION, 2008).

Ao longo dos anos seguintes a eletrônica se desenvolveu bastante surgindo, desta forma, uma diversidade de dispositivos que permitiam a obtenção e manipulação de imagens e vídeos digitais: dispositivos portáteis como câmeras de vídeo, máquinas fotográficas, cartões de memória, *webcams*, etc. e mais recentemente celulares, *smartphones* e *tablets*. A cada dia que passa, o formato digital torna-se mais acessível, seja pelos meios de comunicação e notícias, entretenimento, educação, entre outros.

Além de acessível, também é necessário que o vídeo chegue a essas pessoas com um nível de qualidade que as possibilitem contemplar e interpretar as imagens de forma natural, sem produzir degradação ou desconforto perceptível ao Sistema Visual Humano (SVH), ou seja, sem que existam perdas ou distorções exageradas em seu conteúdo.

No entanto, processos como aquisição, compressão, armazenamento e transmissão, os quais viabilizam e popularizam a difusão de vídeo, são muitas vezes responsáveis por também introduzir artefatos que degradam a qualidade final da imagem (DARONCO, 2009; WANG et al., 2004).

No intuito de melhor avaliar o efeito que tais artefatos podem produzir no SVH foram desenvolvidas métricas objetivas e subjetivas de avaliação de qualidade de vídeo. O projeto

SASQV, desenvolvido por (SANTOS et al., 2010), busca implementar ambas as formas de avaliação em um mesmo conjunto de *software* e *hardware* onde é possível gerar artefatos artificiais, coletar avaliações subjetivas e aplicar métricas objetivas sobre uma base de vídeos, bem como utilizar ferramentas para análise e comparação dos dados obtidos.

O presente trabalho propõe adaptações ao SASQV, buscando aperfeiçoar a ferramenta de geração de artefatos no sentido de torná-los mais similares àqueles encontrados nas trasmissões, fornecendo maior grau de liberdade para a manipulação dos mesmos, além de agregar novos tipos, como por exemplo a simulação de um *streaming* de vídeo. Dada a natureza do projeto SASQV, o qual se utiliza de *software* e bibliotecas distribuídos sob licenças de *software* livre, este projeto também se propõe a construir uma nova *interface* gráfica, não baseada em tecnologias proprietárias, que permita a degradação de vídeos, controle de sessões de avaliação objetiva ou subjetiva e visualização de resultados através de gráficos.

1.1 MOTIVAÇÕES

Este trabalho é motivado pela ampla difusão do vídeo digital, implantado principalmente na forma de TV digital e de *streaming* via internet, sendo o segundo objeto de grande interesse no mercado. De acordo com SANDVINE (2012), no continente norte-americano cerca de 37% do tráfego de internet fixa é dedicado ao *streaming* de vídeo durante o horário onde tradicionalmente ocorre o pico de audiência televisiva, atingindo 41% no caso da internet móvel dentro do mesmo horário.

As estatísticas levantadas por YOUTUBE (2012), uma das maiores bases de vídeos digitais *online* hoje em dia, indicam que em seu site:

- São recebidos mais de 800 milhões de usuários únicos por mês;
- São assistidas mais de 3 bilhões de horas de vídeo a cada mês;
- São armazenadas 72 horas de vídeo a cada minuto;
- São assistidos 500 anos de vídeo através da rede social *Facebook* todos os dias.

Dada a ampla utilização do formato digital quase que em todos os meios de comunicação e entretenimento, avaliar a qualidade de vídeo é uma necessidade para qualquer sistema dedicado a este propósito. O conhecimento da relação entre atributos envolvidos na transmissão e opinião dos telespectadores pode permitir aos operadores do sistema saber qual o impacto de cada atributo na qualidade do vídeo disponibilizado.

A causa das degradações em transmissões digitais se deve ao processo de compressão com perdas que é realizado a fim de se reduzir a quantidade de dados a ser enviada. Dependendo da taxa de compressão e dos codificadores envolvidos, esse processo pode gerar alguns artefatos como blocagem e borramento, diminuindo a qualidade original. Já o efeito de travamento se deve à perda de informações devido ao baixo nível de sinal ou perda de pacotes (ALBINI, 2009).

Além possibilitar o estabelecimento da relação *atributos* X *opinião geral*, a utilização de vídeos degradados artificialmente diminui o custo e a dificuldade em se obter vídeos sincronizados para avaliação, uma vez que não são necessários equipamentos de transmissão, recepção e codificação, nem uma rede de computadores para a simulação de *streaming*.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

- Aprimorar a degradação de vídeos, reprodução, manipulação de sessões, interação e portabilidade da ferramenta SASQV.

1.2.2 OBJETIVOS ESPECÍFICOS

- Aprimorar os algoritmos de geração de artefatos de vídeo digital (blocagem e borramento), permitindo ao usuário manipular os parâmetros de cada artefato;
- Adaptar a ferramenta original para manipular vídeo bruto no formato .yuv;
- Adicionar à ferramenta um simulador de transmissões de vídeo via rede, simulando o *streaming* e seus possíveis artefatos;
- Desenvolver uma nova interface gráfica em linguagem Java de funcionalidade similar à do SASQV;
- Aprimorar a portabilidade da ferramenta para sistemas operacionais diversos.

1.3 ESTRUTURA DO RELATÓRIO

Os conceitos presentes na literatura correlata e que servem de base para analisar e justificar as decisões de projeto estão descritos no Capítulo 2. Neste capítulo há uma breve introdução ao sistema visual humano, uma descrição aprofundada de vídeos digitais e seus

artefatos, apresentação de métricas de avaliação de vídeo e um levantamento de trabalhos relacionados, bem como uma análise comparativa destes.

O Capítulo 3 apresenta o planejamento do desenvolvimento do software. Inicialmente é apresentado o levantamento de seus requisitos e em seguida são apresentadas as especificações, as decisões de projeto e arquitetura bem como suas justificativas.

Neste capítulo as modificações necessárias ao software original também são levantadas. Por fim, um planejamento de trabalho contendo as atividades a serem desenvolvidas pela equipe é apresentado.

No Capítulo 4 está descrito o desenvolvimento do projeto: as ferramentas que permitiram seu desenvolvimento e o controle de suas versões. Em seguida são apresentadas e detalhadas as ferramentas desenvolvidas, a interface gráfica bem como os diagramas de sequência e de classes que descrevem metodicamente a estrutura e funcionamento do software.

A partir do término do desenvolvimento do software proposto, a validação das ferramentas desenvolvidas é realizada comparativamente a ferramentas semelhantes já conhecidas na área. Os resultados são apresentados no Capítulo 5. O principal resultado do projeto é apresentado em seguida: o impacto dos parâmetros inicialmente escolhidos pelo usuário sobre o resultado das métricas objetivas escolhidas.

Para encerrar, são apresentadas as conclusões do projeto e as sugestões para trabalhos futuros que pretendam continuar o aprimoramento da ferramenta.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 VÍDEO DIGITAL

Uma imagem é um registro aproximado de um instante de tempo do mundo real, pois representa uma quantidade de informação limitada suficiente para que qualquer pessoa possa, posteriormente, reconhecer aquela representação assimilando-a à uma realidade. Primeiramente, imagens são registros porque contém em si a captura de cores num instante de tempo, que nada mais são que intensidades luminosas (ondas eletromagnéticas que o olho humano é capaz de perceber).

Imagens são registros aproximados, ou limitados, pois dependem da tecnologia que as obtém e manipulam: as cores nem sempre são fiéis, a iluminação pode acabar atrapalhando a nitidez e a saturação, etc. Por fim, imagens usualmente são registros suficientes porque costumam permitir sua associação a situações e formas reais por qualquer pessoa apesar de suas limitações.

A tecnologia na obtenção de imagens se expande a cada dia, seja aprimorando as técnicas já utilizadas, seja criando novas técnicas. Até algumas décadas atrás, a única forma de obtenção era a analógica: através de filmes sensíveis a luz que a ela deveriam ser expostos por um infinitésimo de segundo através de um obturador; fitas magnéticas, etc.

Para a obtenção de sinais digitais, o sinal analógico (uma imagem é um sinal analógico) deve ser submetido às fases de amostragem e quantização, em que uma amostra é gerada em um instante de tempo (REHME, 2007).

Atualmente a forma digital de captura de imagens e vídeos, criada no fim da década de 1960, está bastante difundida por causa das vantagens que oferece, entre elas:

- editar, adicionar efeitos, corrigir imperfeições, enfim, a manipulação é mais fácil e algumas operações só podem ser realizadas neste formato;
- o arquivo não perde a qualidade ao longo do tempo, como acontece com fitas magnéticas

e filmes, onde o meio de armazenamento afeta diretamente a qualidade do conteúdo;

- a cópia é fiel, não causando modificação no conteúdo;
- o custo de armazenamento tem se tornado cada vez mais baixo;
- a integração com outras tecnologias e equipamentos é mais simples;
- possibilidade de correção de erros a partir de *checksums* e redundância;
- possibilidade de compressão com ou sem perdas;
- a repetibilidade da informação, em qualquer momento.

Vídeos digitais são sequências de imagens digitais, e a maioria conceitos do segundo valem para o primeiro. O contrário pode não ser verdadeiro, uma vez que em um vídeo a presença da variável tempo permite-lhe operações exclusivas, como por exemplo a compressão temporal ou codificação inter-frame, detalhada mais adiante.

A gama de aplicações de vídeos digitais é ampla. Alguns exemplos são: transmissão de TV, *streaming* via internet, exames médicos, uso pessoal, web cams, câmeras em celulares e smartphones, câmeras de segurança, cinema, imagens via satélite, entre outros.

As limitações da tecnologia digital residem na capacidade de armazenamento de dados digitais, velocidade de armazenamento, dispositivos de captura e até mesmo na capacidade humana (por exemplo, de diferenciar cores, tonalidades, etc). Essas limitações moldam a representação digital de imagens, que são formadas por *pixels* (menor estrutura representativa de cor). Desta forma, a informação que se pode agregar a cada *pixel* é limitada e, consequentemente, afeta diretamente a quantidade de cores que cada *pixel* pode assumir.

Antes de explicar sobre o processo de quantização que determinará o conteúdo de cada *pixel* e seu formato, é importante que o conceito de modelo de cores seja introduzido.

Um modelo de cores é um modelo matemático que associa números a cores, ou seja, é uma função matemática de tuplas, de normalmente três ou quatro variáveis, que representam cores no espaço correspondente.

Várias são as formas de construir um modelo de cores e dentre alguns deles estão: RGB (componentes red-green-blue) (Figura 1), YUV (componentes luminância-crominância-crominância) (Figura 2) e CMYK (componentes cyan-magenta-yellow-black) (Figura 3).

O modelo de cores RGB possui três componentes e a partir da combinação delas é possível criar uma infinidade de cores. Da mesma forma, o modelo de cores CMYK combina as

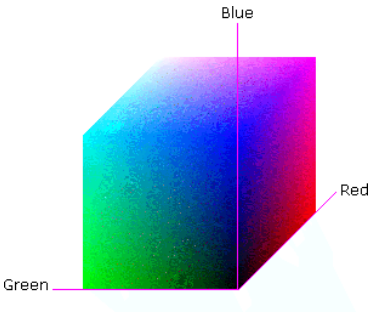

Figura 1: RGB
Fonte: (SYSTEMS, 2012)

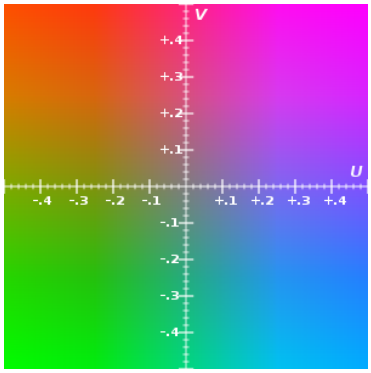

Figura 2: YUV, para Y = 0.5
Fonte: (WIKIPEDIA, 2012n)

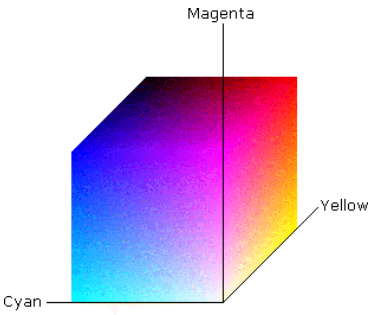

Figura 3: CMYK
Fonte: (SYSTEMS, 2012)

componentes ciano, magenta, amarelo e preto para formar cores. Já no modelo YUV existe uma componente de luminância, ou brilho, e duas componentes de cor efetivamente (na realidade, essas componentes são respectivamente a diferença entre valores padrões de azul e vermelho e a luminância).

De posse destes conceitos iniciais, o significado de quantização torna-se mais simples. Em uma imagem analógica, cada partícula da imagem pode assumir infinitos valores pois a intensidade luminosa registrada, por exemplo em um filme, é uma grandeza física com domínio real. Já em uma imagem digital, cada *pixel* deve ter uma quantidade finita de informação para que possa ser armazenado. Desta necessidade de se limitar o tamanho da informação surge o conceito de quantização: um espaço de cores é dividido em subespaços em que apenas uma cor o representa. Este processo pode ser visto como uma compressão com perdas, pois várias cores são perdidas.

Partindo-se da quantização, é necessário determinar a quantidade de bits por *pixel* (BPP ou profundidade de cor) que é equivalente ao número de níveis de intensidade de cada componente de cor do espaço de cores desejado. Por exemplo, utilizando-se 8 bits para cada componente do modelo RGB, o espaço de cores fica limitado a 256 valores para cada componente que, combinados, geram mais de 16 milhões de cores (2^{24}).

No espaço de cores RGB mostrado na figura 4, é possível observar o efeito da quantização. Os eixos, que a princípio tinham como domínio os reais, foram discretizados para assumir apenas valores inteiros não-negativos. Cada componente pode assumir seis valores e a combinação delas gera um total de $6^3 = 216$ cores. Desta forma, pode-se perceber que cada subespaço assumiu um único valor para representá-lo, ignorando as tonalidades de transição entre cada ponto do espaço.

A quantização representa uma das vantagens da utilização de formatos digitais pois

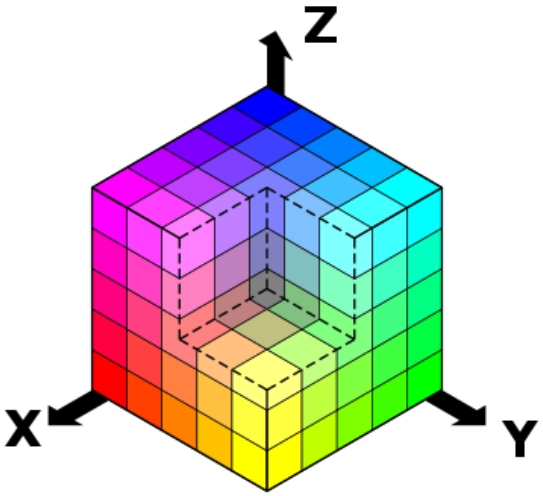

Figura 4: Espaço RGB discretizado.

Fonte: (WIKIPEDIA, 2012d)

muita informação redundante para o SVH (tons de cores muito próximos, por exemplo) pode ser removida. Neste aspecto, um ponto chave da fisiologia humana que favorece a compressão de imagens é o fato de a acuidade visual ser maior em relação à luminância do que à crominância (LAMBRECHT, 2001).

Um conceito importante que aproveita esta característica da visão humana é a subamostragem. Este conceito define que a resolução da camada de crominância pode ser menor que a resolução da camada de luminância, ou seja, para um conjunto de *pixels*, cada um possuirá seu valor de luminância, porém o de crominância poderá ser repetido, desta forma economizando recursos (BRICE, 2000). A figura 5 apresenta, em colunas, alguns esquemas de subamostragem.

Figura 5: Esquemas de subamostragem.

Fonte: (WIKIPEDIA, 2012c)

Onde:

- J - largura da amostra de referência, normalmente igual a 4;
- a - número de amostras de crominância (Cb, Cr) na primeira linha;
- b - número de amostras de crominância (Cb, Cr) na segunda linha.

O fator *alpha* (coeficiente de opacidade) pode ser considerado na notação, mas normalmente é omitido por ter valor igual a J. A notação dos esquemas é obtida considerando-se J:a:b. Desta forma, no esquema 4:1:1 é obtida uma amostra de crominância para cada linha; no esquema 4:2:0 são obtidas duas amostras na primeira linha que serão repetidas na segunda; no esquema 4:2:2 uma amostra de crominância é feita para cada conjunto de dois *pixels* (na horizontal) e, por fim, no esquema 4:4:4 não há subamostragem, uma vez que a crominância é amostrada para cada *pixel*, não ocorrendo compressão por não haver possibilidade de repetição.

Em valores reais, considerando-se 24 bpp (8 para cada componente de crominância e 8 para a componente de luminância), uma imagem sem subamostragem (esquema 4:4:4) de dimensões 4x2, como a da figura, equivaleria à 24x4x2 = 192 bits. A tabela 1 sintetiza o valor em bits desta mesma imagem para cada esquema de subamostragem.

Tabela 1: Tamanho em bits da mesma imagem (4x2) para cada esquema de subamostragem.

Esquema	Total de bits de luminância	Total de bits de Crominância	Total de bits
4:4:4	8x4x2 = 64	(8+8)x4x2 = 128	192
4:1:1	8x4x2 = 64	(8+8)x2 = 32	96
4:2:0	8x4x2 = 64	(8+8)x2 = 32	96
4:2:2	8x4x2 = 64	(8+8)x2x2 = 64	128

O vídeo digital com qualidade *standard* (SDTV) é definido pela norma SMPTE 259M (Society of Motion Picture and Television Engineers). Possui dimensões de 720x480 *pixels* com esquema de subamostragem 4:2:2. O padrão de DVD (Digital Video Disk) utiliza o esquema 4:2:0 de subamostragem.

2.1.1 ARQUIVO EM FORMATO BRUTO

É possível armazenar diretamente os valores das amostras em um arquivo. Este tipo de arquivo, conhecido como arquivo raw, ou bruto, não passa por qualquer processamento, codificação ou encapsulamento. Existe uma grande quantidade de formatos de arquivo de vídeo bruto, cada um com uma organização particular que ordena os valores das componentes de cada

pixel, entre eles os formatos: AYUV, UYVY, CYUV, YUY2, Y41P, Y411, YUVP, Y211, YV16, YV9, Y800 (FOURCC, 2011).

A Figura 6 mostra a organização de um arquivo no formato IYUV (ou I420) de subamostragem 4:2:0 e o respectivo fluxo de bytes em memória, para uma imagem de 24 *pixels*. Primeiramente são armazenados os valores de luminância de cada *pixel*, os valores de crominância U (Cb) para cada conjunto de 4 *pixels* e os de crominância V (Cr) para o mesmo conjunto.

Figura 6: Organização de um arquivo no formato IYUV.
Fonte: (WIKIPEDIA, 2012n)

2.2 CODIFICAÇÃO

O objetivo principal da codificação de vídeos é a compressão de dados que torna viável seu armazenamento e transmissão (DARONCO, 2009).

A codificação ou compressão é o processo que reduz o número de bits de dados digitais a fim de se diminuir a necessidade por recursos, tais como espaço de armazenamento e banda de transmissão. A compressão possui duas categorias básicas: pode ser sem perdas, isto é, apenas informações estatisticamente redundantes são removidas conservando a informação original; e pode ser com perdas, em que informações menos relevantes são descartadas pela necessidade imposta.

Um exemplo de como a compressão é importante atualmente pode ser verificado na transmissão de vídeos por streaming. Considerando-se apenas os valores de luminância (escala de cinza) de um vídeo com definição standard (SD - Standard Definition) de 640x480 *pixels* com 8 BPP, tem-se que cada quadro de imagem ocupa 8x640x480 = 2.45 Mbits de memória. Se o vídeo deve ser apresentado a uma taxa de 30 FPS (*frames per second* - quadros por segundo), a banda necessária para transmitir o vídeo deve ser de 30x2.45 Mbits/s = 73.50 Mbps.

Adicionando-se informações de cor e de áudio, esta taxa chega facilmente aos 270 Mbps numa qualidade SD, e ultrapassa 1.4 Gbps numa qualidade HD (High Definition - alta definição), valores que teriam um custo impraticável para usuários domésticos e até mesmo para empresas (SYSTEMS, 1999). A codificação permite que até 98% do sinal digital original seja removido sem que haja uma degradação inaceitável na qualidade da imagem (CORPORATION, 2008).

A Figura 7 apresenta o processo de obtenção, codificação (etapa A), transmissão, recepção e decodificação (etapa B) de um sinal de áudio e/ou vídeo sendo, por fim, apresentado. As etapas de conversão A/D (analógico-digital) e conversão D/A (digital-analógico) podem ser suprimidas caso a conversão A/D já seja realizada nos equipamentos de obtenção de dados e caso os monitores e dispositivos de armazenamento sejam digitais (REHME, 2007).

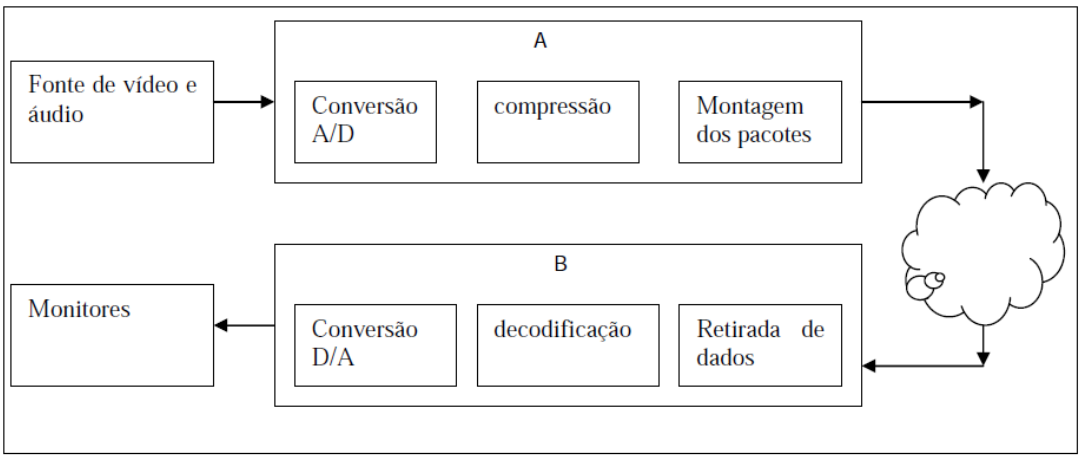

Figura 7: Etapas de codificação e decodificação.
Fonte: (REHME, 2007)

As etapas gerais da compressão são:

- obter as diferenças entre quadros;
- estimar movimento;
- realizar transformações nos domínios espacial e de frequência.

A compressão espacial opera num mesmo *frame* e tem por objetivo remover possíveis redundâncias para diminuir a quantidade de dados que o representam. Já a compressão temporal opera numa sequência de *frames*, buscando encontrar semelhanças entre eles, uma vez que em *frames* sucessivos ocorrem poucas modificações, na maior parte do tempo. Desta forma, a parte do *frame* que não sofreu alterações é repetida do anterior e apenas a parte modificada é efetivamente utilizada para descrever o *frame* a ser apresentado.

De acordo com REHME (2007): “Para a compressão temporal, precisa-se de um ponto de partida ou quadro-chave. Depois dele, apenas as diferenças são descritas”.

Em relação aos mecanismos de codificação de fonte, REHME (2007) afirma que “quanto mais sofisticados forem, normalmente melhor é a relação qualidade versus taxa de bits, porém apresentam maior custo em termos de capacidade de processamento e maior tempo requerido para executar a compressão. Para várias situações, o aumento do atraso entre a captura da imagem e sua apresentação não é tolerável”.

Percebe-se desta forma que existe um equilíbrio entre método e taxa de compressão, degradação da qualidade aceitável e recursos disponíveis, ambos dependentes da aplicação.

Em 1987, a IEC (Comissão Eletrotécnica Internacional) juntamente com a ISO (Organização Internacional para Padronização) organizou um grupo de especialistas com o objetivo de padronizar a compressão de áudio e vídeo digitais, mais conhecido como Moving Picture Experts Group, ou MPEG. Em seu primeiro encontro, realizado em 1988, após o sucesso da digitalização do sinal de áudio (de natureza analógica) e posterior compressão para o desenvolvimento do popular CD (com áudio de alta qualidade), percebeu-se que aplicar a ideia na transmissão de TV diminuiria a largura de banda necessária, possibilitando a emissão de programas interativos, serviços de internet e ainda mais programas (CORPORATION, 2008).

O MPEG desenvolveu uma série de protocolos para as questões que surgiram ao longo do desenvolvimento de novas tecnologias, novos equipamentos e novos padrões internacionais, a fim de padronizar emissões digitais. Os padrões desenvolvidos incluem MPEG-1, MPEG-2, MPEG-4 e, mais recentemente, MPEG-7.

Atualmente, a indústria de transmissão de TV se baseia principalmente no padrão MPEG-2 para transmitir sinal de vídeo digital, áudio e dados pela rede. Este é o padrão para transmissão digital, pois é o único que prevê suporte a transmissão de dados via rede (CORPORATION, 2008).

Os dois principais tipos de compressão do MPEG são as codificações espacial e temporal. O primeiro tipo elimina redundâncias entre *pixels* adjacentes num mesmo *frame*, aproveitando-se, também, da inabilidade do olho detectar certas degradações. O segundo tipo minimiza redundâncias entre *frames* (CORPORATION, 2008).

O processo de codificação espacial envolve as seguintes etapas, segundo (CORPORATION, 2008):

- Transformada Discreta de Cossenos - DCT: cada *frame* é dividido em matrizes de di-

mensões 8x8 *pixels* em que as intensidades luminosas serão transformadas em valores baseados em frequência, chamados coeficientes. Por causa da redundância espacial, muitos coeficientes resultam no valor zero, que podem ser eliminados da série de coeficientes. O resultado é uma compressão com perdas imperceptíveis ao olho humano, em que o mesmo *frame* é representado por menos *bits*.

- Quantização (*Quantization*): os coeficientes da DCT são reorganizados em ordem de importância visual.
- Ponderação (*Weighting*): degradação, ou ruído, são estrategicamente adicionados em regiões mais detalhadas ou complexas, onde é pouco provável que o observador note.
- Varredura (*Scanning*): nesta etapa os coeficientes mais significantes são enviados primeiro, seguidos dos menos importantes e, por fim, uma indicação de que os restantes são iguais a zero.
- Codificação de Entropia (*Entropy Conding*): os coeficientes são reajustados de acordo com sua frequência de ocorrência. Os que mais se repetem são expressos com menor número de *bits*, o que diminui consideravelmente a banda necessária para transmiti-los.

Já a codificação temporal aproveita-se da similaridade entre *frames* sequenciais, codificando apenas a diferença entre eles. Isso é possível devido a dois tipos de codificação temporal: previsão entre *frames* e previsão de movimento (*inter-frame prediction* e *motion prediction*, respectivamente).

A previsão entre *frames* codifica um *frame* completo apenas periodicamente. O resultado, chamado de *intra-coded frame* ou I-*frame*, é utilizado como referência para *frames* anteriores e posteriores.

Predicted Frames, ou P-*frames*, têm por referência um I-*frame* ou um P-*frame* anterior, ou seja, ao invés de transmitir todos os coeficientes DCT, o codificador transmite apenas aqueles coeficientes que sejam diferentes do I-*frame* ou um P-*frame* anterior. No outro lado, o decodificador reconstrói um P-*frame* apenas aplicando as diferenças sobre o I-*frame* ou P-*frame* de referência (CORPORATION, 2008).

Bidirectionally Predicted Frames, ou B-*frames*, podem utilizar tanto I-*frames* ou P-*frames* precedentes quanto subsequentes, como pode ser observado na Figura 8. O I-*frame* inicial é completamente codificado, já o P-*frame* subsequente possui apenas a parte que sofreu modificação em relação ao I-*frame* anterior. O B-*frame* presente na figura, armazena as diferenças em relação ao P-*frame* anterior e ao I-*frame* seguinte.

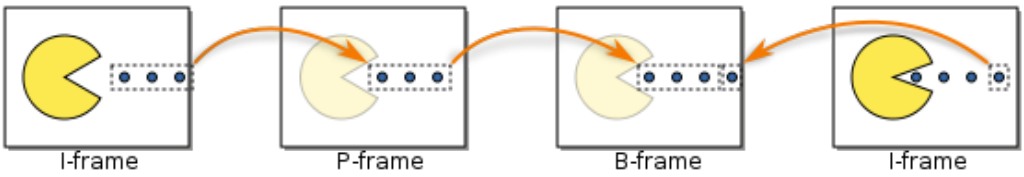

Figura 8: Estrutura de predição de *frames* no MPEG.
Fonte: (WIKIPEDIA, 2012m)

P-*frames* alcançam uma compressão maior que I-*frames*, tipicamente de 20% a 70% do tamanho do respectivo I-*frame* de referência. B-*frames*, por sua vez, podem permitir uma compressão ainda maior, tipicamente de 5% a 40% do tamanho do respectivo I-*frame* de referência (SYSTEMS, 1999).

Apesar de reduzir a quantidade de informação a ser transmitida, quanto menos I-*frames* são utilizados, maior é a vulnerabilidade da transmissão. Qualquer erro que ocorrer num I-*frame* será propagado até que um novo seja enviado corretamente (CORPORATION, 2008). Por este motivo o conceito de GOP (*Group Of Pictures*) é utilizado (CORPORATION, 2008). Cada conjunto é iniciado por um I-*frame*, contendo também P-*frames* e B-*frames*.

O tamanho usual para um GOP é entre 12 e 15 *frames* (SYSTEMS, 1999; CORPORATION, 2008). Um exemplo de GOP pode ser observado na Figura 9.

Figura 9: Estrutura de um GOP.
Fonte: (SYSTEMS, 1999)

A previsão de movimento, por sua vez, envia um vetor de movimento (*motion vector*) para o decodificador. Este vetor resulta da identificação de movimento de objetos que se repetem, desta forma, o decodificador apenas move o objeto para a nova posição.

Para que seja possível transportar vídeos codificados com MPEG por uma rede IP (Internet Protocol), a informação dos *frames* deve ser encapsulada em pacotes MPEG-TS, que serão transportados por pacotes IP posteriormente (SYSTEMS, 1999). Pacotes IP contém sete pacotes MPEG-TS de 188 *bytes*.

A Figura 10 apresenta o encapsulamento típico de um pacote IP. Cada pacote pode conter informação de dois *frames* consecutivos, ou seja, a perda de um pacote pode trazer a perda

de informação referente aos dois *frames*. Vale observar que além de informações de imagem, os pacotes IP também transportam informações de áudio codificado em MPEG e informações de serviço (SYSTEMS, 1999).

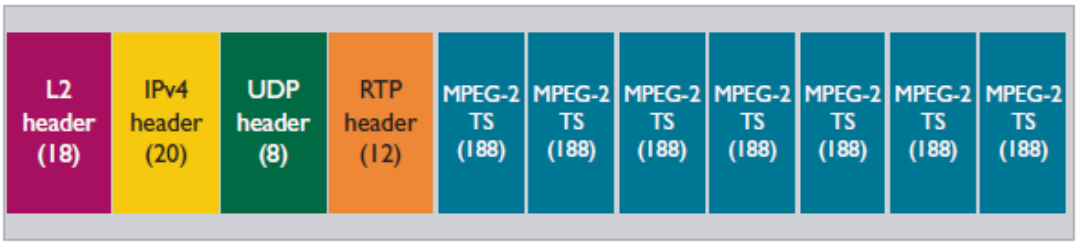

Figura 10: Estrutura de pacote IP.
Fonte: (SYSTEMS, 1999)

2.3 ARTEFATOS

Imagens digitais são alvo de uma grande variedade de distorções durante sua aquisição, processamento, compressão, armazenamento, transmissão e reprodução (WANG et al., 2004). As características peculiares que são observadas nas imagens são chamadas de artefatos (ALBINI, 2009) e podem ser observadas tanto em transmissões digitais quanto em transmissões analógicas.

As imagens digitalmente transmitidas, embora suscetíveis a um grande número de distorções, não sofrem as mesmas distorções que as convencionais, de transmissão analógica (salvo no caso de um sinal analógico ser convertido para ser transmitido em meio digital). O ruído branco gaussiano é um exemplo de distorção que ocorre em meio analógico e pode ser observado na Figura 11.

Figura 11: Ruído branco gaussiano.
Fonte: (PANAGIOTOPOULOU; ANASTASSOPOULOS, 2012)

Em se tratando de vídeos digitais, os artefatos de maior notoriedade são: *blockiness*, *blurriness*, *ringing*, e *noisiness* (FARIAS et al., 2007) e podem ser observados na Figura 12.

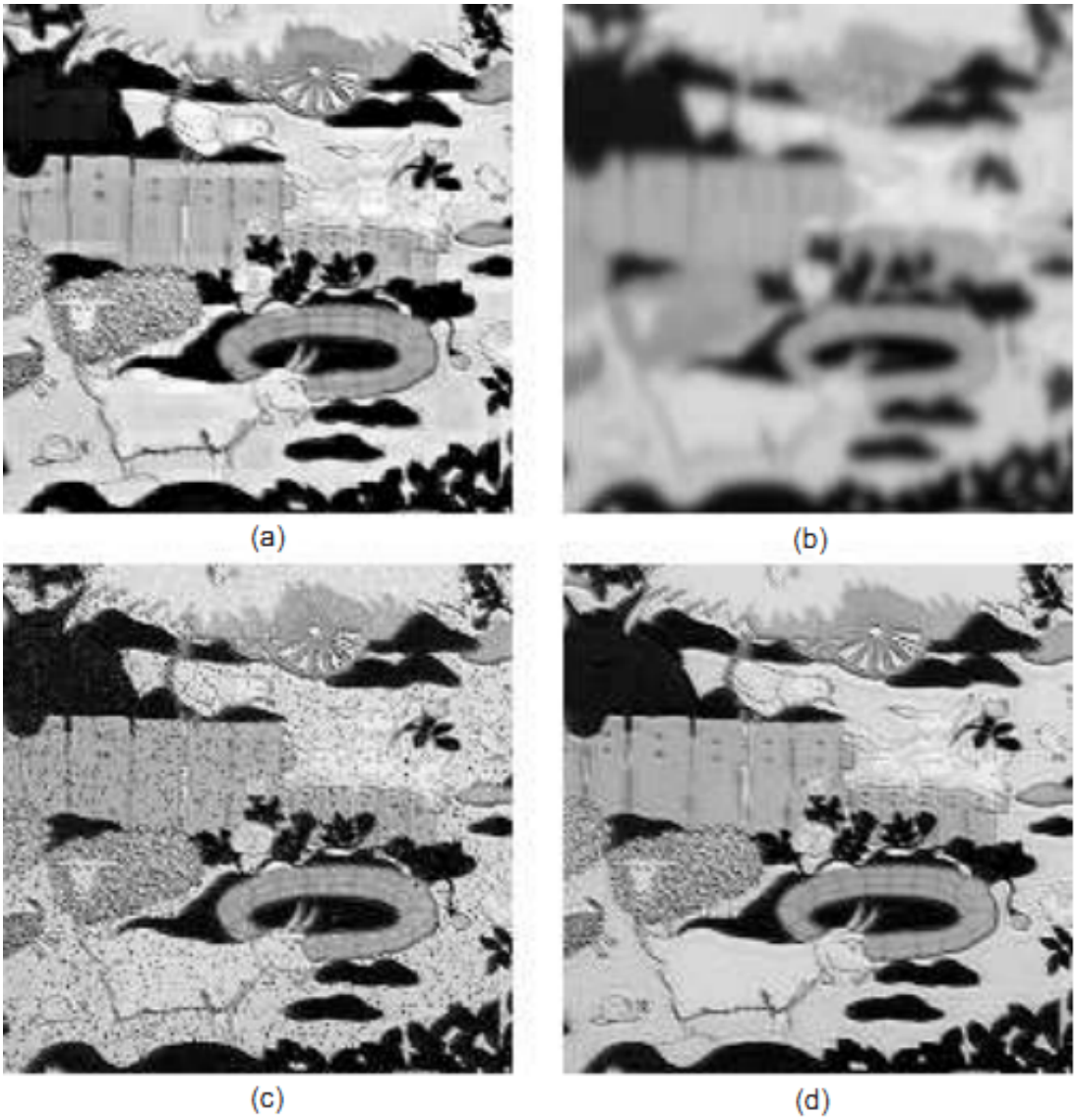

Figura 12: Artefatos digitais (a) blocagem; (b) borramento; (c) *noisiness* e (d) *ringing*.
Fonte: (FARIAS et al., 2007)

Poder adicionar artefatos de forma artificial e controlada a vídeos digitais é de grande importância no campo de avaliação de vídeos (tanto na avaliação objetiva quanto na avaliação subjetiva) uma vez que, a partir de algoritmos, é possível simular distorções reais. Estes algoritmos não são padronizados, mas existem algumas recomendações internacionais, como por exemplo a recomendação P.930 da ITU (International Telecommunication Union)(ITU, 1996) que descreve os princípios para reproduzir as distorções relativas à compressão de vídeo sem efetivamente precisar comprimi-lo.

Nas seções que seguem, alguns dos artefatos mais comuns serão detalhados juntamente com as recomendações para a respectiva simulação.

2.3.1 EFEITO DE BLOCAGEM - *BLOCKING EFFECT*

Este artefato é o mais estudado pela literatura (SILVA, 2009). Ocorre em função de uma quantização grosseira no processo de compressão, resultando em uma distorção ou perda de componentes de alta frequência (ITU, 1996). Normalmente ocorre em superfícies lisas em movimento (ITU, 1996) (FARIAS et al., 2007).

A recomendação P.930 indica que a simulação deste artefato deve, primeiramente, localizar regiões de movimento em uma imagem para posteriormente selecionar em quais possíveis janelas será aplicada uma média dos valores de luminância (ITU, 1996). Desta forma, o cálculo da DCT e todo o processo de compressão é evitado.

2.3.2 EFEITO DE BORRAMENTO - *BLURRING EFFECT*

O artefato de borramento é, visualmente, a perda de nitidez das bordas e de detalhes espaciais em uma imagem, como em áreas com texturas ou ao redor das bordas de objeto de cena (WU; RAO, 2005). É causado pelo algoritmo de compressão no momento em que escolhe codificar bits de resolução ou de movimento (ITU, 1996).

A recomendação P.930 fornece os filtros lineares a serem utilizados para a simulação deste artefato.

2.3.3 EFEITO DE TRAVAMENTO - *JERKINESS*

Este artefato é facilmente percebido em sistemas com baixa taxa de bits para vídeo, como em sistemas de teleconferência (ITU, 1996) em que a imagem parece congelar em determinados instantes. Isso se deve à repetição de *frames* com o objetivo de reduzir a quantidade de informação a ser transmitida ou processada (ITU, 1996).

A recomendação P.930 fornece dois fatores para controlar as repetições numa simulação: o FRF (*Frame Repetition Factor*) e o EFR (*Effective Frame Rate*). Um FRF igual a 3 causa a repetição do terceiro *frame* de uma sequência durante os próximos dois *frames*. O EFR é calculado como 30/FRF, e para o exemplo dado tem valor de 10 FPS.

2.4 MÉTRICAS DE AVALIAÇÃO

A compressão tende a diminuir a fidelidade, ou seja, o problema da codificação é encontrar um equilíbrio entre o nível de compressão necessário para transmitir pelo canal e o nível de fidelidade que se deseja exibir para o espectador (DARONCO, 2009).

Apesar de a transmissão digital garantir que a interferência a ruídos seja mínima, a qualidade da imagem não depende somente de interferência, como foi visto. A compressão de vídeos é excelente na perspectiva do transmissor já que o custo do equipamento torna-se menor, porém na perspectiva do receptor/usuário a compressão pode comprometer muito a qualidade.

Neste contexto de tentar equilibrar qualidade e compressão, surge a necessidade de se criarem métodos capazes de avaliar a qualidade dos vídeos transmitidos tanto da parte dos emissores, para saber até que ponto a compressão não é percebida pelo usuário, ou pelo menos que ela não o incomode a ponto de deixar de assistir a transmissão; quanto da parte do usuário, que colabora para ter um serviço melhor.

Diversas metodologias definidas em normas internacionais foram criadas para avaliar a qualidade da informação multimídia (tanto áudio quanto vídeo), divididas em dois paradigmas: avaliação objetiva e avaliação subjetiva, descritas a seguir.

2.4.1 MÉTRICAS OBJETIVAS

A avaliação objetiva tenta, através de algoritmos, fazer uma previsão aproximada da qualidade percebida pelos observadores (ALBINI, 2009). Os métodos mais simples e mais difundidos são definidos estatisticamente como o MSE (Erro Médio Quadrático) e o PSNR (Razão Sinal-Ruído de Pico) (SILVA, 2009) entre os dados originais e os dados recebidos (no caso de vídeos, o cálculo é aplicado *pixel* a *pixel*).

Outra métrica objetiva, o SSIM (Índice de Similaridade Estrutural), foi elaborada na tentativa de se aproximar a avaliação às percepções do SVH, pois leva em consideração a luminância, estrutura e o contraste (SILVA, 2009). Uma métrica derivada do SSIM é o MSSIM (Índice de Similaridade Estrutural Média), sendo uma média dos valores SSIM calculados a partir do vídeo avaliado (WANG et al., 2004).

2.4.1.1 MSE - MEAN SQUARED ERROR

O MSE é a média das diferenças ao quadrado entre os valores de nivel de cinza dos *pixels*. Considerando que o nível de cinza de cada *pixel* é dado em função de sua posição de

acordo com a equação a seguir:

Equação: G = f(t, x, y), onde:

- G = nivel de cinza do *pixel*, dada sua posição;
- t = *frame* onde o *pixel* está localizado;
- x = posição relativa ao eixo vertical no frame;
- y = posição relativa ao eixo horizontal no frame.

o MSE pode ser calculado conforme a seguinte equação:

Equação:

$$MSE = \frac{1}{(T \cdot X \cdot Y)} \sum_{t}^{T} \sum_{x}^{X} \sum_{y}^{Y} (P_1 - P_2)^2$$

para vídeos com T *frames* de tamanho X x Y (WINKLER, 2005).

Conforme a medida obtida, a seguinte análise é feita: quanto menor o valor do MSE, mais próxima a imagem avaliada está da imagem original (ALBINI, 2009). A imagem da função MSE é o conjunto dos reais, maiores ou iguais a zero, ou seja, um valor zero indica que as imagens são idênticas.

2.4.1.2 PSNR - PEAK SIGNAL TO NOISE RATIO

O PSNR é calculado com base num valor MSE e é dado em escala logarítmica. O PSNR mede a fidelidade entre imagens, ao contrário do MSE que mede diferenças entre imagens (WINKLER, 2005). Esta medida é encontrada conforme a equação a seguir:

Equação:

$$PSNR_{dB} = 10 \cdot log_{10} \frac{(2^n - 1)^2}{MSE}$$

onde n é o número de bits que representa o nível de cinza de cada *pixel*.

A popularidade na utilização desta métrica se deve ao fato de que o cálculo pode ser realizado rapidamente, sendo bastante utilizado para comparar vídeos comprimidos e descomprimidos (SILVA, 2009; RICHARDSON, 2003).

Dentre as limitações desta métrica está a necessidade de sincronização entre as imagens original e degradada e a baixa correlação com as métricas subjetivas definidas na norma ITU-R (SILVA, 2009; ITU, 2002).

Analisando os resultados, valores de PSNR altos indicam alta fidelidade entre as imagens comparadas, enquanto que valores baixos indicam baixa fidelidade.

2.4.1.3 SSIM - STRUCTURAL SIMILARITY INDEX

Esta métrica surgiu com o objetivo de melhor aproximar a avaliação objetiva à percepção humana, pois métricas como o MSE e o PSNR se mostraram ineficientes para tanto (SILVA, 2009). Aplicado sobre valores de luminância, contraste e a estrutura para várias janelas de mesmas dimensões em uma imagem (WANG et al., 2004), tem como equação:

Equação:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

onde:

- x e y são janelas sendo comparadas de dimensões NxN;
- μ_x é a média de x;
- μ_y é a média de y;
- σ_x^2 é a variância de x;
- σ_y^2 é a variância de y;
- σ_{xy} é a covariância de x e y;
- C1 = $(k_1 L)^2$, C2 = $(k_2 L)^2$ são duas variáveis para estabilizar a divisão;
- L é a faixa dinâmica dos valores dos *pixels* (normalmente é $2^{bitsporpixel} - 1$);
- k_1 = 0.01 e k_2 = 0.03. Estes são valores são indicados pelos autores do método (WANG et al., 2004) em decorrência de experimentos realizados.

2.4.1.4 MSSIM - MEAN STRUCTURAL SIMILARITY INDEX

Em termos práticos, normalmente é desejado encontrar-se uma medida de qualidade geral de toda a imagem (WANG et al., 2004). Por esta razão, a média dos índices SSIM é calculada:

Equação:

$$MSSIM(X,Y) = \frac{1}{M}\sum_{i=1}^{M} SSIM(x_j, y_j)$$

onde:

- X e Y são as imagens de referência e distorcida, respectivamente;
- x_j e y_j são as janelas dentro das imagens;
- M é o número total de janelas na imagem.

2.4.2 MÉTRICAS SUBJETIVAS

Por causa das limitações das métricas objetivas, muitos trabalhos foram realizados nos últimos anos para tentar desenvolver um teste objetivo mais sofisticado que se aproximasse dos resultados subjetivos (Watson et al., 1999; WANG et al., 2004). Embora muitas métricas tenham sido propostas, nenhuma ofereceu um nível de qualidade em comparação aos testes subjetivos (SCIENCES, 2003).

A avaliação subjetiva depende de observadores humanos que atribuem notas a partir de suas opiniões sobre a qualidade. Posteriormente, uma análise estatística dos dados coletados é realizada, resultando em uma nota chamada MOS (nota média de opinião) (ITU, 1996) (ALBINI, 2009). Estes tipos de avaliação possuem algumas recomendações estabelecidas por órgãos internacionais com a finalidade de padronizar os procedimentos e ambientes de avaliação, alguns exemplos destas normas podem ser observados na Tabela 2.

Tabela 2: Recomendações da ITU

Nome da Norma	Descrição
ITU-R Rec. BT.500	Metodologias para avaliação subjetiva da qualidade de vídeos em televisores
ITU-T Rec. P.910	Métodos para avaliação subjetiva de vídeos em aplicações multimídia
ITU-T Rec. P.911	Métodos para avaliação subjetiva de dados audiovisuais em aplicações multimídia
ITU-T J.144	Técnicas para avaliação objetiva de vídeo para televisão a cabo na na presença de uma referência
ITU-R BS.1387	Avaliação de sistema de áudio de alta qualidade.

Fonte: (DARONCO, 2009)

As metodologias de avaliação seguem diretrizes que dizem respeito aos critérios para escolha das sequências de imagens, à seleção de observadores, à duração das sessões de avaliação, ao tempo de exposição de cada sequência e intervalos de troca, às condições ambiente, etc.

2.4.2.1 MÉTODO DSIS - DOUBLE STIMULUS IMPAIRMENT SCALE

Comumente usado para avaliação de novos sistemas ou dos prejuízos decorrentes de transmissão. Neste método, ao avaliador é apresentada uma imagem de referência e posteriormente a mesma imagem degradada. As sessões, que devem durar até 30 minutos, podem seguir duas variantes: na primeira, o conjunto referência/degradada é apresentado e ao final o avaliador deve dar uma nota. Na segunda variante, o avaliador é submetido duas vezes ao conjunto referência/degradada e somente ao final lhe é permitido dar a nota. Em ambas a apresentação dos conjuntos referência/degradada é aleatória.

A Figura 13 apresenta as fases de apresentação para cada variante. Em T1 e T3 o avaliador deve observar as imagens. O período T2 é de transição, deve ser apresentada uma tela cinza por 3 segundos. Por fim, o período T4 é o de avaliação, onde o avaliador efetivamente pode dar a nota. A Tabela 3 sintetiza as fases e os períodos recomendados.

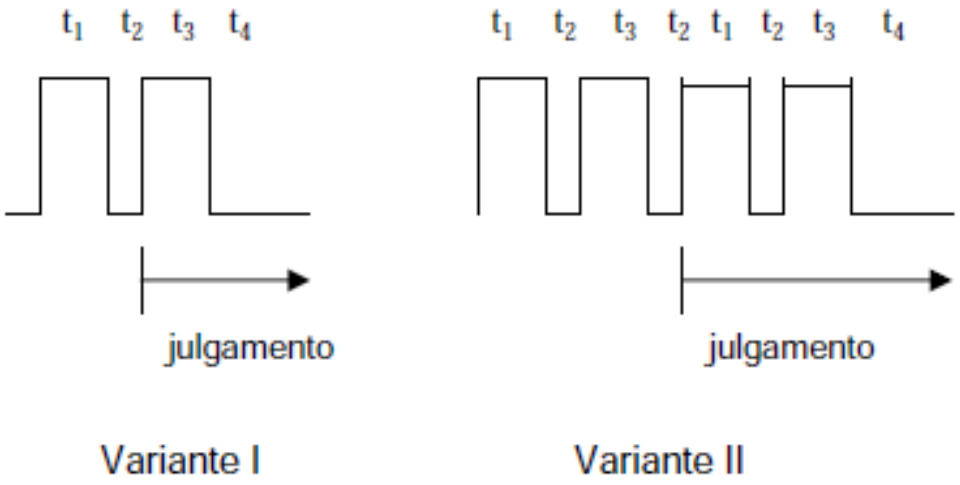

Figura 13: Variantes de aplicação da métrica DSIS.

Fonte: Adaptado de (ITU, 2002)

Tabela 3: Fases DSIS

T1 = 10 s	Imagem de referência
T2 = 3 s	Cinza intermediário, nível de vídeo 200mV
T3 = 10 s	Imagem degradada
T4 = 5-11 s	Cinza intermediário

Fonte: Adaptado de (ITU, 2002)

A escala de avaliação a ser utilizada é dada conforme a Tabela 4.

Tabela 4: Escala de avaliação DSIS

5	Imperceptível (*imperceptible*)
4	Perceptível, mas não irritante (*perceptible, but not annoying*)
3	Pouco incômoda (*slightly annoying*)
2	Irritante (*annoying*)
1	Bastante Irritante (*very annoying*)

Fonte: Adaptado de (ITU, 2002)

2.4.2.2 MÉTODO DSCQS - DOUBLE STIMIULUS CONTINUOUS QUALITY SCALE

Este método também é aplicável para avaliar os efeitos dos meios de transmissão sobre a qualidade da imagem. Assim como no método DSIS, as imagens são apresentadas aos pares. Uma deve ser a imagem original e a outra deve ser a imagem processada pelo sistema em teste. A ordem de qual será apresentada por primeiro, bem como a ordem de apresentação de cada par, é aleatória. Outra diferença do método DSIS está na avaliação: o avaliador deve dar nota para ambas as imagens.

O número de repetições dos vídeos depende da duração da sequência de teste. Cenas mais estáticas devem durar de 3 a 4 segundos, podendo ser repetidas 5 vezes (as duas últimas são avaliadas), já cenas mais dinâmicas, que possuam artefatos variantes no tempo, devem durar 10 segundos e ser repetidas duas vezes (a segunda é avaliada).

As fases da apresentação DSCQS ilustradas na Figura 14 são descritas na Tabela 5. Elas são semelhantes às fases do método DSIS, a diferença está no momento em que a avaliação é permitida.

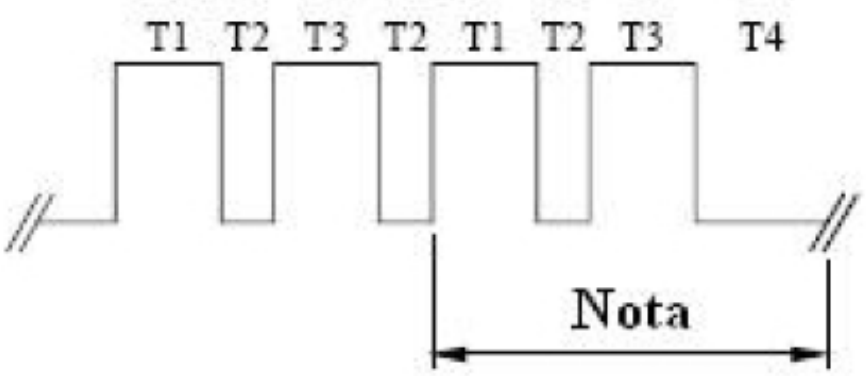

Figura 14: Fases de apresentação do método DSCQS.

Fonte: Adaptado de (ITU, 2002)

Tabela 5: Fases DSCQS

T1 = 10 s	Vídeo de teste A
T2 = 3 s	Cinza intermediário, nível de vídeo 200mV
T3 = 10 s	Vídeo de teste B
T4 = 5-11 s	Cinza intermediário

Fonte: Adaptado de (ITU, 2002)

Para realizar a avaliação, o avaliador deve marcar as notas em uma escala contínua conforme a apresentada na Figura 15.

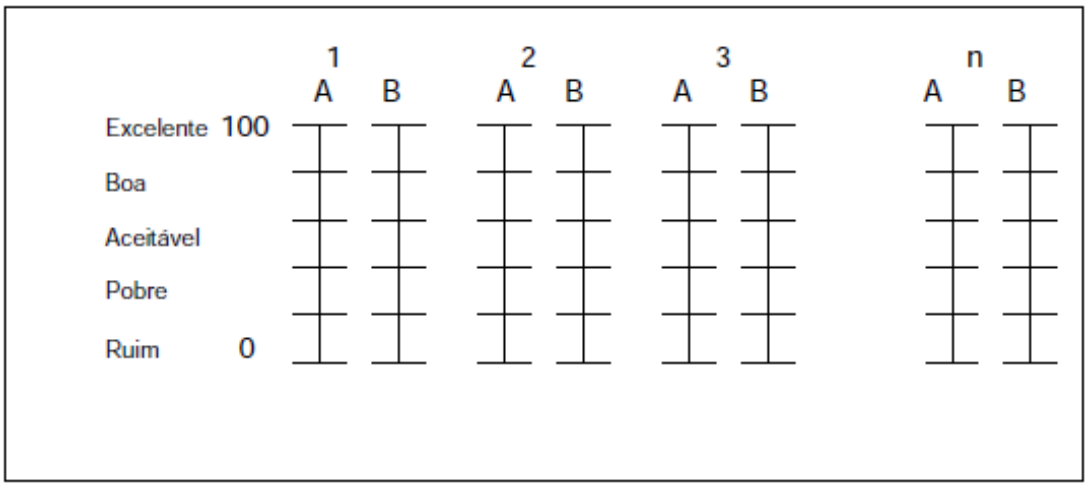

Figura 15: Escala de avaliação da métrica DSCQS.

Fonte: Adaptado de (ITU, 2002)

2.4.2.3 MÉTODO SSCQE - SINGLE STIMULUS CONTINUOUS QUALITY EVALUATION

Neste método, uma série de segmentos de programa são apresentados ao grupo de avaliadores com duração mínima de cinco minutos. Assim como no método DSCQS, a escala de avaliação é contínua. Os avaliadores modificam a nota conforme desejarem ao longo da apresentação das sequências. As notas, que são amostradas duas vezes por segundo, permitem o levantamento de histogramas (REHME, 2007).

Este método é mais adequado para avaliar a qualidade de vídeo em sequências longas e sem referência, o que é mais próximo da realidade.

2.4.2.4 MÉTODO SDSCE - SIMULTANEOUS DOUBLE STIMULUS FOR CONTINUOUS EVALUATION

Para este método, as sequências de referência e de teste são apresentadas de forma simultânea. O avaliador deve julgar a fidelidade do vídeo em teste utilizando a escala contínua, assim como nos métodos DSCQS e SSCQE. A Figura 16 demonstra como deve ser realizada a apresentação.

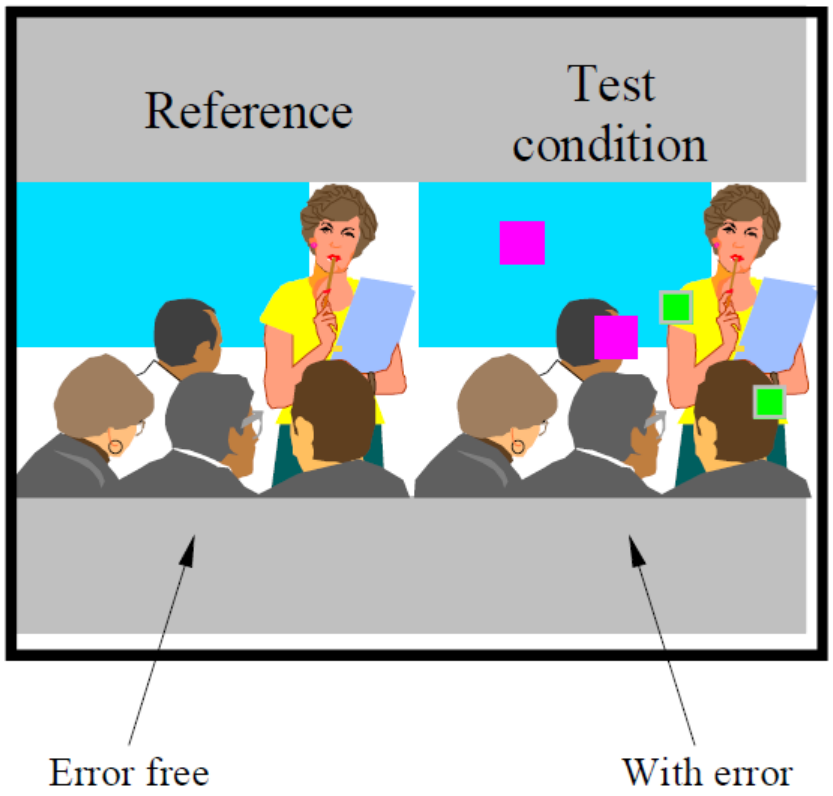

Figura 16: Sessão de apresentação SDSCE.

Fonte: (ITU, 2002)

As sequências podem ser apresentadas lado a lado no mesmo monitor ou em monitores diferentes, desde que tenham as mesmas especificações e características.

2.5 DISTRIBUIÇÕES DE PROBABILIDADE

Distribuição de probabilidade é um conceito fundamental em Estatística, utilizado em nível prático e teórico (LABORATORY, 2012).

As distribuições de probabilidade são funções matemáticas que mapeiam para todo valor dentro de um intervalo definido (discreto ou contínuo), valores de probabilidade de

ocorrência, considerando uma variável aleatória que pode assumir qualquer valor deste intervalo (LABORATORY, 2012). Em outras palavras, é a função que descreve a probabilidade de uma variável aleatória assumir determinados valores (WIKIPEDIA, 2012j).

Distribuições discretas são funções de probabilidade cuja variável pode assumir um número discreto de valores, não necessariamente finito. A definição matemática de uma função de probabilidade discreta, p(x), é uma função que satisfaz as seguintes propriedades:

- A probabilidade de x assumir um valor específico é p(x);
- p(x) é não negativa para todo x real;
- a soma de p(x) para todos os valores possíveis de x é igual a 1.

Distribuições contínuas ou funções de probabilidade contínuas são definidas para infinitos valores dentro de um intervalo. Por este fato, a probabilidade é medida para subintervalos, uma vez que a probabilidade de qualquer ponto é sempre zero (LABORATORY, 2012).

A definição matemática de uma função de probabilidade contínua, f(x), é uma função que satisfaz as seguintes propriedades:

- a probabilidade de x estar entre dois pontos a e b é: $p[a \leqslant x \leqslant b] = \int_a^b f(x)dx$;
- a probabilidade é sempre não-negativa para todo x real;
- a integral da função de probabilidade é: $\int_{-\infty}^{\infty} f(x)dx = 1$

Normalmente, distribuições de probabilidade são definidas por suas funções densidade de probabilidade, que associam a cada valor de x, sua probabilidade de ocorrência, de acordo com a equação a seguir:

Equação: $f(x) = Pr[X = x]$.

Particularmente para distribuições contínuas, a densidade de probabilidade é definida como a integral entre dois pontos, já que para qualquer ponto a probabilidade deve ser zero, como já descrito:

Equação: $\int_a^b f(x)dx = Pr[a \leqslant X \leqslant b]$.

Algumas distribuições de probabilidade bastante conhecidas, como por exemplo a distribuição normal e a uniforme, são detalhadas a seguir.

2.5.1 DISTRIBUIÇÃO UNIFORME

Distribuição uniforme contínua, ou distribuição retangular, é aquela em que todos os intervalos de mesmo tamanho tem a mesma probabilidade de ocorrer (WIKIPEDIA, 2012l). Mais precisamente, a probabilidade de se gerar qualquer ponto em um intervalo contindo no espaço amostral é proporcional ao tamanho do intervalo (WIKIPEDIA, 2012f). Em uma distribuição uniforme discreta, para todos os elementos possíveis a probabilidade é a mesma.

A equação da função densidade de probabilidade contínua dada a seguir, é exemplificada na Figura 17.

Equação:

$$f(x) = \begin{cases} \frac{1}{(b-a)} & \text{para } a \leqslant x \leqslant b, \\ 0 & \text{para } x < a \text{ ou } x > b \end{cases}$$

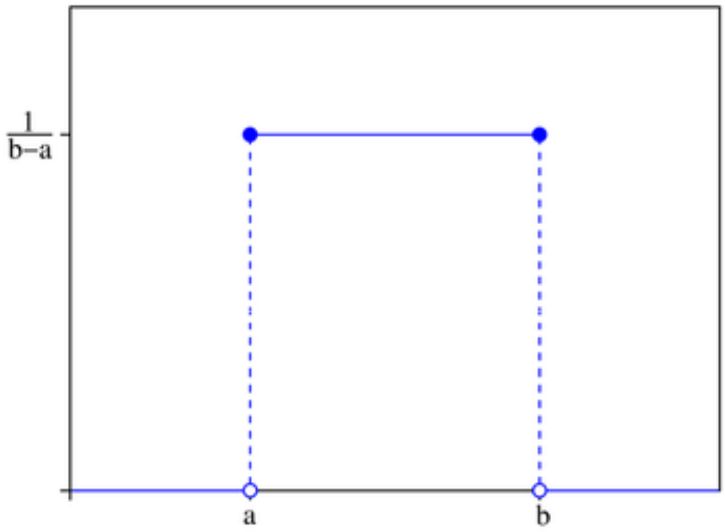

Figura 17: Função Densidade de Probabilidade de Distribuição Uniforme.
Fonte: (WIKIPEDIA, 2012l)

Uma distribuição uniforme que seja definida dentro do intervalo (0, 1) é chamada de Distribuição Uniforme Padrão (U(0, 1)). A partir dela é possível gerar distribuições em diferentes intervalos (a, b) conforme a equação a seguir:

Equação:

$$f(a,b) = a + [(b-a) \cdot U(0,1)]$$

Muitas linguagens de programação, assim como Java e C++, possuem funções nativas para gerar valores uniformemente distribuídos. Em Java, a classe Math possui a função ran-

dom() que gera valores de acordo com uma distribuição uniforme padrão. Ainda, Java possui uma classe especializada no pacote *util* chamada Random, utilizada para gerar valores booleanos, de *bytes*, de pontos flutuantes, etc. Esta classe também gera valores seguindo uma distribuição normal através do método nextGaussian(), que utiliza a transformada Box-Mueller, explicada na próxima seção (ORACLE, 2011).

Na linguagem C++ a função rand() encontra-se na biblioteca padrão (cstdlib ou stdlib.h), assim como a função srand() que inicializa a semente do gerador. O gerador, ao contrário de Java, fornece valores inteiros no intervalo (0, RAND_MAX). A constante RAND_MAX, também da biblioteca padrão, tem valor 32767 (REFERENCE, 2012).

2.5.2 DISTRIBUIÇÃO NORMAL

Conhecida também como Função Gaussiana, esta distribuição de probabilidade tem domínio nos reais no intervalo $(-\infty, \infty)$. É definida de acordo com dois parâmetros: a média (μ), ou valor esperado, e a variância (σ^2), conforme a seguinte equação:

Equação: $f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$.

onde:

- $f(x;\mu,\sigma^2)$ é a probabilidade de ocorrência do valor x;
- μ é o valor da média;
- σ e σ^2 são os valores desvio padrão e variância, respectivamente.

A Figura 18 apresenta diferentes funções densidade de probabilidade normal para diferentes parâmetros de média (μ) e variância (σ^2).

Na Teoria das Probabilidades, o Teorema do Limite Central afirma que, dada certas condições, a média de um número suficientemente grande de variáveis aleatórias independentes, cada uma com média e variância finita, seguirá uma distribuição normal (RICE, 1995).

Em teoria, uma variável aleatória seguindo uma distribuição normal pode assumir valores infinitos, o que é inconveniente computacionalmente. Desta forma, é bastante comum aplicar-se o truncamento desta distribuição, em que um limite inferior e/ou superior é utilizado para adequar os resultado a um intervalo desejado.

Existem algumas formas de se construir computacionalmente um gerador aleatório que segue uma distribuição normal. Dois métodos bastante conhecidos são: transformada de Box-Mueller e algoritmo de Ziggurat.

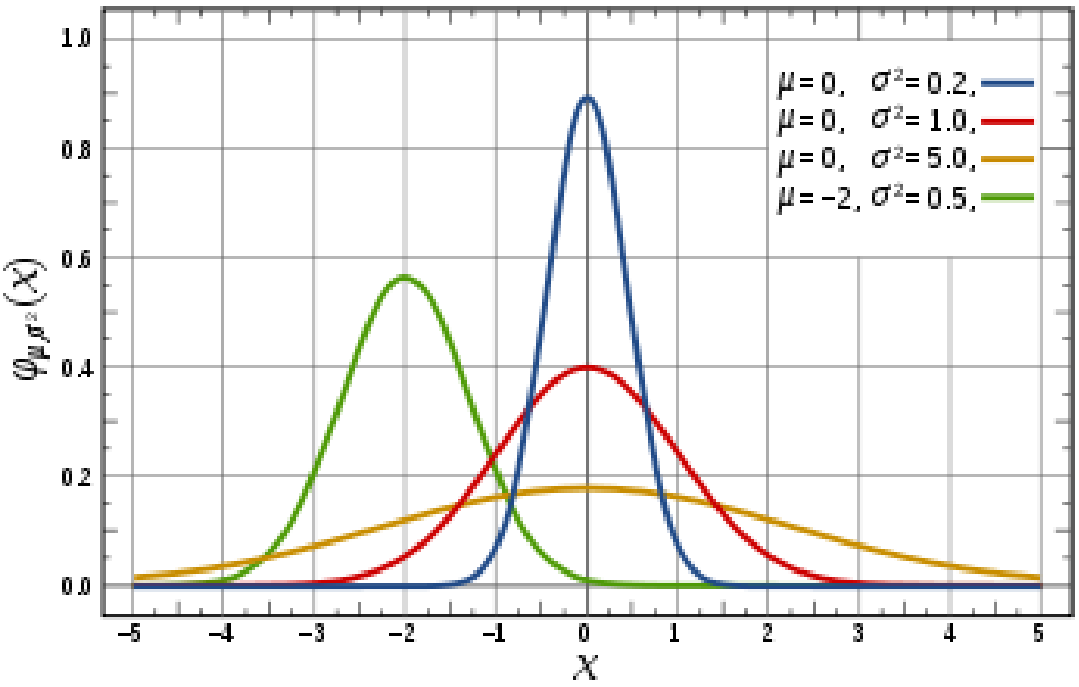

Figura 18: Função Densidade de Probabilidade de Distribuição Normal.

Fonte: (WIKIPEDIA, 2012i)

A transformada de Box-Mueller se baseia num gerador aleatório com distribuição uniforme que pode ser expresso em duas formas: a básica e a polar. Na primeira, o gerador uniforme deve gerar valores no intervalo (0, 1], na segunda forma devem ser gerados valores no intervalo [-1, +1] como valores de entrada (BOX; MUELLER, 1958).

Supondo que U_1 e $U2$ são variáveis aleatórias com distribuição uniforme no intervalo (0, 1] (ou seja, forma básica), as variáveis Z_0 e Z_1, dadas conforme as equações a seguir, seguirão uma distribuição normal com desvio padrão igual a 1.

Equações:

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

$$Z_1 = R\sin(\Theta) = \sqrt{-2\ln U_1}\sin(2\pi U_2)$$

A forma polar da transformada pode ser calculada conforme as equações a seguir. Nesta forma, a vantagem reside no fato de que as funções trigonométricas não precisam ser calculadas diretamente (WIKIPEDIA, 2012a). Considerando que u e v são variáveis uniformemente distribuídas, e $s = u^2 + v^2$, as variáveis z_0 e $z1$ seguirão uma distribuição normal:

$$z_0 = \sqrt{-2\ln U_1}\cos(2\pi U_2) = \sqrt{-2\ln s}(\frac{u}{\sqrt{s}})\cos(2\pi U_2)) = u\sqrt{\frac{-2\ln s}{s}}$$

$$z_1 = \sqrt{-2\ln U_1}\sin(2\pi U_2) = \sqrt{-2\ln s}(\frac{v}{\sqrt{s}})\cos(2\pi U_2)) = v\sqrt{\frac{-2\ln s}{s}}$$

As condições de validade são: garantir que s seja maior que zero e menor que um. Caso alguma destas condições seja violada, deve-se gerar novos valores de u e v até que as condições sejam satisfeitas.

O Algoritmo Ziggurat se baseia, assim como a transformada Box-Mueller em sua forma polar, no algoritmo de rejeição para gerar valores. Normalmente é utilizado quando a geração de uma grande quantidade de números é necessária (MARSAGLIA; TSANG, 2000).

2.5.3 DISTRIBUIÇÃO TRIANGULAR

Esta distribuição, como o nome diz, tem formato triangular. Sua função densidade de probabilidade varia de acordo com três parâmetros: a, b e c, como pode ser observado nas equações a seguir e na Figura 19.

$$\begin{cases} 0 & \text{para } x < a \\ \frac{2(x-a)}{(b-a)(c-a)} & \text{para } a \leqslant x \leqslant c, \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{para } c \leqslant x \leqslant b, \\ 0 & \text{para } b < x \end{cases}$$

Computacionalmente é possível criar um gerador aleatório que segue esta distribuição a partir de um gerador aleatório que siga uma distribuição uniforme no intervalo (0, 1) através das equações a seguir:

$$\begin{cases} X = a + \sqrt{U(b-a)(c-a)} & \text{para } 0 < U < F(c) \\ X = b - \sqrt{(1-U)(b-a)(b-c)} & \text{para } F(c) \leqslant U < 1 \end{cases}$$

2.6 TRABALHOS RELACIONADOS

Dada a escala e a complexidade do problema é natural que existam diversos estudos na área, e portanto, existam ferramentas para esse propósito. Um breve levantamento pode destacar duas ferramentas desenvolvidas no meio acadêmico que possuem finalidades similares

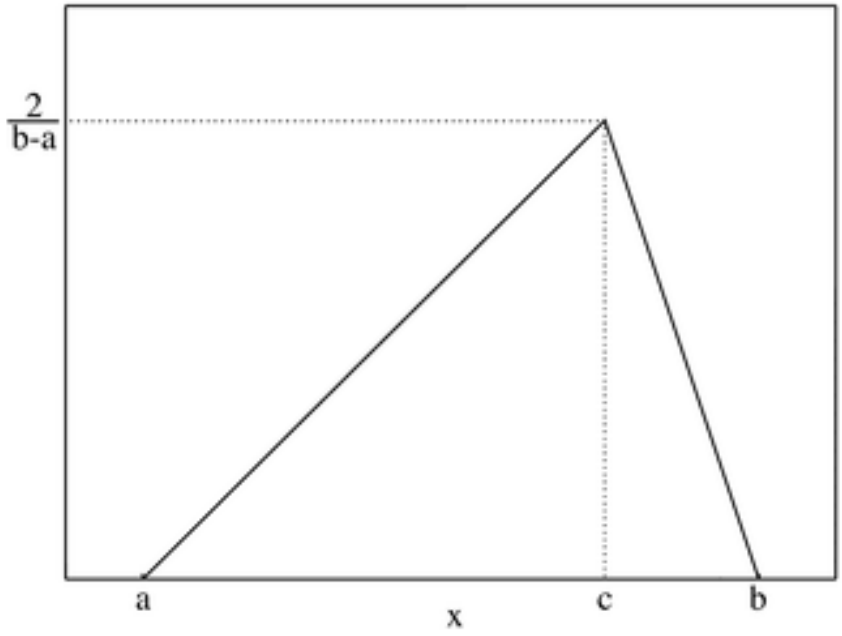

Figura 19: Função Densidade de Probabilidade de Distribuição Triangular.
Fonte: (WIKIPEDIA, 2012k)

à proposta neste projeto, além de uma no âmbito comercial.

A primeira delas é a MSU *Video Quality Measurement Tool*, desenvolvida pelo grupo do laboratório *Graphics and Media Lab* da *Moscow State University* (MSU) (UNIVERSITY, 2012). A ferramenta tem o propósito de avaliar a qualidade de vídeos se utilizando de métricas objetivas. O sistema implementa 20 métricas objetivas, suporta os mais populares formatos de vídeo disponíveis e oferece resultados sumarizados em formato CSV ou um vídeo de vizualização normalizada. Outra ferramenta é o EvalVid, desenvolvido pelo *Telecommunication Networks Group (TKN)* da *Techinical University of Berlin* (TKN, 2012). Esta se propõe a avaliar a qualidade de vídeos transmitidos por meio de uma rede de comunicação, seja ela simulada ou real, fornecendo parâmetros de qualidade de serviço assim como calculo do PSNR *frame* a *frame*.

O *Picture Quality Analysis System* PQA600 da Tektronix promete avaliações de vídeo digital por meio de métricas objetivas que correspondem fortemente à avaliação visual humana (TEKTRONIX, 2011). O PQA600 é comercializado na forma de uma *workstation* completa, possuindo um poderoso processador, dispositivo de exibição de vídeo e também *interfaces* de captura de vídeo digital e analógico.

2.7 RESUMO E CONCLUSÃO DO CAPÍTULO

Comparando-se as metodologias, ambas são importantes e ambas têm suas limitações. A avaliação objetiva é mais precisa considerando que ela avalie a diferença entre um vídeo original e o efetivamente observado. Porém, nem sempre o vídeo original possui uma qualidade que satisfaça o telespectador, sendo esta, portanto, uma de suas desvantagens. A desvantagem mais significativa é a dificuldade em se reproduzir a percepção humana.

A avaliação subjetiva embora seja um processo mais difícil de ser realizado por depender de um grande grupo de pessoas e tenha um custo mais elevado (ALBINI, 2009), segundo (WANG et al., 2004) esta é o tipo de avaliação mais 'correta', pois o próprio telespectador realiza a avaliação.

3 ESPECIFICAÇÃO

O *software* a ser desenvolvido neste trabalho deverá fornecer uma ferramenta flexível de degradação de vídeo digital e também de avaliação subjetiva e objetiva das mídias. Neste capítulo serão apresentados os requisitos de tal sistema, assim como as especificações de desenvolvimento, arquitetura e funcionamento.

3.1 ANÁLISE DE REQUISITOS

A análise de requisitos é parte fundamental de qualquer projeto, buscando atender da melhor forma as necessidades dos futuros usuários. Esta seção descreve os requisitos levantados durante o estudo e planejamento da ferramenta.

3.1.1 REQUISITOS FUNCIONAIS

Requisitos funcionais são aqueles determinados pelas funcionalidades exigidas do sistema, ou seja, as ações que se espera que o sistema execute. Foram levantados os seguintes requisitos funcionais:

- O *software* deverá possuir ferramentas de degradação de vídeos digitais de blocagem e borramento.
- O *software* deverá possuir uma ferramenta de simulação de transmissão *streaming*.
- O *software* deverá ser capaz de criar e gerenciar sessões de avaliação subjetiva.
- O *software* deverá possuir uma ferramenta que realize avaliação objetiva sobre os vídeos da base de dados, conforme as métricas PSNR, MSE e MSSIM.
- O *software* deverá ser capaz de exibir os vídeos existentes na base de dados.
- O *software* deverá exibir os resultados das avaliações objetivas e subjetivas de forma gráfica.

3.1.2 REQUISITOS NÃO-FUNCIONAIS

Requisitos não-funcionais são aqueles ditados por restrições ou exigências de qualidade ou de operação, tais como performance, segurança ou tecnologias envolvidas. Abaixo se encontram os requisitos não funcionais levantados:

- As ferramentas de degradação e métricas objetivas deverão operar sobre vídeos em formato YUV planificado com subamostragem 4:2:0.
- A ferramenta de simulação deverá operar sobre vídeos no formato H.262 encapsulados em um TS.
- O sistema deve fornecer documentação detalhada de auxílio ao uso das ferramentas.
- A ferramenta de exibição deverá ser executada em um sistema capaz de prover um *display* com resolução igual ou maior que a dos vídeos a serem exibidos.
- A *Interface* gráfica do sistema necessita que haja um *Java Runtime Environment* (JRE) instalado no sistema operacional.
- O *player* de vídeo MPlayer, livre e de código aberto, é necessário para mostrar os vídeos durante as sessões.
- O projeto deverá ser desenvolvido utilizando ferramentas de distribuição livre.
- O *software* deve ser capaz de operar independentemente do SGBD (Sistema Gerenciador de Banco de Dados) utilizado.
- O sistema deve ter um tempo de resposta adequado, de forma que este não interfira no *timing* das avaliações subjetivas.
- As ferramentas desenvolvidas devem estar disponíveis independentemente do sistema (*stand-alone*).

3.1.3 DISPOSITIVO DE AVALIAÇÃO SEM FIO

No projeto SASQV original, foi desenvolvido um dispositivo sem fio para que os observadores pudessem realizar avaliações ao longo de sessões subjetivas. O projeto proposto não pretende realizar modificações de *firmware* e *hardware*, apenas utilizar as funcionalidades já implementadas.

O dispositivo suporta sessões de avaliação seguindo as métricas DSIS e SDSCE, as demais estão em modo teste. A Figura 20 apresenta o dispositivo desenvolvido.

Figura 20: Dispositivo sem fio para avaliações subjetivas.
Fonte: (SANTOS et al., 2010)

3.2 ESPECIFICAÇÕES DO SOFTWARE

Esta seção tem o propósito de descrever a arquitetura proposta para o SASQV2, assim como abordar as diferentes linguagens e bibliotecas que ajudam a integrar o programa.

3.2.1 ARQUITETURA DO SISTEMA

Por se tratar da continuação do desenvolvimento do projeto SASQV, este trabalho reutiliza e adapta grande parte dos componentes existentes no trabalho original.

A Figura 21 mostra uma comparação entre as visões gerais de ambos os projetos. A diferença mais notável entre estas arquiteturas é a separação dos componentes de serviço e o de ferramentas, além de também recriar o componente de *Interface* gráfica para melhor atender às necessidades do sistema.

Mais detalhes sobre cada componente da arquitetura são descritos em seções específicas a seguir.

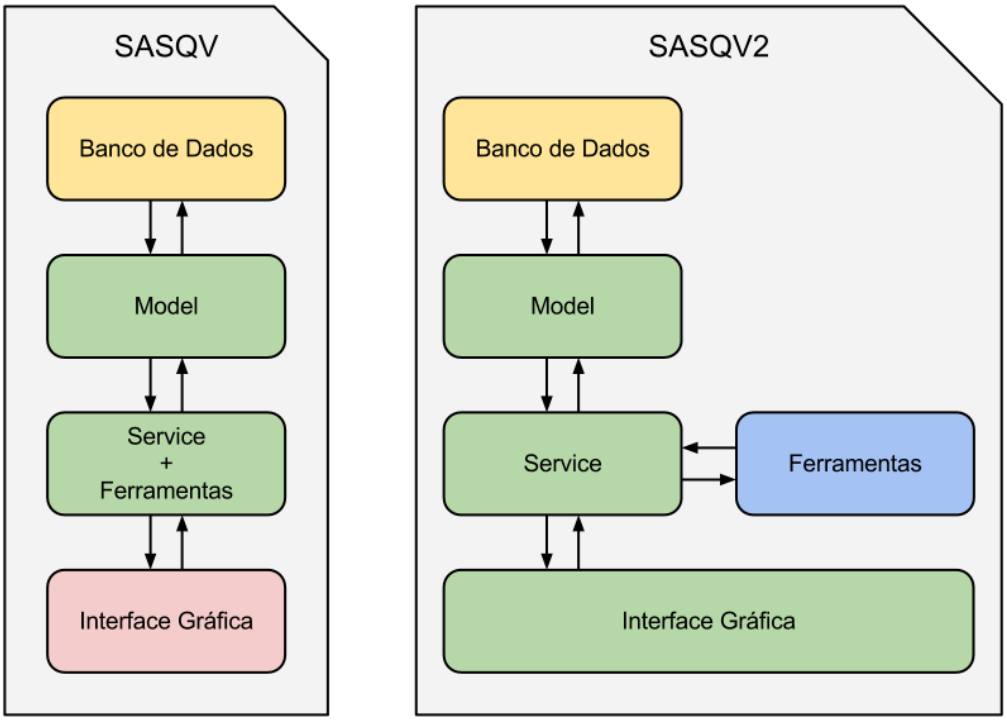

Figura 21: Visão geral das arquiteturas dos sistemas SASQV e SASQV2.
Fonte: Autoria Própria.

3.2.1.1 BANCO DE DADOS

O componente banco de dados foi completamente reaproveitado, seguindo as mesma especificações do SASQV.

O SGBD escolhido foi o MySQL, atualmente o SGDB *open source* mais utilizado no mundo (ORACLE, 2012). O MySQL foi lançado e desenvolvido em 1995 pela empresa Sueca MySQL AB, atualmente incorporada pela Oracle Corporation, na forma de um SGDB que fornece um servidor multi-usuários para bancos de dados relacionais (WIKIPEDIA, 2012h).

Juntamente do MySQL também é utilizado o MySQL Workbench, uma ferramenta distribuída pela Oracle que concilia um cliente MySQL, uma ferramenta de *Database Modde-ling* e um administrador de servidor MySQL. A versão *open source* de ambas as ferramentas é distribuída sob a licensa GNU General Public License (GPL).

O modelo Entidade-Relacionamento (ER) é o mesmo utilizado pelo SASQV, como mostrado na Figura 22.

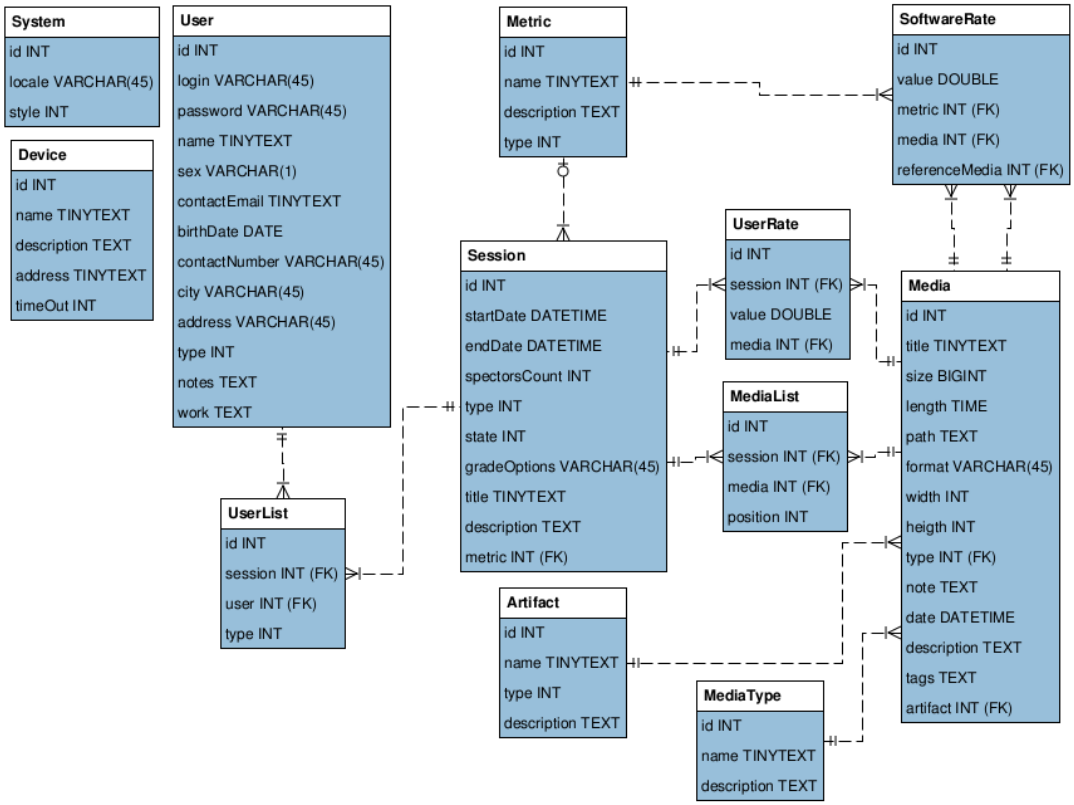

Figura 22: Diagrama Entidade-Relacionamento do SASQV2

Fonte: Autoria Própria

3.2.1.2 MODEL

O componente Model presente no SASQV2 também foi inteiramente reaproveitado do SASQV, cabendo apenas algumas adições. Este componente consiste na implementação de um mapeamento objeto-relacional (MOR) por meio da biblioteca Hibernate.

O Hibernate teve seu desenvolvimento iniciado em 2001 por Gavin King, com o objetivo de melhorar as ferramentas de persistência existentes na plataforma Java (COMMUNITY, 2012b), e é atualmente distribuído sob a licensa GNU Lesser General Public License (LGPL) (COMMUNITY, 2012a).

A biblioteca provê um mapeamento flexível entre objetos Java e tipos SQL, eliminando a custosa necessidade de processamento manual e repetitivo de listas de resultados obtidas via SQL e JDBC. Estima-se que até cerca de 30% do código de uma aplicação possa ser poupado utilizando MOR, além de incrementar a portabilidade do sistema suportando diversas implementações de Bancos de Dados SQL em troca de um pequeno *overhead* de performance.

3.2.1.3 SERVICE

Este elemento também foi reaproveitado, no entanto sofreu diversas adaptações em sua estrutura para comportar novas funcionalidades e continuar suportando as antigas. O componente presente no SASQV se trata de uma fachada de serviço responsável por interpretar mensagens provenientes da *Interface* e distribuí-las em uma das três formas a seguir:

1. Executar uma ação de degradação ou métrica objetiva.
2. Repassar uma ação para o dispositivo sem-fio e esperar por resposta.
3. Repassar uma ação para o elemento model e esperar por resposta.

Na arquitetura do SASQV2 a interpretação de mensagens foi descartada, uma vez que foi prosposta a implementação de uma nova *Interface* em linguagem Java, que deu lugar a métodos de finalidades distintas dentro de cada ação enumerada acima. Com isso a execução de tarefas de degradação ou avaliação por métrica objetiva, antes executadas localmente, passaram a ser delegadas para um novo componente chamado Ferramentas, descrito em 3.2.1.4.

Já o conteúdo dos métodos que repassam as ações para o dispositivo sem-fio e para o model permanecem muito semelhantes aos encontrados na interpretação de mensagens originalmente feito no SASQV, sofrendo apenas pequenas modificações para acomodar a nova estrutura.

3.2.1.4 FERRAMENTAS

O módulo ferramentas constitui um novo componente na arquitetura do *software*, visto que um dos objetivos deste trabalho é desenvolver melhores ferramentas de degradação considerou-se válida a separação deste módulo que originalmente se encontrava juntamente do Service.

Esse módulo é responsável por todas as ferramentas que de alguma forma manipulam vídeos digitais ou auxiliam nesta manipulação. Dessa forma foram desenvolvidas 5 ferramentas distintas, em linguagem C++:

Block Cria o efeito de blocagem nos vídeos.

Blur Cria o efeito de embaçamento nos vídeos.

Netsim Efetua uma simulação de streaming de vídeo digital com perda informação.

Raffle Ferramenta auxiliar para distribuição temporal e espacial dos artefatos.

Metric Efetua avaliação dos vídeos por meio de métricas objetivas.

Estas ferramentas podem receber configurações independentes e flexíveis, diversificando as possibilidades de geração de degradações e artefatos e tornando possível a criação de resultados diferenciados para um mesmo vídeo usado como referência. Dentre essas configurações pode-se destacar a distribuição dos artefatos no tempo e no espaço e a intensidade de aplicação dos mesmos. Cada ferramenta citada será descrita com mais detalhes em 4.2. A independência das ferramentas permite a utilização combinada em sequência, de forma que a cada utilização é possível aplicar artefatos à videos originais ou já degradados, criando inúmeras combinações de degradação. A figura 23 demonstra o fluxo de utilização das ferramentas em conjunto com uma base de vídeos.

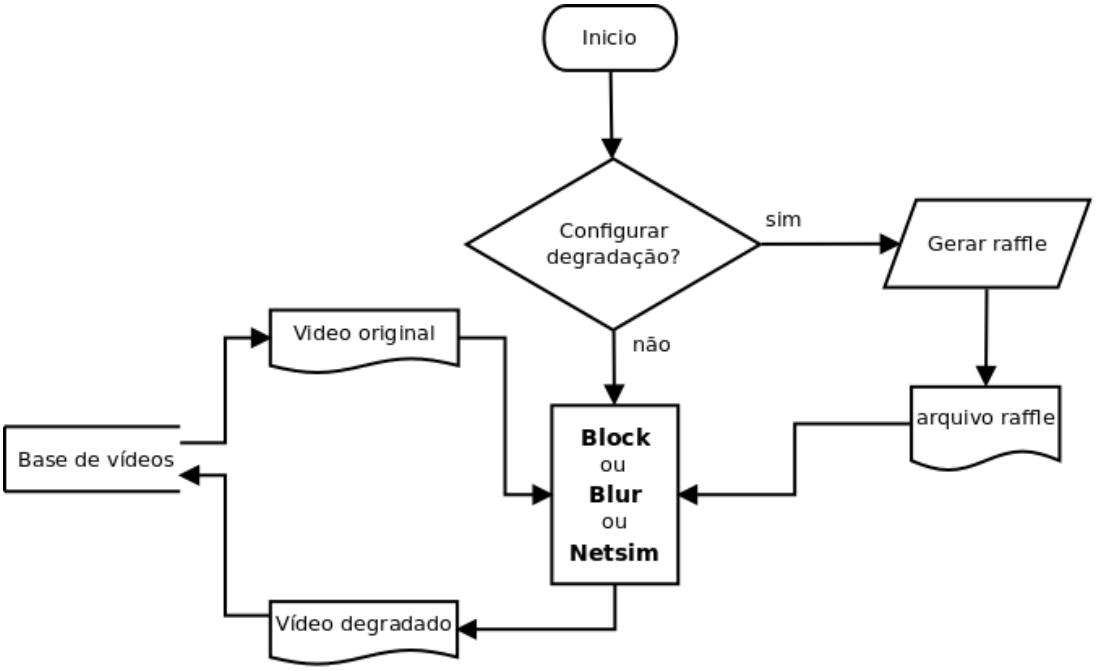

Figura 23: Fluxograma de utilização das ferramentas.

Fonte: Autoria Própria

3.2.1.5 INTERFACE GRÁFICA

Originalmente desenvolvida utilizando a tecnologia Adobe Flex, era necessária a utilização de uma biblioteca exclusivamente para comunicação entre os módulos de Interface e Service. A biblioteca utilizada, denominada merapi, se encontrava em fase de desenvolvi-

mento beta e carecia de funcionalidades, de forma que seria necessário buscar uma biblioteca alternativa para continuar o densevolvimento do projeto.

Além disso, as ferramentas de desenvolvimento em tecnologia Flex não são gratuitas e perderam gradativamente o suporte à sistemas operacionais Linux.

A combinação destes fatores motivou a decisão de desenvolver uma nova *interface* Gráfica utilizando Java, proporcionando melhor integração, reduzindo o envolvimento de diversas tecnologias e bibliotecas e assim aumentando a portabilidade da ferramenta.

3.2.2 LINGUAGENS

Por se tratar da aprimoração e continuação do projeto SASQV a primeira linguagem contemplada foi a presente no mesmo: Java.

Esta foi desenvolvida por uma equipe de programadores liderada por James Gosling, trabalhando para a empresa Sun MicroSystems (WIKIPEDIA, 2012g). Sendo lançada em 1995, era o principal componente da plataforma Java e demonstrava forte inspiração na linguagem C++, compartilhando parte de sua sintaxe e também diversas característica.

No entanto, Java é considerada uma linguagem de mais alto nível, além de ser compilada no formato *bytecode*, o que favorece a portabilidade dos programas, podendo ser executados em diversas arquiteturas e sistemas operacionais, contanto que estes possuam uma implementação da Java Virtual Machine (JVM). Uma das facilidades da linguagem é a sua extensa biblioteca padrão, que contém, entre outros, diversos componentes para construção de *interfaces* gráficas.

A linguagem C++ também passou a ser considerada para o desenvolvimento do projeto, visto o desenvolvimento do módulo de Ferramentas como programas que possam ser usados independentemente. A linguagem teve seu desenvolvimento iniciado por Bjarne Stroustrup em 1979, quando trabalhava na Bell Labs (WIKIPEDIA, 2012b). C++ é uma nova implementação da linguagem C adicionada de novas funcionalidades, tais como suporte a classes, sobrecarga de operadores, herança múltipla e tratamento de exceções. Atualmente, C++ é uma das linguagens mais populares, sendo implementada em diversas plataformas de *hardware* e sistemas operacionais, além de ser usada na implemantação dos mais diversos *softwares* para *desktop*, *drivers* de dispositivos de *hardware* e sistemas embarcados.

Um dos principais atrativos da linguagem para o desenvolvimento das ferramentas deste projeto é a existência de diversas funcionalidades de baixo nível. Essas funcionalidades simplificam e facilitam o acesso e manipulação *byte* a *byte* de *streams* de dados, algo que é usado

extensivamente ao longo da implementação das ferramentas. Outro aspecto importante derivado destas funcionalidades é o impacto na performance do software. Segundo (THE. . . , 2012), para o *benchmark reverse-complement* — onde são efetuadas diversas operações de leitura e escrita em arquivos, além de um leve processamento — a melhor implementação em Java foi 1.9 vezes mais lenta que a melhor implementação em C++. A simplicidade da linguagem também facilita o desenvolvimento de ferramentas *stand-alone* com *Interface* por linhas de comando, além de possuir diversas bibliotecas de *parsing* para comandos e opções.

Ambas as linguagens eram previamente conhecidas pelos integrantes do projeto, os quais também possuem um nível confortável de experiência no desenvolvimento de aplicações com estas linguagems e com diversas ferramentas associadas a elas, como IDE's, bibliotecas gráficas e conectividade com bancos de dados.

3.2.3 DIAGRAMAS DE CASO DE USO

Uma das primeiras fases em um projeto de *software* é a construção de diagramas de Casos de Uso. Estes diagramas auxiliam no processo de levantamento de requisitos e de entidades que deverão interagir com o sistema. Também auxiliam na definição e visualização dos serviços a serem desempenhados pelo sistema.

Os casos de uso elaborados foram divididos em diferentes diagramas, dando enfoque as funcionalidades mais importantes a serem desempenhadas pelo *software*: degradar vídeos, realizar avaliação objetiva e realizar avaliação subjetiva através da criação de uma sessão de vídeos.

O diagrama apresentado na Figura 24 mostra o caso de uso geral. Para que o usuário possa interagir com o sistema é necessário realizar *login* no sistema utilizando usuário e senha válidos. Esta validação é realizada pelo SGBD, representado no diagrama pelo ator de mesmo nome.

O sistema SASQV original previa a existência de dois tipos de ator: administrador e avaliador. O primeiro possuía permissões para acessar o sistema e realizar todas as operações, menos realizar avaliações de vídeo. O segundo, ao contrário do primeiro, só poderia realizar avaliações. Em prol da simplicidade, optou-se por permitir que avaliadores também pudessem realizar as mesmas ações que o administrador, sendo necessário apenas um destes atores.

Concluído o *login*, o usuário pode realizar atividades como criar uma sessão para avaliação subjetiva; degradar vídeos; adicionar ou remover vídeos da base de dados; adicionar, remover ou editar cadastro de usuários; consultar resultados através de gráficos; consultar

tópicos de ajuda; realizar avaliações objetivas utilizando as ferramentas desejadas e, por fim, modificar configurações gerais do sistema.

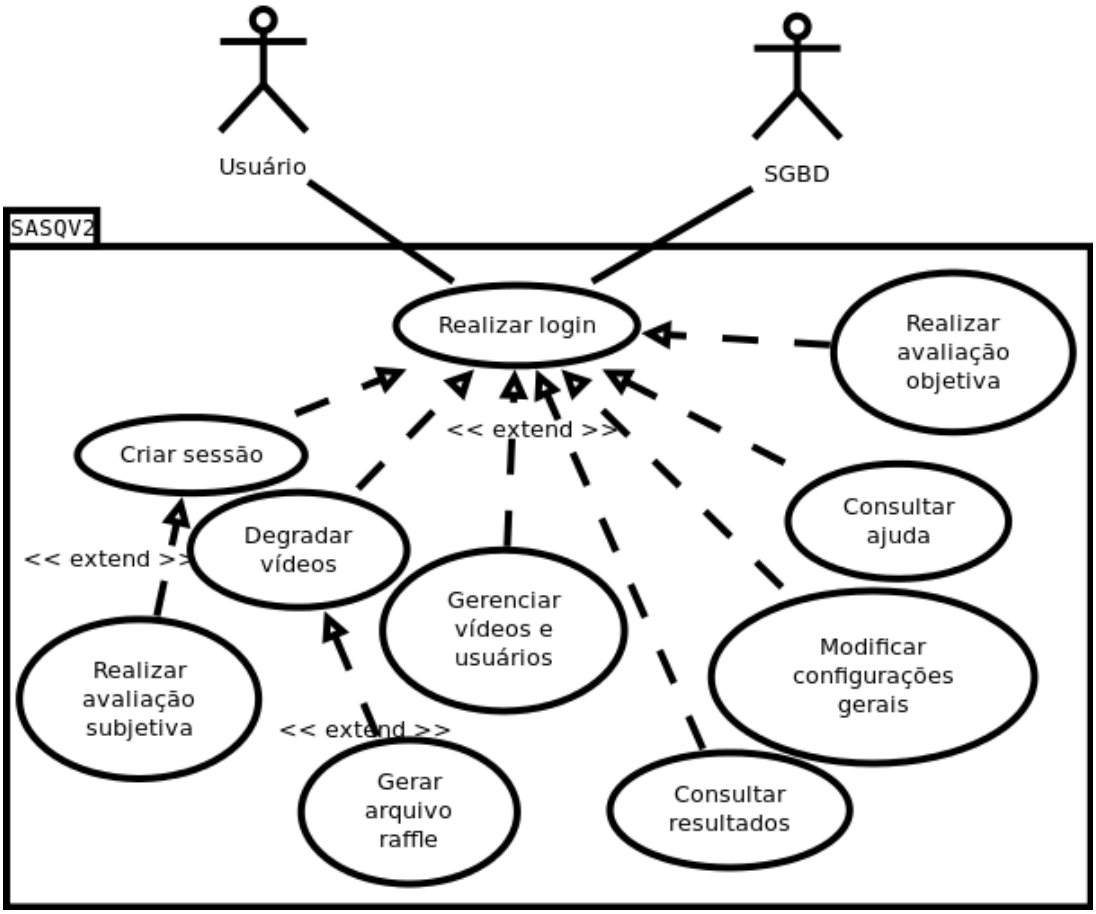

Figura 24: Diagrama de Caso de Uso geral.

Fonte: Autoria Própria

Uma das funcionalidades mais importantes do sistema proposto é a realização de sessões para avaliação subjetiva, apresentados na Figura 25. Para criar uma sessão, primeiramente é necessário definir a identificação assim como os parâmetros de configuração. O próximo passo é a escolha dos vídeos da base de dados que serão apresentados. Por fim, são selecionados os equipamentos que serão utilizados. Todos os passos interagem com o SGBD, seja realizando consultas, seja atualizando-o.

O próximo diagrama, na Figura 26, apresenta duas funcionalidades do sistema: degradar vídeos e realizar avaliação objetiva. A degradação de vídeos depende da seleção dos vídeos desejados e da seleção da degradação a ser aplicada, conforme os parâmetros informados. Para realizar uma avaliação objetiva, basta selecionar os vídeos e métricas desejadas.

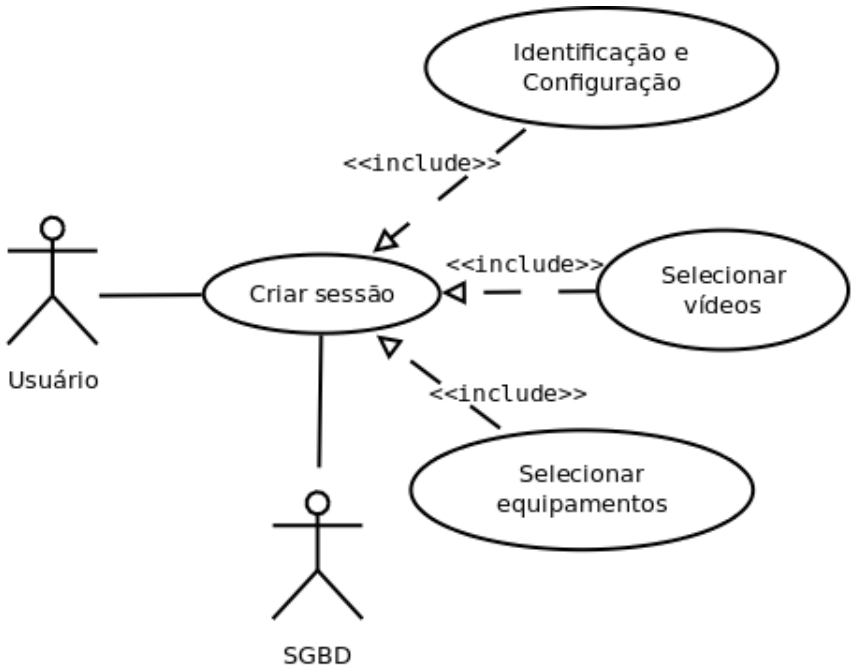

Figura 25: Diagrama de Caso de Uso de sessão.

Fonte: Autoria Própria

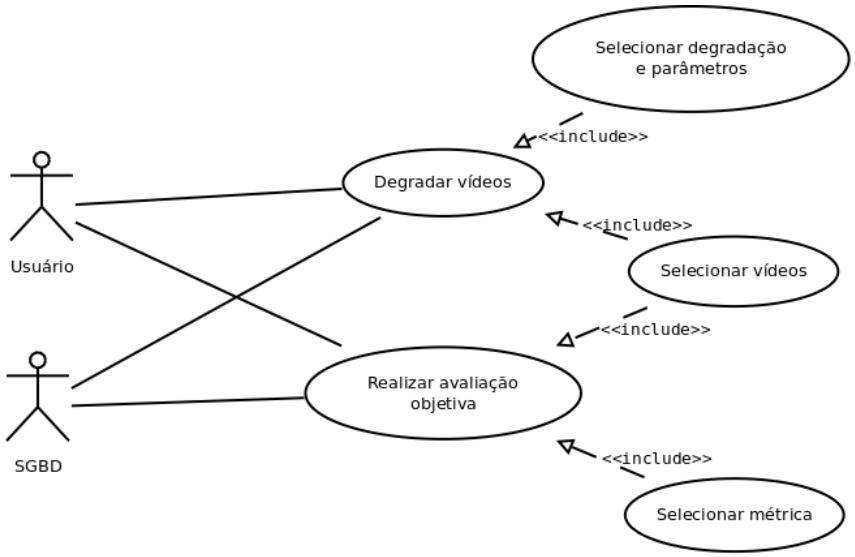

Figura 26: Diagrama de Caso de Uso de degradação.

Fonte: Autoria Própria

3.3 CONSIDERAÇÕES

Para este projeto de *software* procurou-se aproveitar ao máximo os componentes existentes no SASQV, de forma a maximizar o esforço nos aspectos a serem melhorados e não

consumir tempo de projeto retrabalando o que já foi feito.

A adesão ao desenho de *software* apresentado se justifica pela independência entre os componentes, facilitando o desenvolvimento e teste de cada um e também o reaproveitamento em trabalhos futuros. A opção por criar ferramentas independentes para o processo de degradação se mostra eficiente e versátil, visto que deste modo estão prontas para serem reutilizadas e combinadas à outras ferramentas, além de facilitarem a automatização do processo de degradação para o usuário avançado.

Sendo assim, a estrutura planejada foi muito semelhante à do projeto original, com diversos componentes sofrendo adaptações e também a adição de um novo componente dedicado.

4 DESENVOLVIMENTO

Este capítulo descreve como efetivamente foi realizado o projeto, forma de trabalho, ferramentas utilizadas, resultados obtidos (ferramentas e *interfaces*), ou seja, o reflexo das decisões de projeto tomadas que foram abordadas no capítulo anterior.

4.1 FERRAMENTAS UTILIZADAS

4.1.1 SISTEMA OPERACIONAL

O desenvolvimento deste projeto foi totalmente realizado em ambiente Linux, em diferentes distribuições derivadas do Debian: Ubuntu 12.04 e Xubuntu 11.10.

4.1.2 AMBIENTES DE DESENVOLVIMENTO INTEGRADO

A primeira etapa de desenvolvimento planejada foi a de construir um modelo para a nova *interface* gráfica. Tendo em vista a linguagem Java como requisito, optou-se pelo IDE NetBeans (versão 7.1.1) que possui uma ferramenta nativa específica para a construção de *interfaces* gráficas para usuário, a GUI Builder (*Graphical User Interface*). Esta ferramenta permite a construção de formulários no estilo drag-and-drop de containers existentes no Java, como por exemplo JFrame, JPanel, JButton, etc.

Inicialmente, o IDE Eclipse também foi utilizado com a finalidade de se realizar modificações nos projetos originais model e service. Posteriormente, para unificar o projeto em apenas um ambiente de desenvolvimento, os projetos originais foram importados no NetBeans.

4.1.3 OUTRAS FERRAMENTAS

Os códigos em C++ foram desenvolvidos através do editor de textos Vim, um editor de textos gratuito e de código aberto amplamente configurável para permitir edições eficientes

de texto (VIM, 2012). Este editor é uma versão aperfeiçoada do editor Vi, presente na maioria dos sistemas UNIX.

Para criar executáveis dos códigos C++ gerados, foi utilizado o compilador g++. Este compilador faz parte da coleção de compiladores GCC (PROJECT, 2012a) produzida e mantida pelo GNU *Project* (PROJECT, 2012b).

A fim de automatizar o processo de geração de executáveis, a ferramenta *make* foi configurada de acordo. Esta ferramenta, também mantida pelo GNU Project, permite que vários códigos-fonte sejam processados ao mesmo tempo através de um arquivo de configuração chamado *makefile*, que pode executar um compilador para gerar arquivos-objeto, ou executar um *linker* para gerar executáveis (PROJECT, 2012c).

4.1.4 CONTROLE DE VERSÕES

Em função de a equipe contar com dois integrantes desenvolvedores, existe a necessidade de se utilizar uma ferramenta para o controle de versões do projeto. Primeiramente, para realizar testes de algumas funcionalidades foi preciso modificar radicalmente o código o que, em certos casos, não gerou resultados satisfatórios, sendo necessário retornar à um ponto estável e funcional. Além deste motivo, o sistema de controle de versões escolhido, o Git, possui ferramentas de resolução de conflitos eficientes que auxiliam o desenvolvimento simultâneo.

O Git é um software livre e gratuito distribuído sob a licença GNU General Public License versão 2 (CONSERVANCY, 2012).

Aliado a esta ferramenta, foi utilizado um serviço web de hospedagem de projetos que é organizado pelo sistema de controle de versões Git, o GitHub cujo endereço é http://github.com. Existem funcionalidades no estilo rede social como feeds, seguidores e gráficos diversos, bem como funcionalidades de projeto como visualização de pastas e códigos, gráficos de desempenho por usuário, por equipe, por período de desenvolvimento, frequência de código, histórico de modificações, entre outros (GITHUB, 2012).

Na versão gratuita, que foi a utilizada, há uma exigência: que o código seja aberto (GITHUB, 2012). Na versão paga, existem planos que permitem a criação de repositórios privados com times de desenvolvimento. Para ambas as versões o número de colaboradores é ilimitado, assim como o número de repositórios públicos.

O repositório do projeto SASQV2 está hospedado no endereço https://github.com/brunnobga/TCC-Code e pode ser baixado livremente.

4.2 FERRAMENTAS DESENVOLVIDAS

As ferramentas foram planejadas para uma utilização modular, ou seja, para serem utilizadas através da *interface* do SASQV assim como através da linha de comando, de forma independente. Um dos requisitos destas ferramentas é a de que sejam aplicadas sobre vídeos no formato YUV com subamostragem 4:2:0 *progressive*.

4.2.1 RAFFLE

A ferramenta *raffle* tem como objetivo gerar elementos aleatórios multi-dimensionais no domínio dos naturais, armazenado-os em arquivo.

No arquivo gerado, cada coluna representa uma dimensão que deve seguir alguma distribuição de probabilidade, informada via parâmetros. Dentre as distribuições possíveis e seus respectivos parâmetros tem-se:

- Distribuição constante: a dimensão correspondente terá valor constante c para todos os elementos gerados.
- Distribuição uniforme: a dimensão correspondente terá valores uniformemente gerados num intervalo [a, b).
- Distribuição normal: a dimensão correspondente terá valores gerados conforme uma distribuição normal, de média m e desvio padrão d.
- Distribuição triangular: a dimensão correspondente terá valores gerados conforme uma distribuição triangular no intervalo [a, b) com pico em c.

A Tabela 6 sintetiza os parâmetros que devem ser informados conforme o tipo de distribuição desejada.

Tabela 6: Distribuições e respectivos parâmetros para a execução da ferramenta raffle.

Distribuição	Parâmetros			
Constante	-d ou -u	constant	-p ou -r	a
Uniforme	-d ou -u	uniform	-p ou -r	a,b
Normal	-d ou -u	normal	-p ou -r	m,d
Triangular	-d ou -u	triangular	-p ou -r	a,b,c

Adicionalmente aos parâmetros de cada distribuição, a primeira coluna possui uma característica temporal que pode ser ativada, permitindo a repetição de determinado elemento

ao longo de um intervalo constante ou aleatório. Para este sorteio é possível utilizar qualquer distribuição descrita acima.

O comando para utilizar a ferramenta e seus parâmetros são descritos a seguir:

./raffle

`--output`	`-o`	arquivo_de_saída
`--durationdist`	`-u`	tipo de distribuição da duração
`--durationparams`	`-r`	parâmetros da distribuição da duração
`--elements`	`-e`	elementos a serem gerados
`--dist`	`-d`	tipo de distribuição da primeira dimensão
`--params`	`-p`	parâmetros da distribuição da primeira dimensão
...		
`--dist`	`-d`	tipo de distribuição da n-ésima dimensão
`--params`	`-p`	parâmetros da distribuição da n-ésima dimensão
`--help`	`-h`	menu de ajuda

Observações:

- Todo parâmetro -d deve ser imediatamente sucedido por um parâmetro -p
- O parâmetro -u deve ser imediatamente sucedido por um parâmetro -r
- Cada conjunto (-d -p) representa uma dimensão, ou uma coluna, a ser gerada.
- Cabe ao usuário conhecer a ferramenta para adequá-la corretamente as suas necessidades.

Exemplos de utilização através da linha de comando são descritos abaixo. A diferença para a *interface* gráfica reside apenas no fornecimento dos dados, uma vez que a *interface* deve executar o mesmo comando para gerar o arquivo.

No comando a seguir, será gerado um arquivo com nome raffleout.rff contendo 30 elementos. Para este exemplo, será utilizado um valor de duração constante e igual a 1, no próximo exemplo este conceito de duração será melhor detalhado. A primeira dimensão deve seguir uma distribuição uniforme dentro do intervalo [1, 30), a segunda dimensão deve ter valor constante 3 para todos os elementos gerados e a terceira e última dimensão deve seguir uma distribuição triangular no intervalo [3, 20) com pico no ponto 10.

./raffle -o raffleout.rff -u constant -r 1 -e 30 -d uniform -p 1,30 -d constant -p 3 -d triangular -p 3,20,10

O resultado de uma execução deste comando pode ser observado na Tabela 7, a seguir:

Tabela 7: Resultado de execução do commando *raffle* para exemplo 1.

6	3	7	16	3	12	18	3	7	23	3	14	9	3	9	15	3	7
9	3	11	28	3	10	21	3	8	1	3	7	12	3	11	18	3	14
11	3	10	25	3	11	17	3	14	20	3	14	28	3	8	13	3	11
10	3	8	10	3	12	15	3	14	22	3	14	10	3	12	6	3	4
27	3	8	5	3	9	23	3	13	3	3	7	6	3	17	20	3	13

No comando abaixo, será gerado um arquivo de nome raffleout2.rff com 50 elementos de duas dimensões: a primeira seguirá uma distribuição uniforme dentro do intervalo [1,5) e a segunda seguirá uma distribuição normal com média 2 e desvio padrão 1:

./raffle -o raffleout2.rff -u constant -r 3 -e 50 -d uniform -p 1,30 -d normal -p 10,2

A duração, neste exemplo, tem duração constante 3, ou seja, para todo elemento gerado, ele deverá ter duração de 3 elementos no total. O efeito da duração pode ser observado na figura, onde o primeiro elemento (7 10) tem duração 3 tendo a primeira dimensão como referência temporal: (7 10), (8 10) e (9 10). A geração de elementos prossegue até que o número de elementos gerados atinja o desejado, informado no parâmetro -e. Um resultado importante, que pode ser observado na Tabela 8, é que a duração não ultrapassa os limites definidos para a primeira coluna, [1, 30), fazendo com que nem todos os artefatos tenham a duração estabelecida.

Tabela 8: Resultado de execução do commando *raffle* para exemplo 2.

7	10	28	10	17	10	19	8	16	9	20	11	29	8	14	9	21	7	1	8
8	10	19	7	18	10	20	8	4	8	21	11	12	10	15	9	22	7	2	8
9	10	20	7	11	8	21	8	5	8	27	9	13	10	3	12	23	7	3	8
26	10	21	7	12	8	14	9	6	8	28	9	14	10	4	12	28	9	8	8
27	10	16	10	13	8	15	9	19	11	29	9	13	9	5	12	29	9	9	8

No planejamento inicial, as ferramentas de blocagem, borramento e simulador deveriam receber parâmetros para geração aleatória de artefatos. Posteriormente, notou-se que a modularização desta funcionalidade seria possível e poderia trazer algumas vantagens:

- com uma ferramenta específica para a geração de pontos de inserção de artefatos, seria possível armazenar os resultados em forma de arquivo de texto, ocupando muito menos espaço que um vídeo degradado;
- a reprodução de determinada degradação poderia ser realizada. Na ideia inicial, apesar do fornecimento de parâmetros iguais, seria improvável gerar dois vídeos com degradações iguais;

- o arquivo de degradação pode ser aplicado em outros vídeos, a fim de testar o impacto de uma mesma degradação em vídeos com possíveis cenas, objetos, movimentos diferente.

4.2.2 BLOCK

Esta ferramenta é responsável por aplicar o efeito de blocagem sobre o vídeo desejado. Os parâmetros que devem ser fornecidos são descritos a seguir:

./block

`--input`	`-i`	arquivo_de_entrada
`--output`	`-o`	arquivo_de_saída
`--size`	`-s`	dimensões do vídeo de entrada no formato WxH (largura por altura em *pixels*)
`--window`	`-w`	tamanho da janela da DCT
`--levelsdct`	`-l`	níveis DCT à serem eliminados, separados por vírgulas
`--rafflelist`	`-r`	arquivo_raffle
`--help`	`-h`	menu de ajuda

A partir do vídeo desejado, deve-se escolher o nome do vídeo resultante e informar as dimensões do vídeo original, o tamanho da janela a ser utilizada na DCT, os subníveis de energia que deverão ser eliminados (zerados) e o nome do arquivo *raffle* a ser utilizado.

O arquivo *raffle* de entrada deve sempre conter três colunas. A primeira diz respeito ao *frame* em que será aplicado o artefato. A segunda e a terceira coluna devem ter como limite o resultado da divisão da largura e altura do vídeo pelo tamanho da janela desejada, respectivamente, pois seus valores indicam qual grupo de *pixels* será afetado e não o *pixel* inicial de aplicação da DCT. Caso o parâmetro -r e respectivo arquivo de degradação *raffle* sejam omitidos, a blocagem será aplicada a todos os *frames* do vídeo, em sua totalidade.

Os subníveis de energia (diagonais da matriz resultante da DCT) que serão cancelados dependem diretamente do tamanho da janela a ser aplicada. Seja uma janela de dimensões AxA, existem N = 2A-1 níveis de energia. O maior nível, 2A-1, representa o nível de energia DC (*Differential Coding*) (a média dos valores do bloco). Nos demais níveis, conforme o nível diminui, a energia diminui porém a frequência aumenta. São conhecidos como coeficientes AC (*Arithmetic Coding*). A Figura 27 apresenta a organização dos níveis de energia numa janela de dimensões 8x8 *pixels* .

O arquivo gerado deve ter as mesmas características do original: tamanho em bytes, largura e altura.

Como exemplo de utilização temos que na execução do comando a seguir, o arquivo saida.yuv de dimensões 352x288 será gerado. O vídeo original será submetido aos valores

DC	14	13	12	11	10	9	8
14	13	12	11	10	9	8	7
13	12	11	10	9	8	7	6
12	11	10	9	8	7	6	5
11	10	9	8	7	6	5	4
10	9	8	7	6	5	4	3
9	8	7	6	5	4	3	2
8	7	6	5	4	3	2	1

Figura 27: Diagonais de energia do bloco DCT.

Fonte: Autoria própria.

contidos no arquivo raffle_352x288.rff e cada janela de dimensões 8x8 terá os níveis DCT (ou diagonais) 1,2,4,6,7,15 zerados.

./block -i entrada.yuv -o saida.yuv -s 352x288 -w 8 -l 1,2,4,6,7,15 -r raffle_352x288.rff

A única restrição se deve aos valores do arquivo *raffle*, como já mencionado. Este deve conter três colunas, em que a primeira, segunda e terceira colunas devem ter como limites superiores, respectivamente, o número de *frames* do vídeo original e as dimensões largura e altura divididas pelo tamanho da janela (352/8 = 44 e 288/8 = 36).

4.2.3 BLUR

A ferramenta blur aplica o artefato de borramento utilizando para tanto um filtro da média ou um filtro da mediana, de dimensão configurável. O arquivo de vídeo de saída é gerado aplicando-se o artefato de acordo com o arquivo *raffle* fornecido por parâmetro. Este, por sua vez, deve conter duas colunas: a primeira indica o *frame* inicial de aplicação, a segunda indica a duração do artefato. Os parâmetros de configuração e respectivas descrições podem ser observados adiante:

Assim como na ferramenta de blocagem, a omissão do parâmetro -r e do arquivo de degradação *raffle* implica na aplicação do borramento em todo o vídeo, de acordo com os parâmetros fornecidos.

Numa possível execução da ferramenta como indicada abaixo, o filtro da média de dimensões 5x5 *pixels* será aplicado no vídeo original, gerando o vídeo degradado saida.yuv de

./blur

`--input`	`-i`	arquivo_de_entrada
`--output`	`-o`	arquivo_de_saída
`--size`	`-s`	dimensões do vídeo de entrada no formato WxH (largura por altura em *pixels*)
`--blur`	`-b`	tipo do filtro a ser aplicado
`--window`	`-w`	tamanho do filtro
`--rafflelist`	`-r`	arquivo_raffle
`--help`	`-h`	menu de ajuda

dimensões iguais às do original (Full-HD).

./blur -i entrada.yuv -o saida.yuv -s 1920x1080 -b average -w 5

Observações:

- para utilizar o filtro da média, deve-se fornecer o parâmetro -b *average*, para o filtro da mediana deve-se fornecer o parâmetro -b *median*;
- a menor dimensão para qualquer tipo de filtro é 3;

4.2.4 NETSIM

A ferramenta Netsim foi desenvolvida buscando simular degradações ocasionadas pelo processo de decodificação de um streaming de vídeo onde houve perda de informação nas camadas de transporte, rede ou enlace. O Netsim desconsidera, portanto, os artefatos decorrentes do processo de codificação e encapsulamento do vídeo.

As simulações efetuadas pela ferramenta atuam sobre vídeos encapsulados no formato MPEG Transport Stream, conforme definido em (ITU, 2006) e consistem no descarte controlado de porções de informação.

A entidade de dados a ser descartada pode ser tanto um único TS quanto o equivalente a um pacote UDP da camada de transporte, o qual comporta usualmente sete unidades TS.

O descarte é controlado precisamente por meio de um arquivo de configuração fornecido como paramêtro ao programa, podendo este ser gerado pela ferramenta *raffle* descrita em 4.2.1.

Este arquivo de configuração deve conter duas colunas com valores numéricos inteiros.

Cada linha será processada sequencialmente, sendo que o primeiro número indica quantas entidades serão transportadas com sucesso e o segundo quantas serão descartadas. A Figura 28 ilustra a sequência.

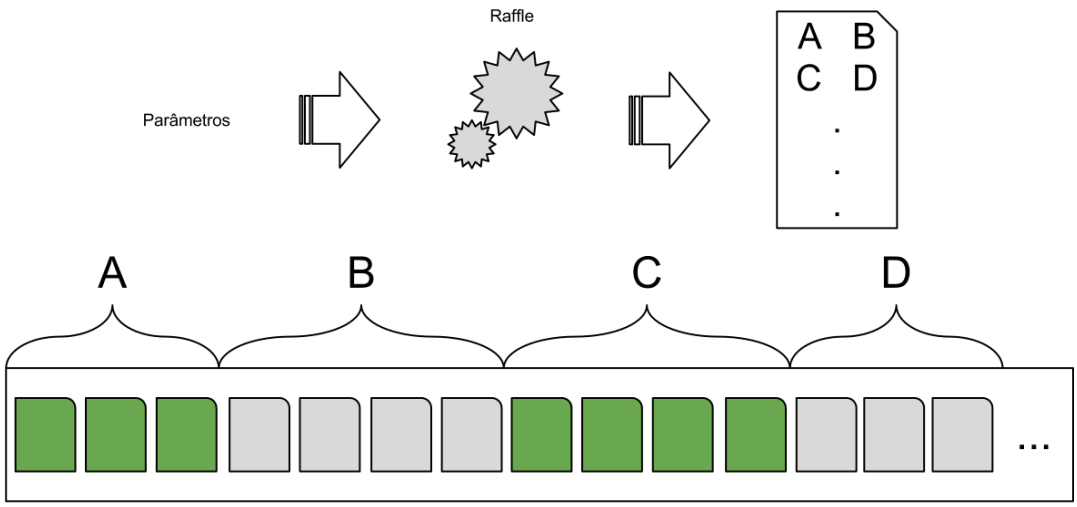

Figura 28: Ilustração do arquivo de configuração de descartes da ferramenta Netsim. Fonte: Autoria Própria.

Neste caso, *A* e *C* indicam o número de entidades que atingem seu destino com sucesso e *B* e *D* indicam quantas serão descartadas. Sendo assim o streaming de vídeo da Figura 35 teria seus primeiros *A* pacotes intactos, seguidos de *B* perdidos, mais *C* pacotes intactos e ainda *D* perdidos novamente.

./netsim

`--input`	`-i`	Define o caminho (absoluto ou relativo) e arquivo do vídeo a ser processado pela simulação.
`--output`	`-o`	Define o caminho(absoluto ou relativo) e arquivo onde será armazenado o vídeo resultante.
`--ts`	`-t`	(opcional) Se presente, a entidade de descarte considerada é um TS, caso contrário a unidade é um pacote UDP.
`--raffle`	`-r`	Indica o caminho (absoluto ou relativo) e arquivo a ser usado como configuração de descartes.

4.2.5 METRIC

A ferramenta Metric se trata da implementação de três métricas objetivas, as mesma encontradas na implementação do SASQV:

- MSE
- PSNR
- MSSIM

Implementada em C++, também na forma de uma ferramenta *stand-alone*, seu funcionamento é baseado em verificar disparidades entre dois vídeos fornecidos seguindo uma das métricas implementadas e fornecer um resultado numérico. Os vídeos a serem comparados devem possuir as mesmas dimensões e número de *frames* para serem passíveis de comparação. A ferramenta Metric pode receber os seguintes parametros:

`--input`	`-i`	Define o caminho(absoluto ou relativo) e arquivo onde um dos vídeos a serem comparados se encontra.
`--reference`	`-r`	Define o caminho(absoluto ou relativo) e arquivo onde o segundo vídeo a ser comparado se encontra.
`--size`	`-s`	Define as dimensões (em *pixels*) dos vídeos a serem comparados. Deve ser fornecida no formato 'largura'x'altura'.
`--metric`	`-m`	Define qual métrica será utilizada na comparação entre vídeos, sendo uma string entre MSE, PSNR ou MSSIM.
`--window`	`-w`	Define qual o tamanho da janela (em *pixels*) a ser usado se a métrica adotada for MSSIM.

4.3 INTERFACE GRÁFICA

A presente seção irá descrever com maiores detalhes as funcionalidades da *interface* gráfica desenvolvida para o SASQV2, separados conforme a finalidade de cada janela presente na mesma.

4.3.1 SESSÃO

A janela de Sessão possui dois momentos: o de configuração, em que os dados são fornecidos em três telas diferentes; e o de sessão propriamente dita, em que os vídeos são apresentados e avaliados de acordo com a métrica escolhida.

A primeira tela trata de configurações básicas da sessão, tal como um nome e descrição para facilitar a identificação, assim como a métrica que será utilizada e o número de espectadores. Devido as limitações encontradas no *hardware*, somente as métricas DSIS e SDSCE foram implementadas. As demais, DSCQS e SSCQE, que estão em modo teste no equipamento de avaliação, não foram implementadas pois a modificação do *firmware* não faz parte do escopo deste projeto.

Na sequência é apresentada uma tela em que pode-se buscar e selecionar vídeos para serem exibidos na sessão de avaliação. Por fim, é apresentada a tela para selecionar quais dispositivos remotos serão utilizados na avaliação. Ao final da configuração, pode-se dar início ao processo de avaliação subjetiva onde os vídeos selecionados serão então exibidos.

Na Figura 29 pode-se observar o ambiente de exibição dos vídeos durante a avaliação segundo a métrica SDSCE. Para a execução desta métrica são executados dois players distintos de forma sincronizada, garantindo que ambos exibam o mesmo *frame*.

Figura 29: Exemplo de sessão de avaliação.

Fonte: Autoria Própria.

4.3.2 FERRAMENTAS

Esta é a janela de maior complexidade no SASQV2, dado que nela se encontram os controles e opções que permitem usar todas as ferramentas descritas em 4.2. Por meio das abas presentes na parte superior da janela é possível acessar três telas diferentes: Gerador de Artefatos, Avaliador Objetivo e Gerador de Aleatoriedade. As funções específicas de cada uma delas são descritas a seguir.

4.3.2.1 GERADOR DE ARTEFATOS

A tela do gerador de artefatos permite controlar de forma gráfica a execução e parametrização das ferramentas *block*, *blur* e *netsim*. Nela pode-se observar três regiões distintas, presente na figura 30:

- Uma lista de busca dos vídeos passíveis de degradação.
- Controladores específicos para cada ferramenta de degradação.

- Uma fila de tarefas de degradação a serem executadas.

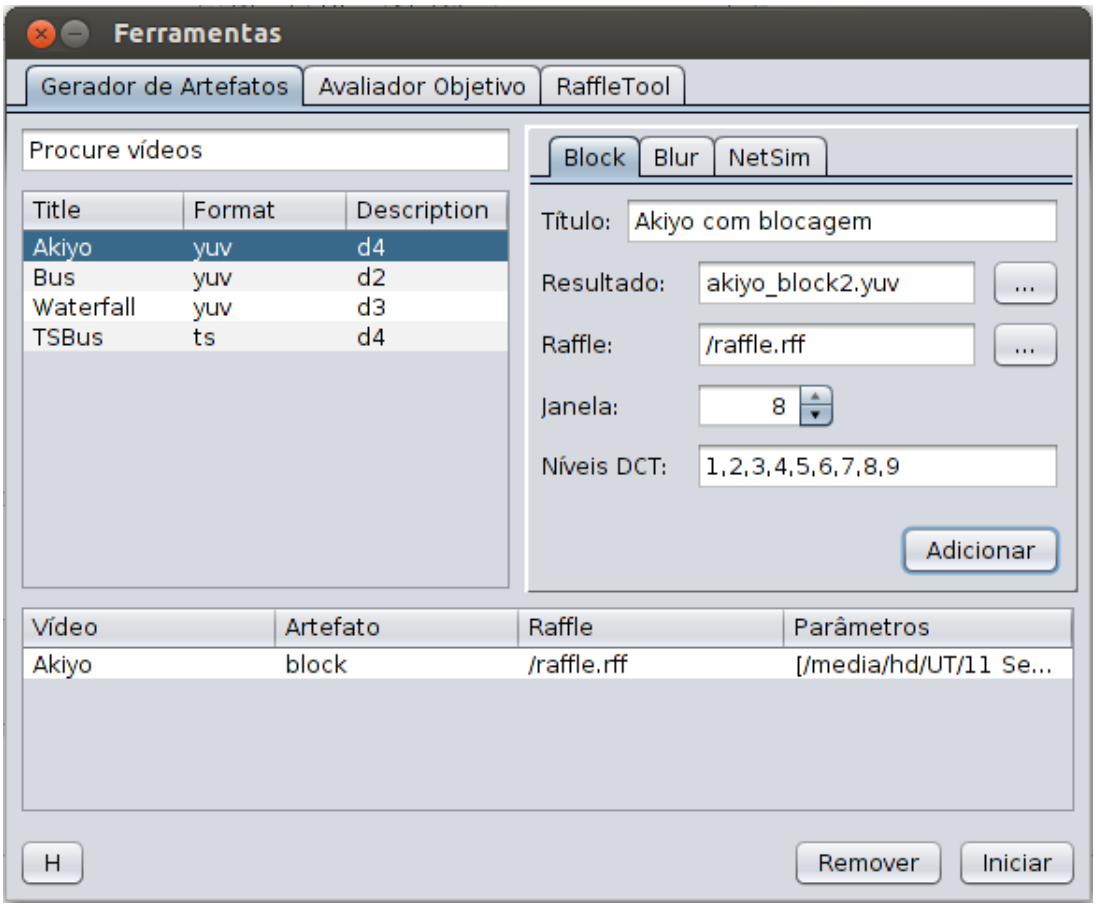

Figura 30: Janela do gerador de artefatos.

Fonte: Autoria Própria.

Para adicionar um artefato em um vídeo é preciso selecionar um vídeo da lista e configurar uma das três ferramentas a disposição. Os parâmetros disponíveis na *interface* são condizentes com os parâmetros apresentados em 4.2 para cada ferramenta. Ao clicar no botão *Adicionar*, uma tarefa com os detalhes do processo selecionado sera adicionada à lista na parte inferior da tela. É possível enfileirar quantas tarefas forem necessárias e então dar início ao processo de degradação dos vídeos clicando em iniciar.

4.3.2.2 AVALIADOR OBJETIVO

Nesta tela existem quatro componentes distintos que permitem configurar e utilizar a ferramenta *metric* citada em 4.2.5:

- Uma lista de busca dos vídeos passíveis de avaliação.

- Uma lista de busca dos vídeos que podem ser tomados como referência.
- Um seletor de métricas objetivas dispiníveis.
- Uma fila de tarefas de avaliação a serem executadas.

Para efetuar uma avaliação objetiva é preciso selecionar dois vídeos a serem comparados, um como referência e outro como amostra, além da métrica desejada e clicar no botão *Adicionar*. Com isso, os vídeos e a métrica atualmente selecionados serão adicionados à lista de tarefas. Ao se clicar no botão *Iniciar* todas as tarefas da lista serão processadas e uma janela de diálogo mostrará o valor resultante de cada avaliação, os quais também podem ser observados na janela de resultados descrita em 4.3.3.

4.3.2.3 GERADOR DE ALEATORIEDADE

Aqui é possível configurar e utilizar a ferramenta *raffle* de 4.2.1 por meio dos seguintes campos apresentados:

- Nome do arquivo a ser gerado.
- Número de elementos a serem gerados.
- Quantidade de colunas no arquivo de saída.
- Tabela de configuração de distribuições estatísticas.

Na tabela de configuração, a primeira linha tem o propósito especial de determinar a duração dos artefatos gerados segundo uma distribuição probabilística a ser especificada. As linhas seguintes permitem a configuração das distribuições utilizadas nas *n* colunas presentes no arquivo. Os parâmetros de configuração são exatamente os mesmos fornecidos à ferramenta *raffle* em 4.2.1

4.3.3 RESULTADOS

A janela de resultados é responsável pela visualização e comparação visual dos resultados obtidos por meio de métricas objetivas ou subjetivas. Nesta tela pode se selecionar um entre dois métodos de visualização: resultados por sessão e resultados por vídeo referência. No primeiro, o sistema reúne todas as notas subjetivas de uma determinada sessão e as exibe na forma de um gráfico de barras, onde cada barra representa um vídeo, indicando o valor da

média e desvio padrão. Já o segundo exibe um gráfico de pontos, onde o eixo das abscissas apresenta valores referentes à métrica objetiva e o eixo das ordenadas à métrica subjetiva, sendo que cada ponto representa um vídeo degradado a partir de um mesmo vídeo referência.

4.3.4 CONFIGURAÇÕES

Esta janela tem o simples propósito de permitir configurar alguns aspectos do sistema SASQV2, tais como os diretórios padrão e os intervalos usados nas fases de apresentação durante a avaliação subjetiva.

4.3.5 AJUDA

A tela de ajuda consiste de uma ampla área de texto dedicada à exibição de páginas HTML, na qual foram confeccionados os documentos de ajuda. Esta janela conta ainda com uma estrutura de navegação topificada no canto esquerdo, proporcionando o acesso direto a uma determinada seção.

4.4 CONSIDERAÇÕES

A escolha pelo desenvolvimento de ferramentas e módulos independentes na fase de projeto mostrou sua eficiência ao longo do desenvolvimento. A primeira vantagem foi vista na efetivação da fase de *design* de código para cada ferramenta em paralelo, divididas entre os membros do projeto. Essa mesma característica foi vista também durante o desenvolvimento, onde os integrantes podiam trabalhar de forma distribuída. Contando também com a ferramenta *git* e o repositório *Github* foi possível definir metas semanais de desenvolvimento e integração, minimizando a necessidade de reuniões, as quais foram realizadas usualmente no horário de aula e aos fins de semana.

5 RESULTADOS

Com o conjunto de ferramentas desenvolvidos neste projeto aliado a uma base de vídeos originais é possível produzir e avaliar uma nova e diversificada base com vídeos degradados em qualidade e quantitade variável. Neste Capítulo serão verificados os resultados obtidos a partir do processamento de vídeos efuados por estas ferramentas. Primeiramente uma avaliação subjetiva das ferramentas de degradação utilizando diversas amostras retiradas de um vídeo da base. A seguir é feita a validação dos resultados obtidos pela ferramenta de métricas objetivas, tomando como base os videos obtidos em (NYU, 2012). Por fim é feita uma análise das notas de avaliação objetivas buscando mensurar o impacto que determinadas degradações podem ter sobre um conjunto de vídeos com características diferentes.

5.1 VALIDAÇÃO DAS FERRAMENTAS DE ARTEFATOS

Foram efetuados diversos processos de degradação sobre o vídeo bus, obtido em (GROUP, 2008), empregando diferentes paramêtros a cada processamento e para cada ferramenta, buscando identificar visualmente e os artefatos produzidos e determinar sua semelhanca com os vistos em situações reais.

A primeira ferramenta a ser avaliada é a *block*. Para a avaliação foram gerados 14 vídeos onde é efetuado o processo de blocagem, com blocos 8x8, de forma integral — para todos os blocos em todos os quadros. A cada novo vídeo gerado foram eliminadas de forma cumulativa diagonais na matriz resultante da transformação DCT, partindo da diagonal mais inferior, até que no último vídeo restasse somente a componente DC da transformada. Na Figura 31 são apresentados recortes de 160 por 160 *pixels* a partir da posição 64x160 retirados do *frame* de número 60 de cada um dos 14 vídeos obtidos, sendo que o primeiro recorte foi obtido do vídeo original.

Observando os recortes com cautela é possível notar degradações extremamente sutis a partir do recorte *07*, verificadas com mais facilidade nos arredores da publicidade na lateral do ônibus.No recorte de número *10* as regiões de alta frequência da imagem denunciam

degradações mais acentuadas, e no *11* já não é mais possível identificar o que está escrito na placa de publicidade, embora ainda seja razoavél identificar a cena da imagem. Nos três recortes seguintes o efeito de blocagem passa a tomar conta da imagem e apenas objetos de grande escala passam a ser identificáveis, sendo que no recorte de número *14* até mesmo a identificação da cena fica prejudicada.

Para a avaliação da ferramenta *blur* foram feitas duas sequências de testes, a primeira delas utilizando o filtro de médias e a segunda utilizando o filtro da mediana. Em cada um dos testes foram utilizadas máscaras matriciais de tamanhos 3x3, 5x5 e 7x7. Nas Figuras 32 e 33 são apresentados recortes de 160 por 160 *pixels* a partir da posição 64x160 retirados do *frame* número 60 de cada vídeo.

Na validação visual da ferramenta *raffle* foi gerado um arquivo de sorteio contendo 5 mil elementos. A distribuição no tempo foi configurada como triangular no intervalo de 0 a 10 com pico no *frame* 5, sendo que a duração de cada artefato foi distribuída uniformemente no intervalo de 1 a 3 *frames*. Já a distribuição no espaço foi uniforme em toda a largura e altura do *frame*. A Figura 34 mostra a sequência de recortes retirados dos 11 *frames* contendo blocos gerados a partir do arquivo raffle obtido com a configuração acima. Para facilitar a visualização, foi eliminada a componente DC da DCT para cada bloco sorteado.

Para o processo de validação da ferramenta *netsim* foi necessário obter um vídeo encapsulado em um TS, para isso foi utilizada a ferramenta ffmpeg (FFMPEG, 2012) para converter o vídeo bus para a codificação MPEG-2 encapsulado em um TS com o seguinte comando:

```
ffmpeg -i arquivo_de_entrada -s cif -sameq -f mpegts -vcodec
mpeg2video arquivo_de_saída
```

O TS resultante foi então processado pela ferramenta, configurada para fazer os seguintes descartes:

```
10 100 1000 10 100 10 1 10 100 50
```

A figura 35 contém amostras retiradas dos *frames* 1, 9 e 40, onde podem ser verificados artefatos como *jerkiness*, *blocking* e *bleeding*.

5.2 VALIDAÇÃO DA FERRAMENTA DE MÉTRICAS

Para validação dos valores de MSE, PSNR e MSSIM calculados pela ferramenta *metric* foi feita a comparação destes com os obtidos pela ferramenta MSU *Video Quality Measurement*

Tool (TEAM; GROUP, 2012). Foram utilizados 5 vídeos diferentes, e os respectivos vídeos degradados, obtidos em (NYU, 2012). Os resultados das comparações se encontram nas tabelas 10, 9 e 11.

Tabela 9: Comparação dos resultados de PSNR.

Video	MSU	SASQV2	Erro (%)
aircraft	23.49501	23.49501	0.00000
liberty	29.21472	29.21472	0.00000
ship	25.56667	25.56667	0.00000
stockholm	18.90942	18.90942	0.00000
whale	20.59830	20.59830	0.00000

Fonte: Autoria próprio

Tabela 10: Comparação dos resultados de MSE.

Video	MSU	SASQV2	Erro (%)
aircraft	290.78970	290.78970	0.00000
liberty	77.91273	77.91273	0.00000
ship	180.47363	180.47362	-0.00001
stockholm	835.86987	835.86987	0.00000
whale	566.53094	566.56611	0.00621

Fonte: Autoria própria

Tabela 11: Comparação dos resultados de MSSIM.

Video	MSU	SASQV2	Erro (%)
aircraft	0.79457	0.80723	1.59306
liberty	0.85415	0.85627	0.24843
ship	0.70462	0.70628	0.23530
stockholm	0.42486	0.43327	1.97948
whale	0.81918	0.82240	0.39320

Fonte: Autoria própria.

Observando os resultados para MSE e PSNR pode-se dizer que os resultados são fiéis e que a ferramenta se comporta como o esperado. A respeito do MSSIM é difícil dizer ao certo o motivo da pequena diferença encontrada nos resultados, especialmente pela variedade de parâmetros a serem levados em conta nesta métrica. Levando em conta a documentação do MSU sabe-se que este utiliza pesos uniformes para toda a janela de cálculo, mas não se encontrou o tamanho adotado ou o método de escolha das janelas possíveis. O teste utilizando a ferramenta presente no SASQV2 utiliza janelas 8x8 adotando janelas não sobrepostas.

5.3 ANÁLISE DE IMPACTO SOBRE MÉTRICAS OBJETIVAS

Para este teste foi utilizada a combinação de três distribuições de artefatos geradas pela ferramenta raffle, que foram concatenadas para criar três rajadas de ruídos em um conjunto de vídeos. As configurações adotadas foram as seguintes:

1. Distribuição 1
    - Duração: uniforme de 1 a 5
    - Frames: normal com $\mu = 25$ e $\sigma = 10$
    - largura: uniforme de 1 a 10
    - altura: uniforme de 20 a 30
2. Distribuição 2
    - Duração: constante em 3
    - Frames: uniforme de 70 a 100
    - largura: uniforme de 10 a 40
    - altura: uniforme de 10 a 20
3. Distribuição 3
    - Duração: uniforme de 3 a 5
    - Frames: normal com $\mu = 220$ e $\sigma = 40$
    - largura: uniforme de 1 a 43
    - altura: normal com $\mu = 28$ e $\sigma = 5$

O número de artefados em cada *frame* obtido a partir desta configuração é apresentado na figura 36.

Esta configuração foi então usada na aplicação de blocagem em três vídeos diferentes: Akiyo, CoastGuard e Foreman, todos possuindo 300 *frames* de duração e obtidos em (XIPH.ORG, 2012). Estes vídeos foram escolhidos pelas suas características de textura e movimento, possibilitando uma análise variada.

Nas figuras 37 e 38 pode-se acompanhar a evolução das métricas MSE e MSSIM quadro a quadro para os três vídeos degradados. Observando o comportamento das métricas nota-se

que, apesar da natureza e da distribuição dos artefatos utilizados ser exatamente a mesma para todos os vídeos, os valores obtidos possuem picos diferentes, embora o comportamento seja semelhante.

Outro fato notável é a análise individual das métricas, que dependendo do conjunto de *frames* analisado pode levar a conclusões diferentes a respeito do nível de degradação encontrado em cada vídeo.

5.4 RESUMO E CONCLUSÃO DO CAPÍTULO

Neste capítulo foram apresentados resultados visuais proporcionados pelo uso das ferramentas desenvolvidas durante o projeto, demonstrando a flexibilidade e usabilidade do sistema desenvolvido. Também foram validados os resultados obtidos pelas avaliações objetivas e discutidos aspectos relevantes sobre as possíveis conclusões retiradas a partir dos valores obtidos.

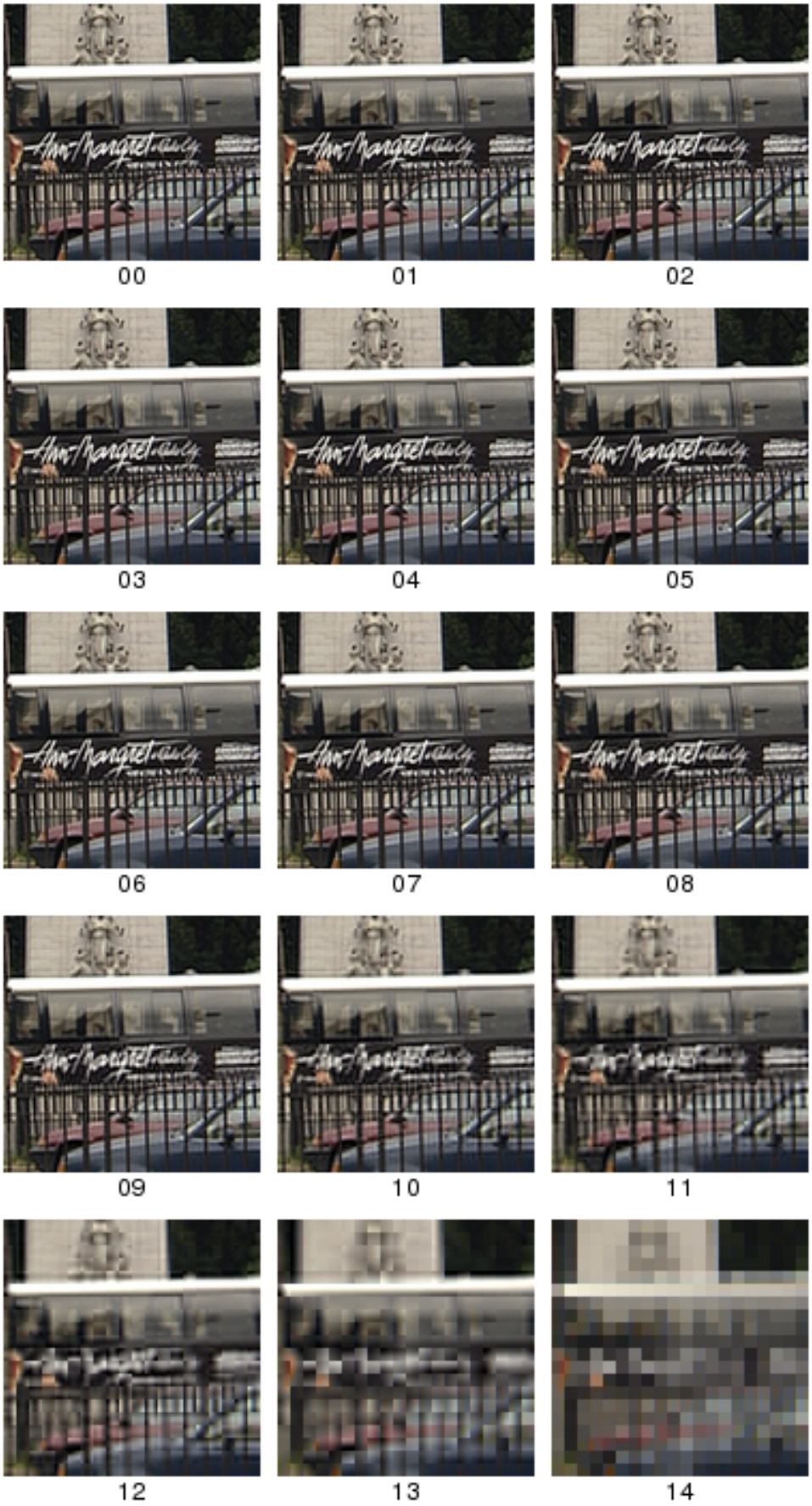

Figura 31: Sequência de degradações eliminando gradativamente diagonais da DCT.

Fonte: Autoria Própria.

Figura 32: Sequência de borramentos aplicando filtro da média com matrizes (00) 3x3, (01) 5x5, (02) 7x7.

Fonte: Autoria Própria.

Figura 33: Sequência de borramentos aplicando filtro da mediana com matrizes (00) 3x3, (01) 5x5, (02) 7x7.

Fonte: Autoria Própria.

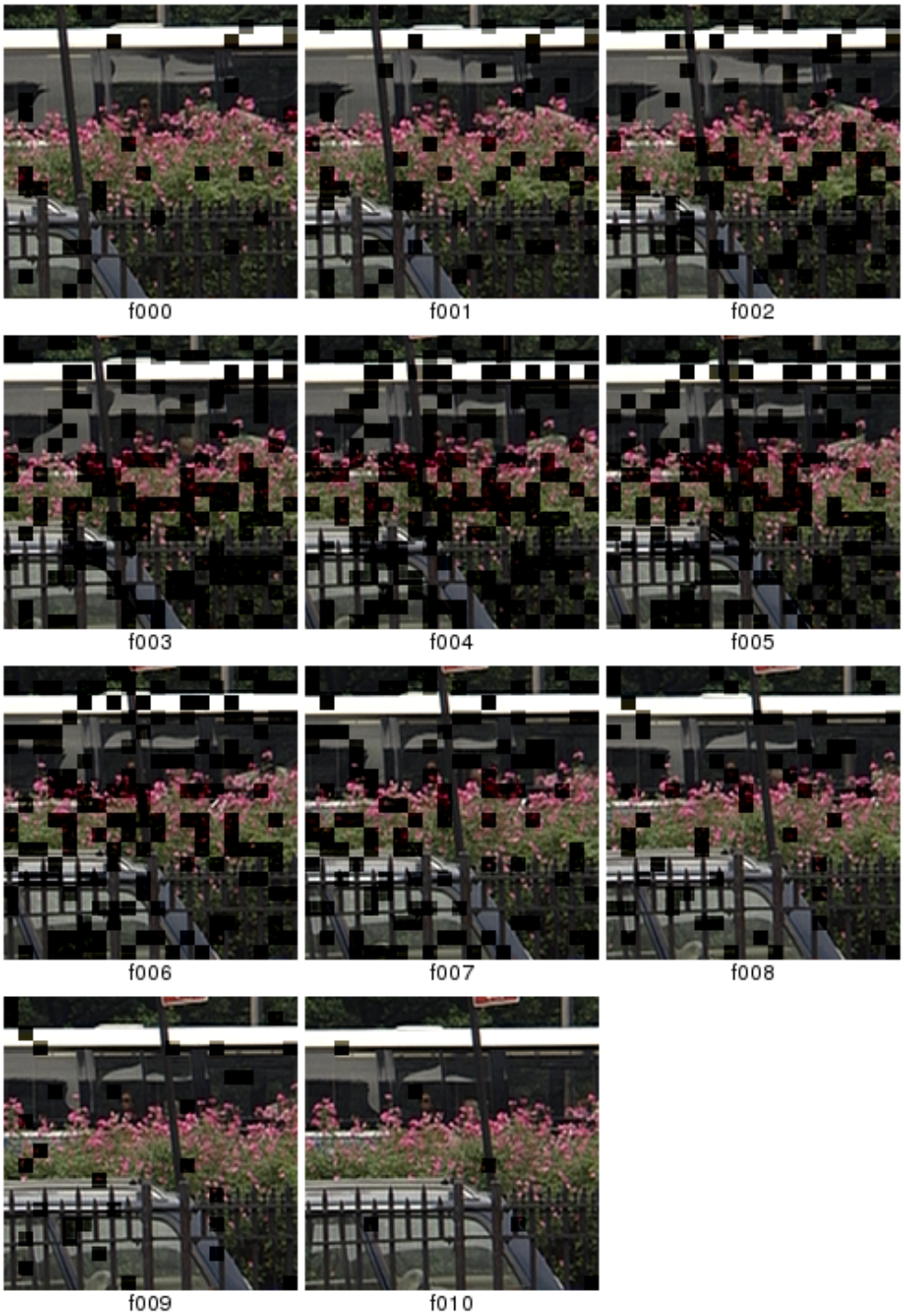

Figura 34: Sequência de *frames* degradados a partir de um arquivo gerado pela ferramenta *raffle*.

Fonte: Autoria Própria.

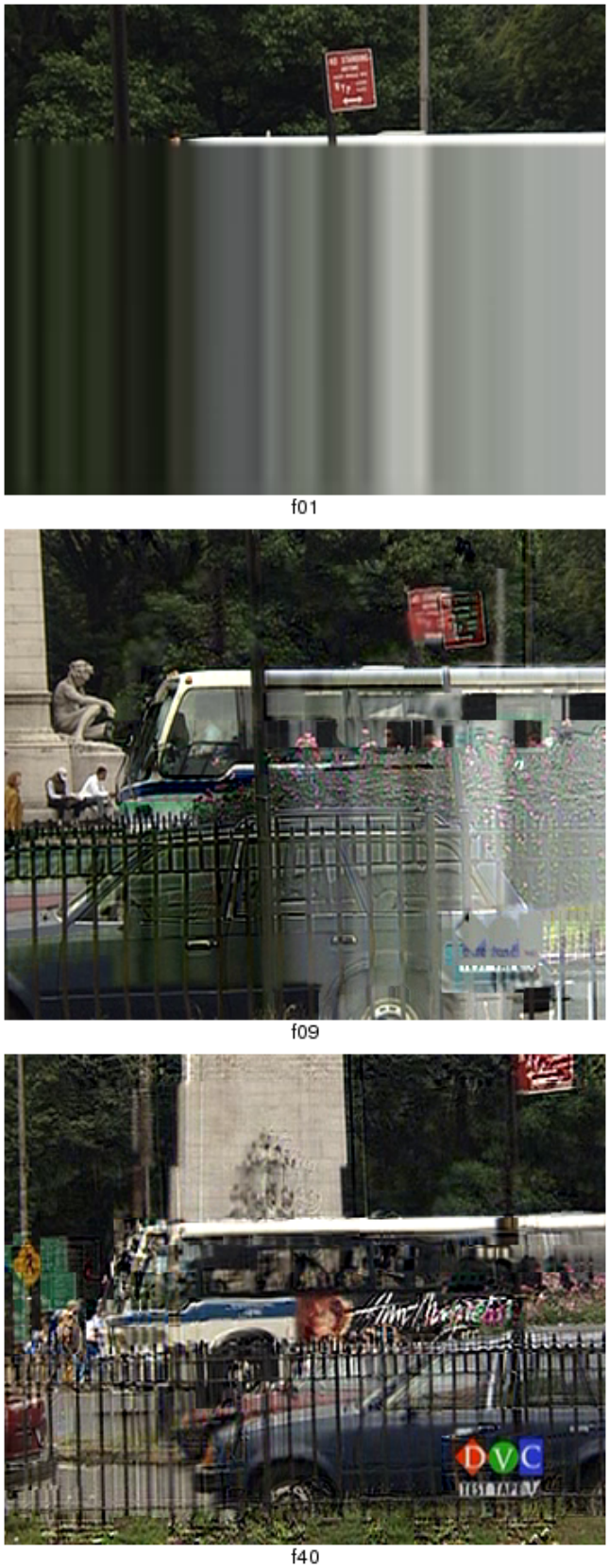

Figura 35: Amostra de artefados obtidos com a ferramenta *netsim* (de cima para baixo: *bleeding, blocking* e *jerkiness*).

Fonte: Autoria Própria.

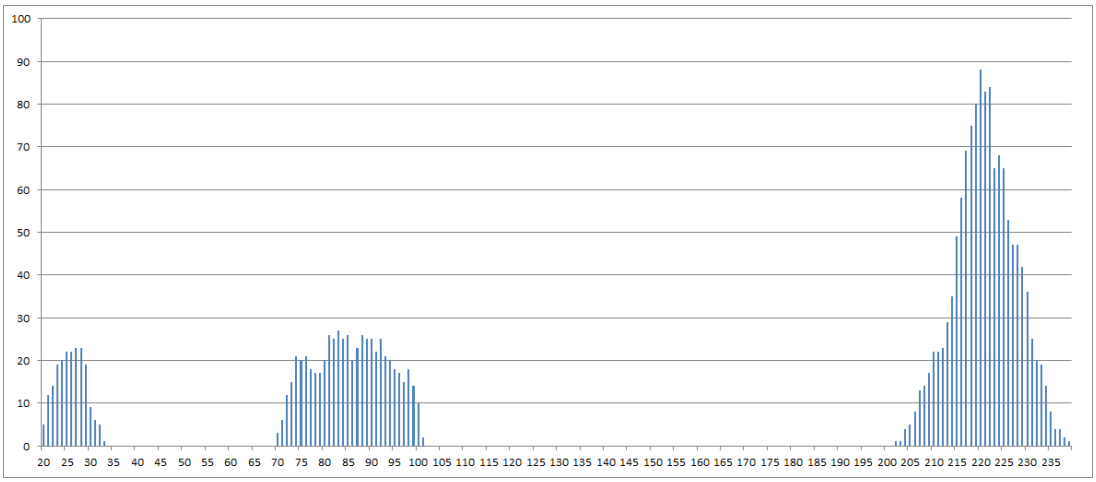

Figura 36: Histograma dos artefatos produzidos

Fonte: Autoria Própria.

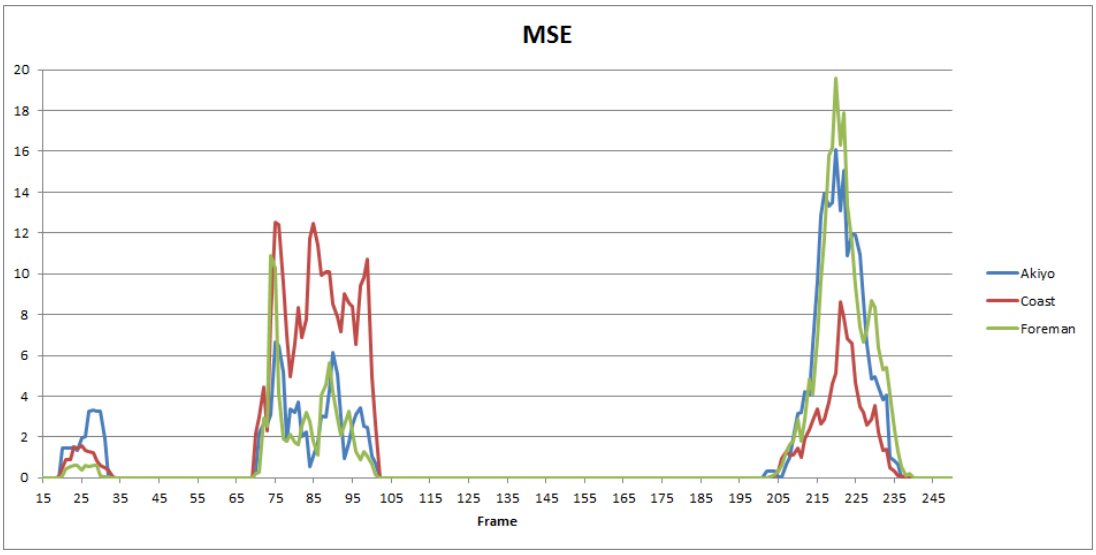

Figura 37: MSE quadro a quadro

Fonte: Autoria Própria.

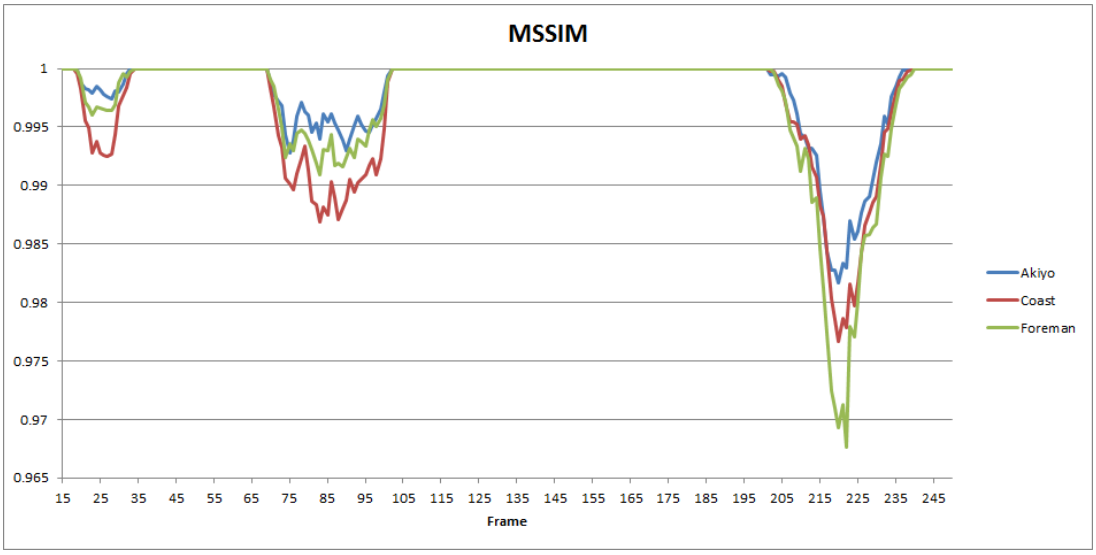

Figura 38: MSSIM quadro a quadro

Fonte: Autoria Própria.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 CONCLUSÕES

O SASQV2 permite a utilização de vídeos em seu formato bruto, ao contrário do sistema original. Arquivos neste formato são bastante utilizados em bases de dados para testes e validações na literatura pois não sofreram nenhum tipo de perda de informação desde sua captura.

Os principais resultados do projeto das ferramentas são modularização e flexibilidade. É possível utilizar qualquer ferramenta independentemente da *interface* para se obterem vídeos degradados. Ainda, o desacoplamento da ferramenta *raffle*, que gera arquivos de degradação, das demais, abriu algumas possibilidades importantes. É possível utilizar como *input* nas demais ferramentas arquivos gerados em outros *softwares*, desde que possuam o mesmo formato *raffle*. A decisão de projeto em favor da utilização de arquivos de degradação permitiu que a mesma degradação pudesse ser reproduzida ou aplicada em outros vídeos. Isso significa que, apesar da aleatoriedade agregada, a reprodução de degradações é possível sem a necessidade de se armazenar o arquivo de vídeo resultante, armazenando-se apenas um arquivo de texto.

Outra diferença em relação ao sistema original reside na *interface* de Ajuda, que não existia originalmente. Desde o início do projeto foi dada a devida importância a este componente do *software*, uma vez que a utilização de todo o sistema e de suas ferramentas depende do completo entendimento dos detalhes de utilização. Esta *interface* foi desenvolvida na linguagem HTML e pode ser acessada através do SASQV2 assim como de qualquer *browser*.

O aprimoramento inicialmente proposto para a degradação de vídeos foi alcançado através da parametrização de cada artefato, que permite o completo controle da aplicação de degradações tanto espacial quanto temporalmente. O sistema original aplicava apenas as degradações de borramento e blocagem de forma generalizada, ou seja, todo o vídeo era degradado da mesma forma. Além de aprimorar estas ferramentas, o SASQV2 conta com um simulador de artefatos gerados em *streaming*. As decisões de projeto do simulador permitiram facilitar a usabilidade da ferramenta pois foi eliminada a necessidade de uma rede completa e

interfaces físicas devidamente configuradas para simular perdas, como visto em trabalhos dessa natureza.

Em uma avaliação visual os artefatos gerados se mostraram bastante próximos dos reais, embora a previsão de movimento e localização de objetos nas imagens não tenha sido implementada

Com a troca da *interface* de FLEX para Java eliminou-se a necessidade de uma ferramenta de desenvolvimento paga, atendendo ao requisito de projeto de utilizar ferramentas de distribuição livre (definido em 3.1.2). Além disso, o controle, desenvolvimento e *debug* de código tornou-se mais simples, pois foram utilizadas bibliotecas há bastante tempo estáveis, ao contrário das utilizadas para a comunicação FLEX/Java.

A implementação de métricas subjetivas foi restringida pelo *firmware* do equipamento de avaliação, que não foi modificado por não fazer parte do escopo do projeto. Mesmo assim, outras métricas podem ser implementadas de forma simples pois o código foi desenvolvido para dar o devido suporte, bastando que o *firmware* seja corrigido para atender às normas.

A portabilidade do sistema foi alcançada ao ser possível executá-lo com apenas um arquivo .jar (executável Java), tendo como requisito a instalação do JRE.

Por fim, é possível evidenciar o papel multidisciplinar deste projeto, que integra conhecimentos adquiridos ao longo do curso de Engenharia de Computação como técnicas de programação, gerenciamento de projetos relações interpessoais, estrutura de dados, técnicas de processamento digital de imagens, entre outras. Como resultado desta integração, este projeto se mostra uma solução de engenharia que busca o aperfeiçoamento de uma tecnologia cada vez mais presente no dia a dia das pessoas.

6.2 TRABALHOS FUTUROS

Uma sugestão na parte de *hardware* é a implementação das métricas subjetivas que estão em modo teste para que seja possível realizar sessões de avaliação.

Já na parte de *software*, podem ser desenvolvidas novas ferramentas que parametrizem outros artefatos como efeito escada, *ringing*, contornos falsos, *jerkiness*, etc. Sugere-se o contínuo aperfeiçoamento dos artefatos já desenvolvidos para que se aproximem cada vez mais dos reais. Pode-se implementar a geração de níveis aleatórios a serem eliminados na DCT da blocagem. No NetSim é possível implementar novas formas de perda de pacotes como embaralhamento, ruído e atrasos.

Sobre o banco de dados, uma sugestão é alterá-lo de forma a permitir o armazenamento de dados referentes a avaliação objetiva que sejam calculados *frame* a *frame*, o que por sua vez pode permitir novas visualizações de resultados. Outra implementação sugerida é o recarregamento de sessões de avaliação, para que não seja necessário recriar uma mesma sessão.

REFERÊNCIAS

ALBINI, F. L. **Geração e Avaliação de Artefatos em Vídeo Digital**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2009.

BOX, G. E. P.; MUELLER, M. E. A note on the generation of random normal deviates. **The Annals of Mathematical Statistics**, v. 29, p. 610–611, 1958.

BRICE, R. **Newnes guide to digital television**. Newnes, 2000. ISBN 9780750645867. Disponível em: <http://books.google.com.br/books?id=FD4fAQAAIAAJ>.

COMMUNITY, J. **About - Hibernate**. 2012. Disponível em: <www.hibernate.org/about.html>. Acesso em: 04 de Junho de 2012.

COMMUNITY, J. **History - Hibernate**. 2012. Disponível em: <www.hibernate.org/about/history>. Acesso em: 04 de Junho de 2012.

CONSERVANCY, S. F. **Distributed Version Control System**. 2012. Disponível em: <http://git-scm.com/about>. Acesso em: 19 de junho de 2012.

CORPORATION, J. U. **Reference Guide to Digital Video Technology, Testing and Monitoring**. 2008.

DARONCO, L. C. **Avaliação Subjetiva de Qualidade Aplicada à Codificação de Vídeo Escalável.** Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2009.

FARIAS, M. C. Q.; FOLEY, J. M.; MITRA, S. K. Detectability and annoyance of synthetic blocky, blurry, noisy and ringing effects. **IEEE Transactions on Signal Processing**, v. 55, p. 2954–2964, 2007.

FFMPEG. **A complete, cross-platform solution to record, convert and stream audio and video.** 2012. Disponível em: <http://ffmpeg.org/about.html>. Acesso em: 11 de julho de 2012.

FOURCC. **YUV Pixel Formats**. 2011. Disponível em: <fourcc.org/yuv.php>. Acesso em: 16 de Junho de 2012.

GITHUB, I. **GitHub Social Coding**. 2012. Disponível em: <https://github.com/about>. Acesso em: 15 de março de 2012.

GROUP, V. T. R. **YUV Video Sequences**. 2008. Disponível em: <trace.eas.asu.edu/yuv/index.html>. Acesso em: 22 de Abril de 2012.

ITU, I. T. U. **T-REC-P.930 - Principles of a Reference Impairment System for Video**. 1996.

ITU, I. T. U. **ITU-R BT.500-11 - Methodology for the subjective assessment of the quality of television pictures**. 2002.

ITU, I. T. U. **T-REC-H.222.0 - Transmission multiplexing and synchronization**. 2006.

LABORATORY, I. T. **NIST/SEMATECH e-Handbook of Statistical Methods**. 2012. Disponível em: <http://www.itl.nist.gov/div898/handbook/>. Acesso em: 20 de março de 2012.

LAMBRECHT, C. J. V. D. B. **Vision Models and Applications to Image and Video Processing**. [S.l.]: Springer, 2001.

MARSAGLIA, G.; TSANG, W. W. The ziggurat method for generating random variables. **Journal of Statistical Software**, v. 5, 2000.

NYU, P. I. of. **Quality Assessment Database**. 2012. Disponível em: <vision.poly.edu/index.html/index.php?n=HomePage.QualityAssessmentDatabase>. Acesso em: 23 de Junho de 2012.

ORACLE. **Java Standard Api Ed. 6**. 2011. Disponível em: <http://docs.oracle.com/javase/6/docs/api/>. Acesso em: 16 de junho de 2012.

ORACLE. **MySQL Market Share**. 2012. Disponível em: <www.mysql.com/why-mysql/marketshare>. Acesso em: 02 de Junho de 2012.

PANAGIOTOPOULOU, A.; ANASTASSOPOULOS, V. Super-resolution image reconstruction techniques: Trade-offs between the data-fidelity and regularization terms. **Information Fusion**, v. 13, n. 3, p. 185 – 195, 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1566253510000941>.

PROJECT, G. **GNU Compiler Collection**. 2012. Disponível em: <http://gcc.gnu.org/>. Acesso em: 21 de junho de 2012.

PROJECT, G. **GNU Project**. 2012. Disponível em: <http://www.gnu.org/philosophy/philosophy.html>. Acesso em: 21 de junho de 2012.

PROJECT, G. **Make**. 2012. Disponível em: <http://www.gnu.org/software/make/>. Acesso em: 21 de junho de 2012.

REFERENCE, C. **C++ Reference**. 2012. Disponível em: <http://www.cplusplus.com/reference>. Acesso em: 16 de junho de 2012.

REHME, J. F. **Avaliação da qualidade de vídeo trafegando sobre redes IP.** Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2007.

RICE, J. **Mathematical Statistics and Data Analysis (Second ed.)**. [S.l.]: Duxbury Press, 1995. ISBN 0-534-20934-3.

RICHARDSON, I. **H.264 and Mpeg-4 Video Compression: Video Coding for Next-Generation Multimedia**. Wiley, 2003. ISBN 9780470848371. Disponível em: <http://books.google.com.br/books?id=ECVV_G_qsxUC>.

SANDVINE, I. **Fall 2010 Flobal Internet Phenomena Report**. 2010. Disponível em: <http://www.sandvine.com/downloads/documents/2010 Global Internet Phenomena Report.pdf>. Acesso em: 15 de março de 2012.

SANTOS, E.; SOUZA, E.; LASS, M. **Desenvolvimento de um sistema de auxílio à avaliação objetiva e subjetiva de vídeos digitais**. 2010. TCC Universidade Tecnológica Federal do Paraná.

SCIENCES, I. T. **Final Report from the Video Quality Expertes Group on the Validation of objective models of Video Quality Assessment - Phase II**. 2003. Disponível em: <ftp://vqeg.its.bldrdoc.gov/Documents/VQEG_Approved_Final_Reports/>. Acesso em: 16 de junho de 2012.

SILVA, E. S. R. da. **Especificação, Projeto e Desenvolvimento de Ferramentas de Auxílio à Avaliação Subjetiva de Vídeo**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2009.

SYSTEMS, A. **What is CMY/CMYK Color Space?** 2012. Disponível em: <http://www.acasystems.com/en/color-picker/faq-cmy-color.htm>. Acesso em: 16 de junho de 2012.

SYSTEMS, C. Not all packets are equal, part i. **IEEE Internet Computing**, v. 1089-7801, p. 70–75, 1999.

TEAM, S.; GROUP, M. V. **A complete, cross-platform solution to record, convert and stream audio and video.** 2012. Disponível em: <http://compression.ru/video/quality_measure/video_measurement_tool_en.html>. Acesso em: 11 de julho de 2012.

TEKTRONIX, I. **Picture Quality Analysis System**. 2011. Disponível em: <www,tek.com/datasheet/picture-quality-analysis-system>. Acesso em: 23 de Junho de 2012.

THE Computer Language Benchmarks Game. 2012. Disponível em: <shootout.alioth.debian.org>. Acesso em: 17 de Junho de 2012.

TKN. **EvalVid - A video Quality Evaluation Tool-set**. 2012. Disponível em: <www.tkn.tu-berlin.de/menue/research/evalvid>. Acesso em: 23 de Junho de 2012.

UNIVERSITY, M. S. **Graphics and Media Lab**. 2012. Disponível em: <graphics.cs.msu.ru/>. Acesso em: 23 de Junho de 2012.

VIM. **Vim**. 2012. Disponível em: <http://www.vim.org>. Acesso em: 21 de junho de 2012.

WANG, Z.; BOVIK, A. C.; SHEIKH, H. R. Image quality assesment: From error visibility to structural similarity. **IEEE Transactions on Image Processing**, v. 13, p. 600–612, 2004.

Watson, A. B. et al. Design and performance of a digital video quality metric. In: Rogowitz, B. E.; Pappas, T. N. (Ed.). **Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series**. [S.l.: s.n.], 1999. (Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, v. 3644), p. 168–174.

WIKIPEDIA. **Box-Mueller Transform**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Box–Muller_transform>. Acesso em: 16 de junho de 2012.

WIKIPEDIA. **C++**. 2012. Disponível em: <en.wikipedia.org/wiki/C++>. Acesso em: 14 de Junho de 2012.

WIKIPEDIA. **Chroma Subsampling**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Chroma_subsampling>. Acesso em: 16 de maio de 2012.

WIKIPEDIA. **Color Model**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Color_model>. Acesso em: 16 de maio de 2012.

WIKIPEDIA. **Digital Video**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Digital_video>. Acesso em: 23 de junho de 2012.

WIKIPEDIA. **Distribuição uniforme**. 2012. Disponível em: <http://pt.wikipedia.org/wiki/Distribuição_uniforme>. Acesso em: 16 de junho de 2012.

WIKIPEDIA. **Java**. 2012. Disponível em: <en.wikipedia.org/wiki/Java_(programming_language)>. Acesso em: 14 de Junho de 2012.

WIKIPEDIA. **MySQL**. 2012. Disponível em: <en.wikipedia.org/wiki/MySQL>. Acesso em: 02 de junho de 2012.

WIKIPEDIA. **Normal Distribution**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Normal_distribution>. Acesso em: 16 de junho de 2012.

WIKIPEDIA. **Probability Distribution**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Probability_distribution>. Acesso em: 16 de junho de 2012.

WIKIPEDIA. **Triangular Distribution**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Triangular_distribution>. Acesso em: 16 de junho de 2012.

WIKIPEDIA. **Uniform distribution (continuous)**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Uniform_distribution_(continuous)>. Acesso em: 16 de junho de 2012.

WIKIPEDIA. **Video Compression Picture Types**. 2012. Disponível em: <http://en.wikipedia.org/wiki/Video_compression_picture_types>. Acesso em: 25 de junho de 2012.

WIKIPEDIA. **YUV**. 2012. Disponível em: <http://en.wikipedia.org/wiki/YUV>. Acesso em: 10 de março de 2012.

WINKLER, S. **Digital Video Quality: Vision Models And Metrics**. John Wiley & Sons, 2005. ISBN 9780470024041. Disponível em: <http://books.google.com.br/books?id=NDNfMaht37cC>.

WU, H. R.; RAO, K. R. **Digital Video Image Quality and Perceptual Coding**. [S.l.]: CRC Press, 2005. ISBN 9780824727772.

XIPH.ORG. **Xiph.org Test Media**. 2012. Disponível em: <media.xiph.org/video/derf>. Acesso em: 10 de Março de 2012.

YOUTUBE, L. **YouTube Statistics**. 2012. Disponível em: <http://www.youtube.com/t/press_statistics>. Acesso em: 23 de junho de 2012.