UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIENCIAS EXATAS E NATURAIS
CURSO DE CIENCIA DA COMPUTAÇÃO

CRISTÓVÃO CARLOS DE FREITAS OLIVEIRA

CRIAÇÃO DE *CARTOONS* BASEADA EM FILTRAGEM ESPACIAL DE IMAGENS E DE VÍDEOS DIGITAIS

MOSSORÓ
2018

CRISTÓVÃO CARLOS DE FREITAS OLIVEIRA

CRIAÇÃO DE CARTOONS BASEADA EM FILTRAGEM ESPACIAL DE IMAGENS E DE VÍDEOS DIGITAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal Rural do Semi-Árido como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Leandro Carlos de Souza

Coorientador: Prof. Dr. Daniel Faustino Lacerda de Souza

MOSSORÓ

2018

C257c Carlos, Cristóvão.
CRIAÇÃO DE CARTOONS BASEADA EM FILTRAGEM ESPACIAL DE IMAGENS E DE VÍDEOS DIGITAIS / Cristóvão Carlos. - 2018.
50 f. : il.

Orientador: Leandro Carlos de Souza.
Coorientador: Daniel Faustino Lacerda de Souza.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência da Computação, 2018.

1. Processamento Digital de Imagem. 2. Cartoon. I. Carlos de Souza, Leandro, orient. II. Faustino Lacerda de Souza, Daniel, co-orient. III. Título.

CRISTÓVÃO CARLOS DE FREITAS OLIVEIRA

CRIAÇÃO DE CARTOONS BASEADA EM FILTRAGEM ESPACIAL DE IMAGENS E DE VÍDEOS DIGITAIS

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciência da Computação.

Defendida em: 18 / 04 / 2018.

BANCA EXAMINADORA

Leandro C. Souza

Prof. Dr. Leandro Carlos de Souza - (UFERSA)
Orientador e Presidente

Daniel Faustino L de L

Prof. Dr. Daniel Faustino Lacerda de Souza - (UFERSA)
Coorientador

Leiva Casemiro Oliveira

Prof. Dr. Leiva Casemiro Oliveira - (UFERSA)
Primeiro Membro

Paulo Henrique Lopes Silva

Prof. Dr. Paulo Henrique Lopes Silva - (UFERSA)
Segundo Membro

Nunca Renuncie seus Sonhos.

AGRADECIMENTOS

Agradeço a minha esposa Daniela Santos por está ao meu lado em todos os momentos difíceis.

Agradeço a minha família por me apoiar em toda minha jornada neste curso superior.

Agradeço a todos os professores da graduação que me ensinaram e orientaram nesta jornada.

Agradeço ao meu Orientador Prof. Dr. Leandro Carlos de Souza e o meu coorientador Prof. Dr. Daniel Faustino Lacerda de Souza, por me ajudarem a produzir este trabalho.

Agradeço a banca por me dar a oportunidade de expor o meu trabalho.

Agradeço a Deus acima de tudo por está respirando e em boa saúde.

RESUMO

Desde os primeiros grupos de humanos, a arte se revela em uma forma de expressão da individualidade, da sociedade e do pensamento abstrato. À medida que a tecnologia se desenvolve, os artistas tentam incorporá-la como ferramenta para a geração de arte. Uma forma de expressão artística está ligada à geração de imagens, como a criação artificial e digital de pinturas, fotos ou vídeos, semelhantemente, o *cartoon* que tem se demonstrado como uma forma de arte bastante ativa na última década, tendo em vista o seu aparecimento no mercado de entretenimento, como os filmes de animação que lucram milhões com bilheteria, as programações infantis que fazem parte do cotidiano matutino televisivo, os desenhos educativos e entre outros. Em resumo, o mercado de animações com o intermédio dos *cartoons* é bastante promissor e a computação está diretamente ligada com este mercado.

Utilizando o conhecimento de Processamento Digital de Imagem (PDI) é possível fazer a adição de efeitos em imagens e vídeos digitais. Desta forma, este trabalho propõe a implementação de um algoritmo para geração de *cartoons* a partir de imagens e vídeos, utilizando elementos de filtragem espacial.

Palavras-chave: Processamento Digital de Imagem, *Cartoon.*

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

PDI	Processamento Digital de Imagem
RGB	Sistema de Cor aditivo, *Red, Green, Blue*
CMY	Sistema de Cor Subtrativo, utilizado para impressão *Ciano, Magenta, Yellow*
XYZ	Sistema de Cor Aditivo, utilizado para padronizar o espectro de cores
HSL	Sistema de Cor Aditivo, padrão por paleta cilíndrica
FPS	*Frames per second*, Quadros por segundo
JVM	*Java Virtual Machine,* Máquina Virtual Java
API	*Application Program Interface,* Interface de Programação de Interfaces
YIQ	Sistema de cor aditivo, adotado para imagens monocromáticas
GUI	*Graphical user interface*, Interface Gráfica de Usuário
KLT	*Kernel-Level Thread*, *Thread* de nível de Núcleo
CPU	*Central Processing Unit,* Unidade central de processamento
IDE	*Integrated development environment,* Ambiente integrado de desenvolvimento

SUMÁRIO

1. INTRODUÇÃO

Junto com o crescente avanço da tecnologia, o homem busca cada vez mais novas formas de entretenimento. Dentre elas pode-se citar a manipulação de imagens e vídeos digitais para criação de *cartoons* – desenhos em forma de caricatura, que tem bordas destacadas e com contornos em cores vivas, geralmente feito à mão com canetas ou lápis de desenho. Desde o século XIX, os *cartoons* estão presentes no cotidiano, seja em jornais, fotografias, revistas, notícias da imprensa e até em quadrinhos animados.

Nas últimas duas décadas o mercado de animações cresceu, não só pela capacitação de mão de obra, mas também quanto ao crescimento do potencial tecnológico utilizado, como o Processamento Digital de Imagem (PDI). Por exemplo, o Filme "*Frozen* Uma aventura Congelante" produzida pela *Disney*, somente no ano de 2013 arrecadou mundialmente cerca de US$ 1,27 bilhão e se tornou um dos maiores sucessos de todos os tempos em bilheteria (RUSSO, 2015).

As aplicações na área de PDI são muitas, são exemplos do uso de PDI: A remoção de ruído de uma fotografia antiga; A técnica de biometria para a autenticação de usuário; A identificação de um corpo celeste por meio de fotografia e extração da faixa *gamma* em uma imagem; O uso de filtros de raios x para melhor visualização de anomalias no corpo humano para a medicina. O interesse nos métodos de PDI vem de duas principais áreas: O processamento de dados de imagens; O melhoramento da informação visual para interpretação humana (GONZALES & WOODS, 2009).

A área de PDI estuda métodos genéricos como a filtragem espacial e a filtragem de intensidade, que são métodos para manipulação de fotos e vídeos digitais e se fazem necessárias para a criação do algoritmo genérico de *cartonização.* Este trabalho propõe a criação de um algoritmo de *cartonização*, que disponibiliza um método genérico para a criação de *cartoons*, baseando-se em imagens e vídeos digitais.

1.1 Problema

A produção dos *cartoons* como forma arte, depende quase que exclusivamente da mão de obra de um ser humano especialista com o uso de ferramentas como as mesas digitalizadoras para desenho, e este trabalho é demasiadamente demorado e complexo. Como o número de câmeras fotográficas profissionais e de smartphones existentes hoje em dia só tem crescido e também o uso de redes sociais para compartilhamento dessas mídias produzidas, pessoas sem talento algum para produzir arte procuram alguma forma de também a utilizar. A *cartonização* de imagens e vídeos digitais pode ser feita por essas pessoas. Este trabalho propõe construir um algoritmo que consiga gerar *cartoons* de forma genérica sobre imagens e vídeos digitais.

1.2 Justificativa

A área da animação na computação é uma das mais desafiadoras e complexas no que diz respeito a busca de novas tecnologias. O potencial de material para trabalhos nesta área só tem crescido com a introdução das redes sociais com o compartilhamento e o crescimento em massa das mídias digitais. A utilização deste algoritmo poderia ser feita através das demais ferramentas de edição de imagens e vídeos que redes sociais como *Instagram* e *Facebook* possuem, ou até mesmo a implementação em uma das grandes I*ntegrated Develpment Environment* (IDEs) para edição de fotografias e vídeos profissionais.

1.3 Objetivos

O objetivo deste trabalho é construir um algoritmo para cartonização de imagens e vídeos e posteriormente demonstrar os resultados obtidos, para isto é necessário:

- Definir como o algoritmo trabalha e define a padronização das imagens;
- Definir como o algoritmo trabalha e define a padronização dos vídeos;
- Expor a fundamentação do algoritmo de *cartonização*;
- Expor o processo de produção do algoritmo de cartonização;
- Expor trabalhos correlatos;
- Expor um estudo de caso para o algoritmo implementado;

1.4 Organização

Além deste capitulo introdutório, este trabalho é constituído de mais quatro capítulos. No **Capítulo 2**, é apresentado uma introdução ao PDI, abordando todos os aspectos necessários para a implementação do algoritmo, falando a respeito dos fundamentos de imagens digitais e dos fundamentos de vídeos digitais, assim como, abordar os métodos de filtragem espacial e as tecnologias utilizadas e também realizar uma breve explanação sobre trabalhos relacionados a este. No **Capítulo 3**, são mostradas todas as características do algoritmo e expõe-lo em pseudocódigo e fluxogramas. No **Capítulo 4**, apresenta uma discussão e demonstra os resultados obtidos através da ferramenta implementada. No **Capítulo 5**, expõe as considerações finais da solução proposta e direcionamentos que podem aprimorar a qualidade dos resultados e possíveis trabalhos futuros.

2. REFERENCIAL TEORICO E TRABALHOS CORRELATOS

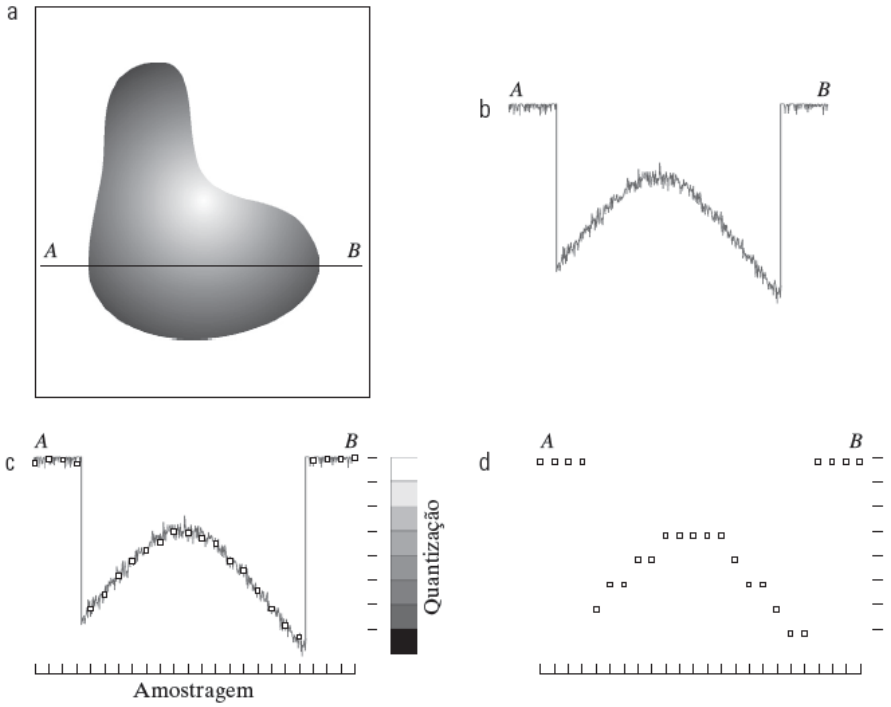
Este capítulo apresenta conceitos relacionados com imagens e vídeos digitais. Ele apresenta também conceitos relacionados com a luz, que se torna essencial para a captação, e fundamentação de imagens e vídeos. Além disso, são expostos alguns fundamentos de filtragem espacial e a apresentação de trabalhos relacionados com a área em estudo.

2.1 Fundamentos de Imagens Digitais

Uma imagem digital pode ser definida por uma função bidimensional discreta, *f (x, y)*, em que *x* e *y* são coordenadas espaciais no plano, e a amplitude de *f* é a intensidade ou o *pixel* da imagem no referido ponto. Os valores de *f* são quantizados de tal modo que a sua intensidade indica uma cor específica na imagem. A Figura 1 apresenta uma imagem e sua respectiva visualização como uma função discreta, onde em (a) foi definido um intervalo para exemplificar a quantização de uma imagem, em (b) mostra a função linear que representa a intensidade do pixel de forma continua, (c) mostra a como é realizado a quantização da imagem transformando da função continua para a função discreta, e (d) os pontos discretos da função A-B. (GONZALES & WOODS, 2009).

Para definir um sistema de cor é necessário responder algumas perguntas: Como se classificam as cores? Como o ser humano percebe as cores? Como representar as cores em um sistema numérico? (CASTLEMAN, 1996).

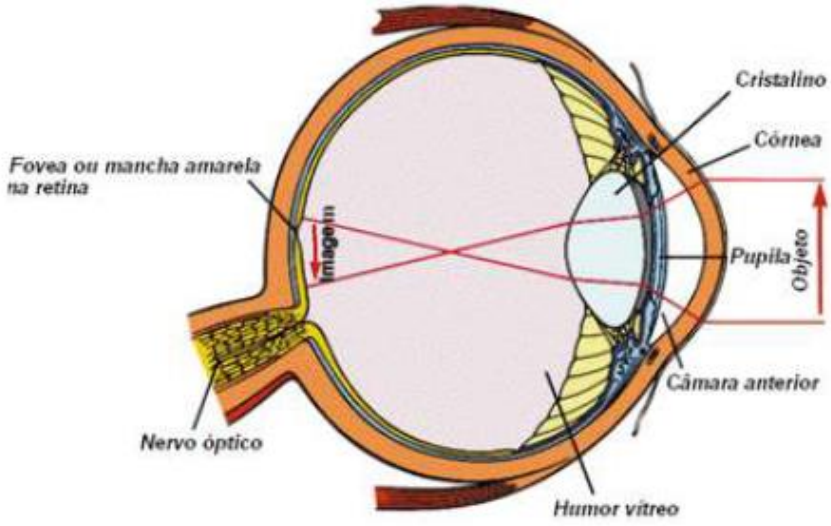
Figura 1: Quantização de uma Imagem Digital



Fonte: (GONZALES & WOODS, 2009)

O olho humano é similar à uma câmera fotográfica, sua função é converter os sinais de luz em impulsos nervosos, a Figura 2 mostra a anatomia do olho humano demonstrando onde a luz é refletida para que as células fóton-sensores sejam ativadas.

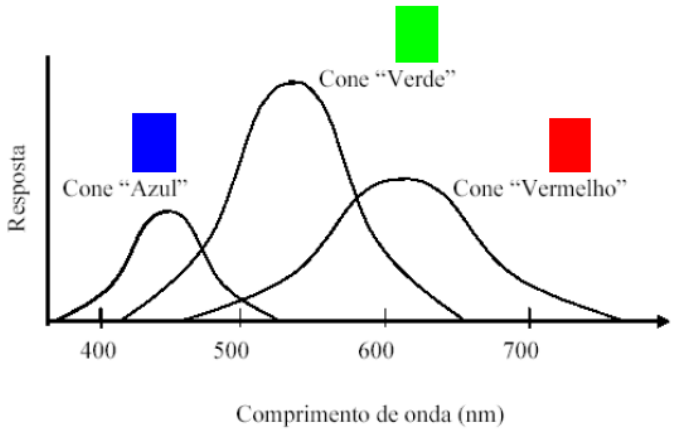
Figura 2: O olho humano.



Fonte: (GONZALES & WOODS, 2009)

A luz é uma onda eletromagnética que se propaga a 300.000km/s, pode ser calculada através da formula: $v = \lambda * f$, em que v é a velocidade, λ é o comprimento de onda e f é a frequência da onda. A cor é o resultado da percepção de diferentes comprimentos de onda da luz. As cores se propagam com diferente frequência para cada espectro de cor, de forma análoga, o olho humano responde aos impulsos nervosos de diferente forma para cada frequência luminosa (FILHO & NETO, 1999), como mostra a Figura 3, o ser humano na sua evolução considerou que cores verdes e vermelhas representassem vida, como o alimento, e com o passar das gerações o olho humano se adaptou a estas cores de forma diferente.

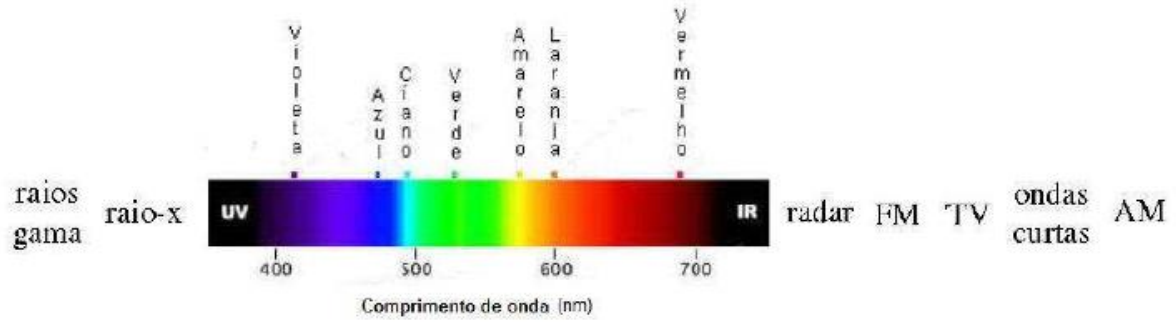
Figura 3: Comprimento de onda visível ao ser humano



Fonte: (FILHO & NETO, 1999)

O comprimento de onda do espectro visível compreende de 400nm até 700nm. A faixa de frequência visível compreende de 4.3x10^14 Hz até 7.5x10^14 Hz (CASTLEMAN, 1996). A Figura 4 mostra o espectro eletromagnético no comprimento de onda visível ao olho humano.

Figura 4: Espectro Visível



Fonte: (GONZALES & WOODS, 2009)

As propriedades da cor que os sistemas discretos utilizam, seguem o padrão de *Matiz, Saturação, Luminosidade:*

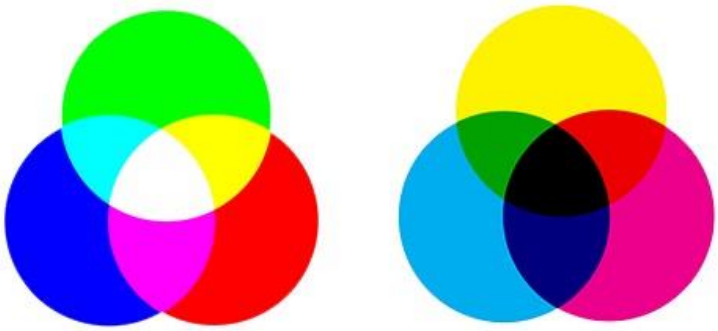
- *Matiz* ou *Hue:* atributo referente ao comprimento de onda dominante da cor.
- *Saturação* ou Intensidade: atributo referente à pureza da cor, ou seja, quantidade de luz branca incidente.
- *Luminosidade* ou *Brilho:* atributo referente à quantidade de luz da cor, ou seja, o seu aspecto claro ou escuro.

Um sistema de cor é um modelo que representa as propriedades das cores em contexto especifico. Os principais sistemas são: *RGB, CMY, XYZ, HSL, YIQ.* O espaço da cor *Gamut*: É universo de cores representado por um sistema de cor. A classificação do universo:

- Aditivo, quando os valores partem do 0 como valor mínimo e chegam ao 255 como valor máximo, tem efeito de adicionar, a soma de todas as cores chega na cor branca.
- Subtrativo, os valores partem do 255 como valor mínimo e chegam ao 0 como valor máximo, tem efeito de subtrair a cor da luz branca, pois, descrevem o que acontece com a cor quando a luz é refletida pela superfície de algum objeto.

A Figura 5 mostra o *gamut* de cores em um sistema aditivo e subtrativo a esquerda temos o RGB e a direita temos o CMY.

Figura 5 Sistemas de Cores Aditivos e Subtrativos



Fonte: Autoria Própria

O sistema de cor YIQ é um padrão de cor adotado para transmissão de cores em sistemas televisivos datado do início dos anos 90. O padrão de cor YIQ é excelente para representação de imagens monocromáticas. É possível transformar a imagem em RGB para YIQ pela formula matricial processando *pixel* a *pixel* pela Equação (2.1).

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.229 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.221 & -0.523 & 0.312 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.1)$$

2.2 Fundamentos de Vídeos Digitais

Um vídeo digital é um sistema de reprodução, armazenamento e transmissão de sucessivas imagens que dão efeito de movimento, as quais podem estar acompanhadas de sons. A estrutura de um vídeo é composta de duas resoluções:

- A resolução espacial a que diz respeito da resolução geométrica e a resolução de cor.
- A resolução temporal a que diz respeito do número de quadros/*frame* por segundo (FPS).

Para implementação deste trabalho, vídeo é uma sucessão de imagens de mesmo tamanho com mesma resolução geométrica e de mesma resolução de cor, onde é transformado em um *array* de inteiros positivos, para posteriormente ser processado. A Figura 6 demonstra como os *frames* são fatiados em uma sequência de imagens, a quantidade de *frames* que o vídeo possuí em uma determinada fração de segundo caracteriza seu *framerate*.

Figura 6: *FrameRate* Vídeo

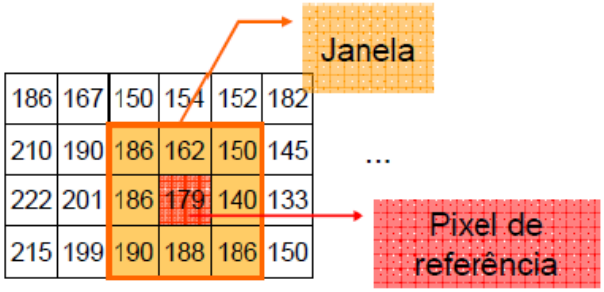
Fonte: frame-rate MediaCollege.com (2018)

2.3 Filtros

O termo filtro foi trazido do processamento de sinais no domínio da frequência, no qual a tarefa de filtragem significa aceitar ou deixar passar certos componentes de frequência (GONZALES & WOODS, 2009). Em PDI, técnicas de filtragem são transformações da imagem pixel a pixel, que não dependem apenas do nível de cinza de um determinado pixel, mas também do valor dos níveis de cinza dos pixels vizinhos. O processo de filtragem é feito utilizando matrizes denominadas máscaras ou janelas, as quais são aplicadas sobre a imagem. Exemplo de máscara 3x3 aplicada ao pixel 179 da Figura 7.

A aplicação da máscara com centro na posição (x, y) no caso o *pixel* 179, sendo "*x*" o número de uma dada linha e "*y*" o número de uma dada coluna sobre a imagem, consiste na substituição do valor do *pixel* na posição (x, y) por um novo valor que depende dos valores dos *pixels* vizinhos e dos pesos da máscara, por exemplo com o somatório de todos os valores dividido pelo total dos elementos da máscara, gerando uma nova imagem com a eliminação das linhas e colunas iniciais e finais da imagem original. Os filtros espaciais podem ser classificados em passa-baixa, passa-alta ou passa-faixa. Os dois primeiros são os mais utilizados em processamento de imagens. O filtro passa-faixa é mais utilizado em processamentos específicos, principalmente para remover ruídos periódicos.

Figura 7: Mascara ou Janela de Filtragem



Fonte: (GONZALES & WOODS, 2009)

2.3.1 Filtragem de Intensidade

São filtros denominados filtros de intensidade, aqueles que modificam somente o valor do *pixel f (x, y),* e não modificam a posição de um dado *pixel* em relação a sua imagem original, não gerando deformações na imagem no que diz respeito a posição. São exemplos de filtros de intensidade:

- Filtro de negativo, sua função é multiplicar por -1 todos os *pixels* da imagem invertendo suas cores pelo espaço da cor, logo onde é branco se torna preto, e onde é preto se torna branco. A Figura 8 mostra o filtro de negativo em ação (GONZALES & WOODS, 2009).
- Filtro de tons de cinza multiplica a entrada da cor pela transformação em YIQ como será visto posteriormente. A Figura 9 mostra o resultado do Filtro de Tons de Cinza em ação (GONZALES & WOODS, 2009).

Figura 8: Filtro Negativo

(a) Imagem Original (b) Filtro Negativo

Fonte: Autoria Própria

Figura 9: Filtro Tons de Cinza

(a) Imagem Original (b) Filtro Tons de Cinza

Fonte: Autoria Própria

2.3.2 Filtragem no Domínio Espacial

São filtros denominados filtros espaciais, aqueles que modificam diretamente o valor do *pixel f (x, y)*, tomando por base os valores dos *pixels* em sua vizinhança, com o uso do mascaramento, esta técnica atua manipulando diretamente os valores dos *pixels* na imagem. Esta técnica é computacionalmente mais eficiente e requer menos recursos de processamento para a sua realização se comparadas com as filtragens no domínio da frequência.

As operações no domínio espacial, são baseadas no escopo da ação, podem ser classificadas como:

- Pontuais, cada *pixel* na imagem de saída depende única e exclusivamente do *pixel* de origem, mapeando diretamente da imagem de entrada para imagem de saída.
- Locais, são as máscaras resumidamente, nas operações locais, o valor de saída de um *pixel* em determinada coordenada dependerá dos valores de entrada da sua localização e também dos seus *pixels* vizinhos (CASTLEMAN, 1996).

2.3.3 Filtros Lineares

Funcionam por meio de vizinhanças escolhendo um determinado ponto (*x*,*y*), onde *x* representa a linha e *y* a coluna. Este filtro destaca-se por alterar a imagem por meio de uma função linear, onde pode-se alterar todos os valores dos *pixels* pela determinada função. Esse filtro cria uma nova imagem excluindo as linhas e colunas sem diminuir a resolução.

A filtragem linear baseia-se em dois conceitos, o da convolução e o da correlação, em que convolução é a forma como a máscara é alternada dentro da imagem, este processo visa calcular a soma dos pontos de cada posição, ou seja, convolução é o processo de

deslocamento do filtro. Já a correlação é mostrada na Equação (2.2). Já a convolução é apresentada na Equação (2.3), sendo que a única diferença é o fato de girar em 180° o primeiro filtro (GONZALES & WOODS, 2009),

$$g(x,y) = M(x,y) \circ f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} M(s,t) f(x+y, y+t) \tag{2.2}$$

$$g(x,y) = M(x,y) \bullet f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} M(s,t) f(x-y, y-t) \tag{2.3}$$

em ambas as equações, *M(s, t)* é a máscara aplicada, e a função *f(x, y)* é o plano da imagem e pode-se observar a rotação em 180° pela inserção do sinal de menos em *s* e *t*. Nesta classificação de filtros, encontram-se os filtros lineares de suavização (filtros de média e filtros passa-baixa) servem para suavizar os *pixels* das imagens através da aplicação de uma máscara para calcular a média entre os *pixels* vizinhos por toda a imagem (*pixel*-a-*pixel*).

2.3.4 Operador de *Prewitt*

O filtro de *Prewitt* ou operador de *Prewitt* é um filtro passa-alta e tem por objetivo eliminar as baixas frequências e atenuar as altas frequências, e suas características são de realçar o contraste e destacar as bordas, linhas, curvas e manchas na imagem. Em filtragem passa-alta os componentes de alta frequência não são alterados, enquanto os de baixa frequência são removidos. Isto faz com que os detalhes finos da imagem sejam enfatizados. Tecnicamente, trata-se de um operador da diferença, computando uma aproximação do gradiente da função de intensificação da imagem. Em cada ponto da imagem, o resultado do filtro de *Prewitt* é, ou o vetor gradiente, ou a norma deste vetor (GONZALES & WOODS, 2009).

O filtro de *Prewitt* baseia-se em convolver uma imagem com uma filtragem pequena separando nas direções vertical e horizontal, e tem um custo computacional relativamente baixo. Matematicamente, o filtro usa duas mascaras 3x3 que são convolvidas com a imagem original para calcular aproximações das derivadas, uma para vertical e outra para horizontal como mostra a Equação (2.4),

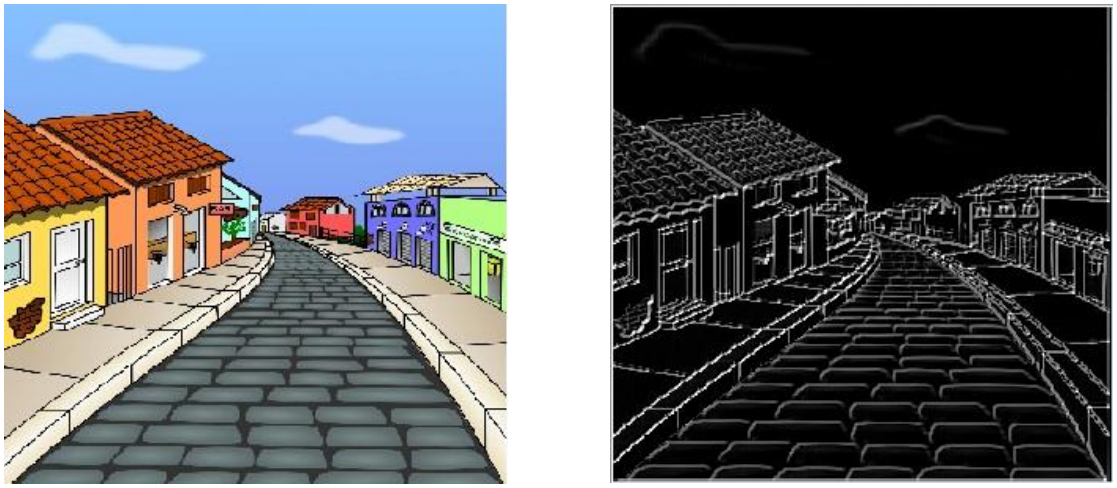
$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} e \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.4}$$

para calcular o resultado do *pixel* para o operador de *Prewitt*, usa se a Equação (2.5),

$$P(x,y) = \sqrt{px^2 + py^2} \tag{2.5}$$

em que *px* é o resultado do *pixel* para a máscara vertical e *py* é o resultado do *pixel* para a máscara horizontal. O resultado da operação de Prewitt pode ser vista na Figura 10.

Figura 10: Operador de *Prewitt*



(a) Imagem Original (b) Filtro de Prewitt

Fonte: Autoria Própria

2.4 Trabalhos correlatos

NARADRAN (Narandran, 2013) propôs a conversão de uma imagem em *cartoon* para que o resultado se parecesse com uma imagem pintada a mão. Neste trabalho, dois processos independentes são aplicados à imagem de entrada: a detecção de bordas, e o processamento em passa-baixa para suavização dos contornos. Os resultados dos dois processos são combinados para a geração do resultado final. Para detecção de bordas na imagem foram utilizados vários filtros: *Canny Edge*, *Operador* de *Sobel*, e *Diferença* Gaussiana. Para a suavização foram utilizados os filtros: *Bilateral Filter* e *Mean Shift Filter*. O algoritmo foi feito para possibilitar o chaveamento entre os filtros passa-alta e passa-baixa obtendo vários resultados distintos.

K. DADE *(Kevin 2012)* propôs para seu trabalho produzir a partir de fotografias e objetos *3D* tiradas por um *smartphone Android* imagens *cartonizadas.* O pré-processamento tanto dos objetos *3D* quanto das fotografias são pré-processadas para a padronização da entrada, através de um *array* de *pixels*. Este *array* de *pixels* é processado paralelamente de duas formas distintas:

- Detecção de bordas como pré-processamento para a detecção de bordas o filtro da mediana com mascaramento de 7x7 é passado sobre o *array* de *pixels* para filtrar possíveis borrões na imagem e priorizar as bordas, o passo seguinte é a detecção utilizando o filtro *Canny Edge*.
- A suavização é feita através do *Bilateral Filter* com um mascaramento de 9x9 iterado 14 vezes para dar a suavização necessária, o *Bilateral* Filter tem um efeito de suavização continua e iterativo, ou seja, pode reprocessar a mesma entrada *n* vezes e o seu resultado é uma suavização linear preservando as bordas e seus contornos. O

processo finaliza recombinando a saída do *Canny Edge* com a saída do *Bilateral Filter* produzindo um excelente resultado.

WINNERMÖLLER H.; OLSEN C.; GOOCH B. (*Holger, Sven, Bruce 2008*) demonstraram um algoritmo capaz de gerar *cartoons* em tempo real. Este trabalho divide o *frame* em dois tipos de processamento, um para o filtro passa-alta e outro para o filtro passa-baixa. O filtro passa-alta é encarregado de criar as bordas, foi utilizado o filtro da *Diferença Gaussiana* para criação dos contornos. O filtro passa-baixa utilizado neste trabalho foi o *Bilateral Filter* iterativo e posteriormente utiliza-se uma quantização da luminância para homogeneização das cores. O processo é finalizado recombinando as partes.

JOHN C. (Canny *1086*) este artigo descreve uma abordagem computacional para criação da detecção de borda. O processo de detecção de bordas de *Canny* baseia-se em dois critérios básicos de desempenho, i.e., os critérios de detecção e localização. Estes critérios estão sujeitos ainda a um terceiro, conhecido como injunção de resposta múltipla, que força o processo a detectar uma única borda onde existe somente uma borda verdadeira. O principal objetivo do trabalho de *Canny* é o desenvolvimento de um detector ótimo para o tipo de bordas mais comum em imagens digitais, i.e., as bordas tipo degrau. Uma das principais constatações de *Canny* é que o operador ótimo encontrado é muito semelhante à função gerada pela primeira derivada da função Gaussiana. *Canny* também propôs um processo de afinamento de bordas conhecido como supressão não máxima e um outro processo conhecido como histerese, cuja função é a de eliminar a fragmentação das bordas causada pelo ruído da imagem.

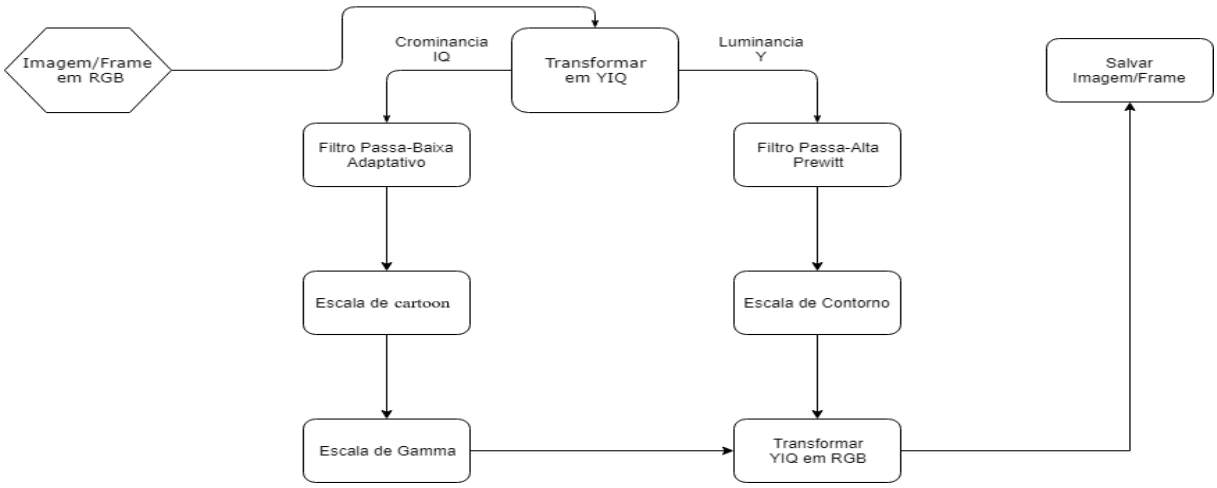
3. O ALGORITMO DE *CARTONIZAÇÃO*

Este capitulo é divido em duas partes, o tratamento para Imagem e o tratamento para vídeo:

- O processamento da imagem, o algoritmo trabalha com a luminância e crominância e não com os valores das cores propriamente dita em RGB, assim é necessário como pré-processamento transformar a imagem de RGB para YIQ. O algoritmo trabalha com o valor da luminância da imagem para calcular as bordas nos contornos com *Prewitt*, e a crominância da imagem para calcular as escalas de *cartoon* e escala de *gamma*. Todo o processamento do algoritmo segue o diagrama da Figura 11.
- O processamento de vídeo, o algoritmo trabalha fatiando um vídeo de entrada em uma sequência de *frames*, com o objetivo de processar *frame* a *frame* todo o vídeo para posteriormente recombinar os frames e renderizar o vídeo.

O processo para imagem começa com a divisão em dois processos concorrentes, uma parte processa o filtro passa-alta para detecção de bordas, e a outra parte que processa o filtro passa-baixa para suavização e quantização da imagem. O filtro passa-alta é processado pelo operador de *Prewitt* no qual se obtém as bordas através de um quantizado que é chamado de **escala de contorno**, variando este valor pode-se obter mais ou menos borda no resultado final. O filtro passa-baixa trata de suavizar a imagem ou *frame* para limpar possíveis ruídos, primeiramente é processado através do filtro passa-baixa adaptativo no qual é o filtro da media 3x3 que varia para cada janela como entrada, depois utiliza-se da quantização do *gamut* de cores para o processamento da **escala de *cartoon***, variando este valor pode-se obter mais ou menos cores no espectro do resultado, posteriormente é passado um filtro de *gamma* para clarear ou escurecer o resultado da escala de *cartoon*, este processo é chamado de **escala de gamma**. O processo finaliza com a combinação do filtro passa-alta e o filtro passa-baixa transformando o YIQ em RGB.

Figura 11: Algoritmo de Cartonização



Fonte: Autoria Própria

Após transformar a imagem para YIQ o algoritmo busca encontrar as bordas através do operador de *Prewitt*. No *pixel* cuja máscara está sendo operada por *Prewitt*, utiliza-se a mesma máscara para calcular a média do valor de luminância desta máscara para posteriormente ser utilizada como parâmetro para escala de **contorno**. Após calcular a média, calcula-se a variância, V, para a mesma máscara de *Prewitt* pela Equação (3.1),

$$V = \sum_{n=1}^{9} (b[x,y] - m(n))^2 \tag{3.1}$$

em que $b[x,y]$ é o valor calculado pelo operador de *Prewitt*, e m é o valor mediano dessa borda na posição da máscara em *n*. O cálculo do desvio padrão global que é realizado pela Equação (3.2) será posteriormente utilizado para definir a escala de contorno,

$$Sd = \sqrt{\left(\frac{V}{(x-2)*(y-2)}\right)} \tag{3.2}$$

o V é variância dos *pixels* da imagem na máscara utilizada, *x* e *y* são os valores da altura e largura respectivamente da imagem ou *frame*. Neste ponto o usuário entra com os valores de contorno é **c**, *gamma* é G e *cartoon* é Ca, no qual estes valores serão modificados em tempo de execução e serão passados para o algoritmo através das seguintes Equações (3.3), (3.4) e (3.5),

$$C = (c' + 10)/10.0 \tag{3.3}$$

o resultado C é o valor que o algoritmo 1 irá utilizar na verificação das bordas, o valor c' é o valor introduzido em tempo de execução para o algoritmo, sua implicação para o resultado é a adição de mais ou menos bordas,

$$G = (g' + 10)/10.0 \tag{3.4}$$

o resultado G é o valor que o algoritmo irá utilizar na Equação (3.6), o valor g' é o valor introduzido em tempo de execução do algoritmo, sua implicação para o resultado é a modificação da intensidade luminosa, clareando ou escurecendo a imagem,

$$Ca = (d')^2 \tag{3.5}$$

o resultado Ca é o valor que o algoritmo 1 irá utilizar para modificar os parâmetros de *cartonização* na Função Aplicar Escala de *Cartoon*, a Equação (3.7) demonstra como este valor é aplicado no algoritmo, o valor d' é o valor introduzido em tempo de execução do algoritmo, sua implicação para o resultado é a modificação da representação do número de cores possíveis no resultado,

$$Eg = (\frac{cO}{255})^{Ga} * 255 \tag{3.6}$$

$_{Eg}$ representa o resultado do algoritmo para cada canal de cor, e $_{Ga}$ é o valor da Escala *Gamma* introduzida pelo usuário em tempo de execução,

$$Ec = cO\%Ca \tag{3.7}$$

em que valor $_{Ec}$ é divido para cada canal de cor, uma para vermelho, verde, azul. Os valores de $_{cO}$ são os valores do *pixel* da imagem original processada pelo filtro Passa-Baixa adaptativo dado na equação (3.8),

$$m = (cO + m3p3)/9 \tag{3.8}$$

em que o resultado $_{m}$ é o valor da cor no *pixel* para cada canal uma para vermelho, verde, azul após ser processado por um filtro da média 3 por 3, a $_{m3p3}$ é demonstrada na sessão Filtros do capitulo 2.

Algoritmo 1: Escala de Contorno

```
1:    Para toda Imagem Repita
2:        Se (bordas[x,y] < (media+contorno*Sd))
3:        {
4:            Aplicar Escala de Cartoon no pixel
5:            Aplicar Escala de Gamma no pixel
6:        }
7:        Senão
8:            Aplicar cor preta no pixel
9:    Fim repita
10:   Fim repita
```

Finalizando o processo salvando a imagem ou *frame* em um arquivo *jpeg*.

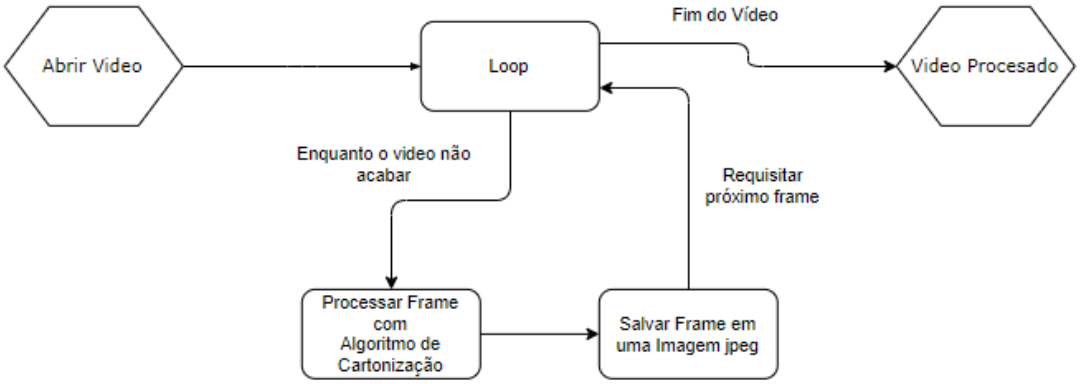
3.1 Vídeos

Um vídeo é uma sequência de imagens a uma taxa constante de variação, logo para transformar um vídeo é necessário definir todos os parâmetros que o vídeo original adota, como sua taxa de *bits*, sua taxa de *frames* por segundo, o tamanho do *frame.* Todos estes valores são definidos no momento da abertura do vídeo onde o algoritmo é capaz de converter um *frame* de um vídeo em uma imagem, como é demonstrado na Figura 12 e Algoritmo 2.

Algoritmo 2: Convertendo vídeo em sequencias de imagens

1: Após carregar o vídeo desejado na memória, o *FFMPEG* calcula o número de *frames* que o vídeo possui = *f*
2: **Repita p=1; p<f+1; f++**
3: Capturar *frame* p
4: Processar *frame* p com o Algoritmo de *Cartonização*
5: Salvar *Frame* em imagem *jpeg*
6: Fim repita

Figura 12: Converter Vídeo em Sequencias de imagens



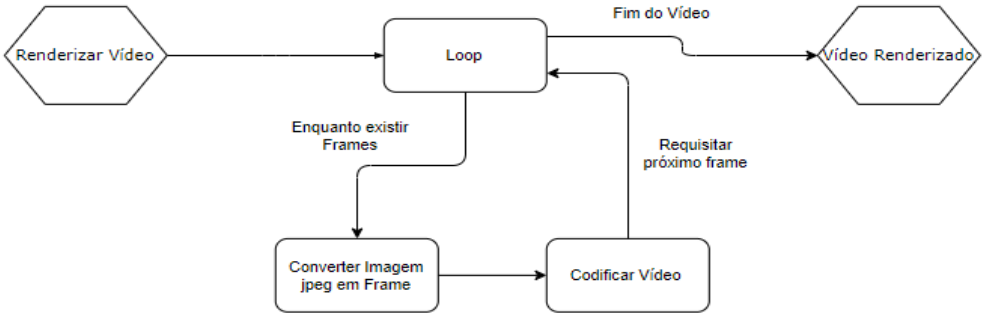
Fonte: Autoria Própria

Após o Algoritmo 2 percorrer todo o vídeo, é criado um diretório qualquer para armazenar todas as imagens no formato *jpeg* com o *frame* processado. Para converter a sequência de imagens em um vídeo, é necessário conhecer o tamanho do vídeo em número de *frames* e *frames* por segundo, com estes dados é calculado o tempo do vídeo exatamente como o original para não possuir cortes. Tempo = *frames*/*frames* por segundo, o resultado é dado em segundos. Utiliza-se o Algoritmo 3 para converter a sequência de imagens no vídeo processado, a Figura 13 demonstra este processo.

Algoritmo 3: Converter sequência de imagens em vídeo

1: Inicializa valores de NumFrames = número de *frames* e Fps = *frames* por segundo, carrega o *frame* i da imagem correspondente no diretório anteriormente definido na imagem a ser codificada.
2: **Repita i=1; i<NumFrames; i++**
3: Codificar vídeo com entrada do algoritmo o *frame* i
4: **Fim repita**
5: Salvar Resultado do algoritmo 4 em um arquivo mp4

Figura 13: Converter Sequência de imagens em vídeo



Fonte: Autoria Própria

4. SIMULAÇÃO E RESULTADOS

Este capítulo aborda os tópicos de simulação e resultados, no qual o algoritmo implementado foi testado para obter valores de desempenho. Um estudo de caso foi idealizado para o algoritmo no qual foi desenvolvido uma interface gráfica para visualização dos resultados, e uma sessão de resultados para discutir sobre os parâmetros utilizados e os resultados obtidos.

4.1 Simulação

A simulação deste algoritmo foi implementada em *Java* e para isto, este trabalho exige incorporação de algumas tecnologias que se fazem necessárias, como:

- A subclasse *BufferedImage* que herda todos os atributos da classe *Image* que o *Java* implementa, com a diferença que cria um *buffer* em memória da informação contida na imagem e é possível modifica-la em tempo de execução os valores dos *pixels*, o tamanho da imagem, o tipo de rasterização, o tipo da cor utilizada, ou seja, todas as ferramentas necessárias para utilizar neste trabalho.
- O *Java Thread* que é um conjunto de bibliotecas utilizada no *Java*, serve para criar caminhos paralelos sobre um algoritmo sequencial e delegar um dado conjunto de código a um outro núcleo de processamento simultaneamente, caso exista.
- A biblioteca *FFMPEG* que é um conjunto de bibliotecas de *software* que tem a capacidade de trabalhar com arquivos multimídia, pode ser utilizado para codificar vários tipos de arquivos de multimídia, como fotografias nas extensões: *jpeg*, vídeos em *mp4*, *mpeg*, *flv*, e entre outros e também *stream* de áudio e vídeo, esta biblioteca pode ser utilizada para converter vídeo em uma sequência de fotografias, e vice-versa.

Esta sessão apresenta os resultados obtidos com o algoritmo desenvolvido. Diante destes resultados foi criado uma tabela com a variação de resolução e tempo de processamento para várias entradas distintas. Foi utilizado um computador Intel i5-2310 2.911~*GHz 8gb RAM DDR3 1600MHz,* com versão 1.8 do *JDK* do *Java. S*endo o algoritmo sequencial, só é utilizado um núcleo para processamento, e através disto foi obtido estes valores como é demonstrado na tabela a seguir:

Tabela Resultados: Tempos do algoritmo

Resolução	Imagem: Borboleta	Imagem: Pôr do Sol	Imagem: Praia	Imagem: Mulher
SD (640x480)	320ms ~ 413ms	315ms ~ 410ms	335ms ~ 420ms	321ms ~ 417ms
HD (1280x720)	650ms ~ 732ms	632ms ~ 741ms	648ms ~ 751ms	643ms ~ 725ms
FHD (1920x1080)	1,33s ~ 1.4s	1.31s ~ 1.39s	1.32s ~ 1.41s	1.32s ~ 1.37s
UHD (3840x2160)	2.2s ~ 2.31s	2.2s ~ 2.32s	2.2s ~ 2.21s	2.2s ~ 2.24s

Fonte: Autoria Própria

A variação das imagens não interfere muito no resultado do tempo e sim no tamanho, sendo o algoritmo na notação assintótica de *O* ele é: $O(m+p)$, onde m é o número de linhas e p o número de colunas na imagem, é necessário percorrer de forma linear toda a imagem ocasionando assim um custo moderado para imagens de alta-resolução, logo a idealização da paralelização deste algoritmo para *(m+n)/p* onde *p* é o número de processadores concorrentes, poderá chegar a uma notação $O(log_p(m+n))$, e fazendo assim teoricamente o algoritmo 80% mais rápido que este patamar não perdendo performance com os métodos bloqueantes.

Métodos bloqueantes são aqueles que fazem uso de uma variável global, quando duas ou mais *threads* utilizam a mesma variável, pode ocorrer problema de sincronia de dados, logo é necessário bloquear um método para que outro não acesse até que tenha certeza que estes valores são corretos.

4.2 Estudo de Casos

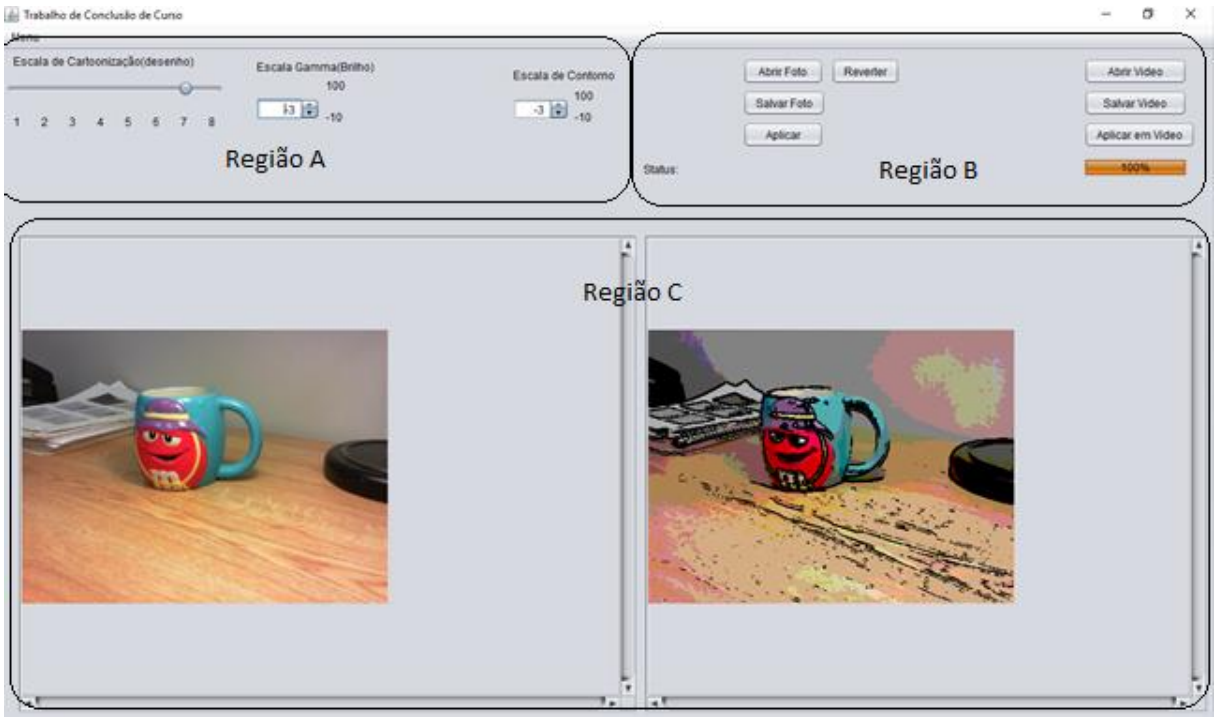
Uma interface gráfica foi desenvolvida para testar resultados e demonstrar os valores propostos para o algoritmo, ela é composta de 3 regiões distintas, o menu de ações, onde é possível abrir imagens e vídeos, A região dos Parâmetros, onde é possível modificar as variáveis do programa contendo a escala de *cartonização*, a escala de *gamma* e a escala de contorno. A região de visualização é compreendida de duas divisões, a primeira parte do lado

esquerdo onde tem uma visualização da imagem original, e a segunda parte do lado direito que tem uma pré-visualização do resultado, como é visto na Figura 15.

A interface foi idealizada para fácil manuseio, a criação foi dividida em 3 regiões distintas como é visto na Figura 14:

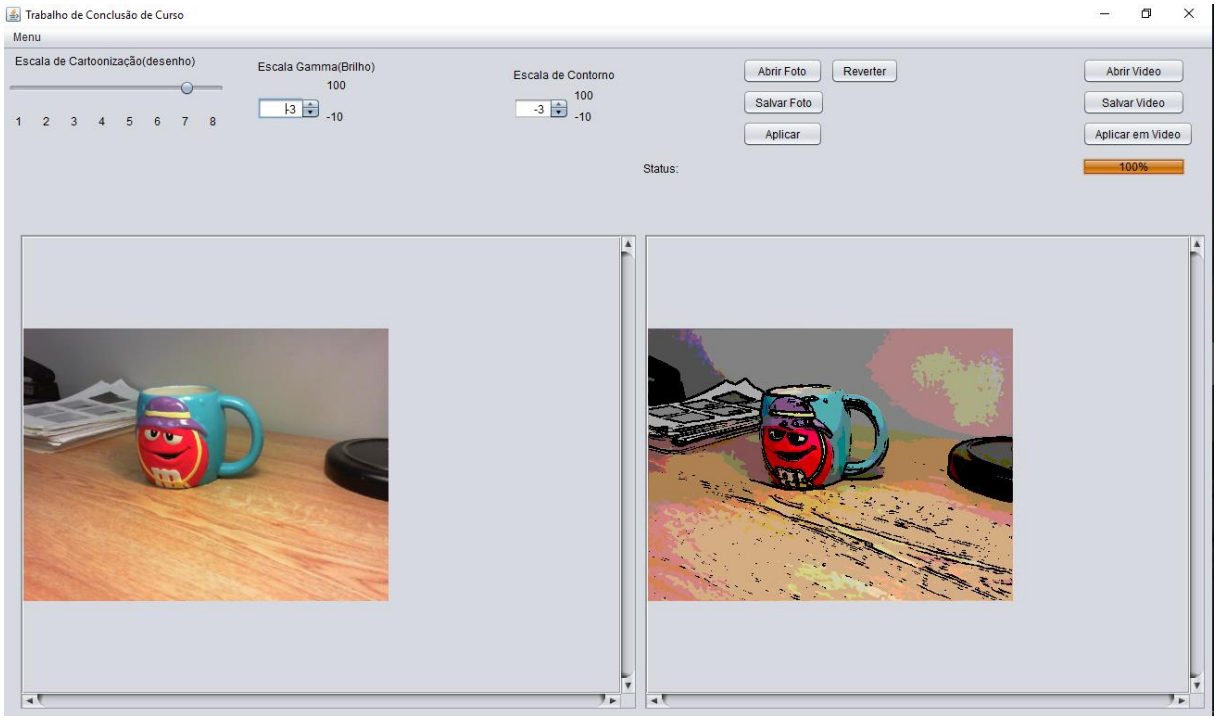
- Região A; o menu de modificação para modificar os valores em tempo de execução para o algoritmo.
- Região B; o menu de ações, para abrir, modificar caminho e salvar os arquivos renderizados.
- Região C; o menu de visualização para comparar as imagens originais com os resultados obtidos.

Figura 14: A Interface Gráfica



Fonte: Autoria Própria

Figura 15: A Interface Gráfica



Fonte: Autoria Própria

4.3 Resultados

Esta sessão apresenta os resultados obtidos com a aplicação desenvolvida e demonstrada na sessão anterior. O programa tem 3 parâmetros para modificação:

- A escala de ***cartoon*** *(Ca),* que diz respeito a paleta de cores que a saída do algoritmo vai possuir, seus valores oscilam de 1 a 8, onde 1 a escala detém o maior número de cores na paleta e 8 detém o menor valor da escala.
- A escala de ***Gamma*** *(Ga)* que diz respeito a modificação de intensidade de luminosidade dos *pixels* onde é controlado a intensidade do branco na saída, seus valores oscilam entre -10 como valor mínimo que condiz para a cor mais brilhante e +100 para o valor máximo e condiz com as cores mais escuras.
- A escala de ***contorno*** (*Co*) que controla a intensidade das bordas do desenho seus valores oscilam entre -10 e +100, onde -10 é o número máximo de linhas de contorno e +100 é o valor mínimo.

Os resultados seguintes serão mostrados sobre uma escala definida como *CgCo* (*Ca, Ga, Co*), onde os valores de *Ca* correspondem a escala de *cartoon*, *Ga* correspondem a escala de *Gamma*, e *Co* correspondente com a escala de contorno.

4.3.1 Borboleta

Esta sessão demonstra os resultados obtidos variando as escalas anteriormente mencionadas. A Figura 16 é utilizada para demonstrar a variação das escalas, ela foi escolhida, pois existe uma alta nitidez e alta variação de cores, ser paisagem e um alto contraste.

Figura 16: Borboleta

Fonte: blue-morpho-butterfly PublicDomainPictures.net (2018)

Figura 17 Resultado obtido com os parâmetros: ***CgCo:***(1, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Imagens com muita nitidez são ótimas para destacar as bordas, pois o algoritmo prioriza as diferenças bruscas de cor como é visto na Figura 17.

Figura 18 Resultado obtido com os parâmetros: ***CgCo:*** (2, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Figura 19 Resultado obtido com os parâmetros: ***CgCo:*** (3, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

As Figuras 17, 18, 19 e 20 demonstram como a escala de *cartoon* trabalha modificando significativamente a cada nível da escala. Variações sutis estão ocorrendo na imagem como pode ser vista a medida que o valor de cartonização é aumentado a quantidade de cores que a imagem possui é diminuída.

Figura 20 Resultado obtido com os parâmetros: ***CgCo:*** (4, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Figura 21 Resultado obtido com os parâmetros: ***CgCo:*** (5, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Como é evidenciado através da Figura 21, a partir do valor 5 na escala de *cartonização* imagens coloridas começam a se parecer mais com *cartoons* devido à baixa quantização de cores.

Figura 22 Resultado obtido com os parâmetros: ***CgCo:*** (6, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

As Figuras 22 e 23, mostram a ação da Escala de Cartoon, mudanças sutis são evidenciadas quando a escala de *cartoon* é aumentada.

Figura 23 Resultado obtido com os parâmetros: ***CgCo:*** (7, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Bons resultados também são encontrados modificando a escala de *Gamma* como é evidenciado na Figura 23 em comparação com a Figura 25.

Figura 24 Resultado obtido com os parâmetros: ***CgCo:*** (8, 0, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

A escala máxima de *cartonização* é evidenciada na Figura 24 e se a semelha muito com uma imagem desenhada ou pintada.

Figura 25 Resultado obtido com os parâmetros: ***CgCo:*** (7, -9, 0)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

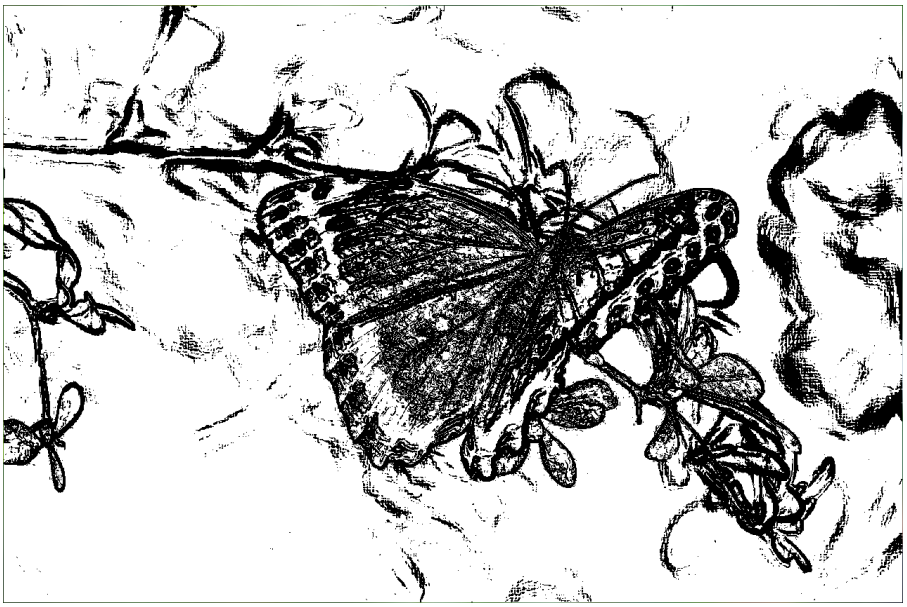
A variação da escala de *gamma* é bastante nítida na Figura 25, com a mesma escala de *cartonização* comparada com a da Figura 23.

Figura 26 Resultado obtido com os parâmetros: ***CgCo:*** (6, -6, -4)

Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Pode-se obter variações de cores modificando a escala de brilho e contorno com uma mesma entrada como mostra a Figura 26.

Figura 27 Resultado obtido com os parâmetros: ***CgCo:*** (1, -10, -10)



Fonte: Modificado de blue-morpho-butterfly PublicDomainPictures.net (2018)

Como é visto na Figura 27, pode-se encontrar fotos monocromáticas através da introdução do valor máximo para escala de *Gamma*, pois com este valor todas as cores são levadas no branco.

4.3.2 Pôr do sol

A Figura 28, pôr do sol foi escolhida para uso, pois é o oposto da imagem fonte da sessão anterior. Pois possui baixo contraste, pouca iluminação, baixa nitidez e pouca variação de luz.

Figura 28: Pôr do Sol

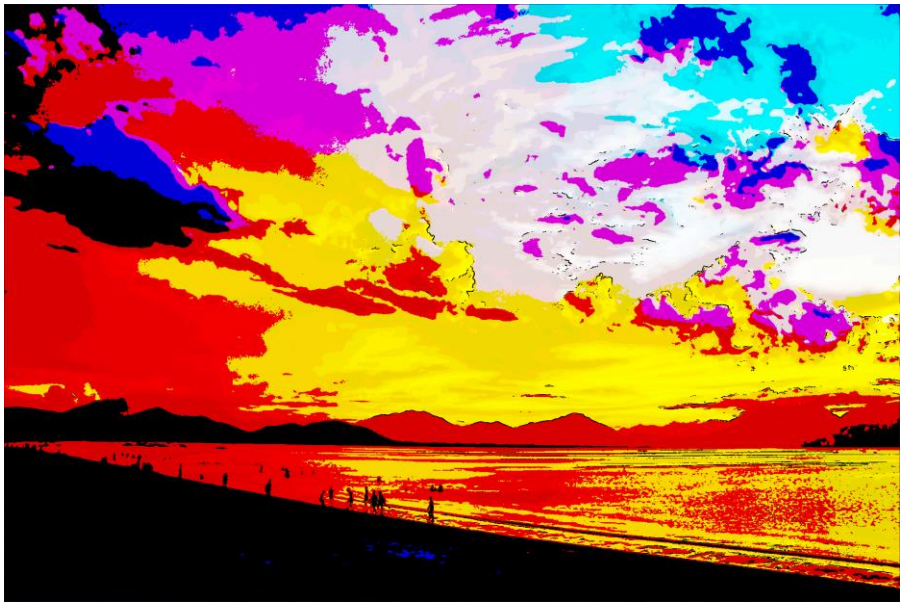
Fonte: burning-sky PublicDomainPictures.net (2018)

Figura 29 Resultado obtido com os parâmetros: ***CgCo:*** (1, 0, 0)

Fonte: Modificado de burning-sky PublicDomainPictures.net (2018)

Imagens com baixo contraste não evidenciam as faixas de contorno criadas pelo algoritmo como é visto na Figura 29.

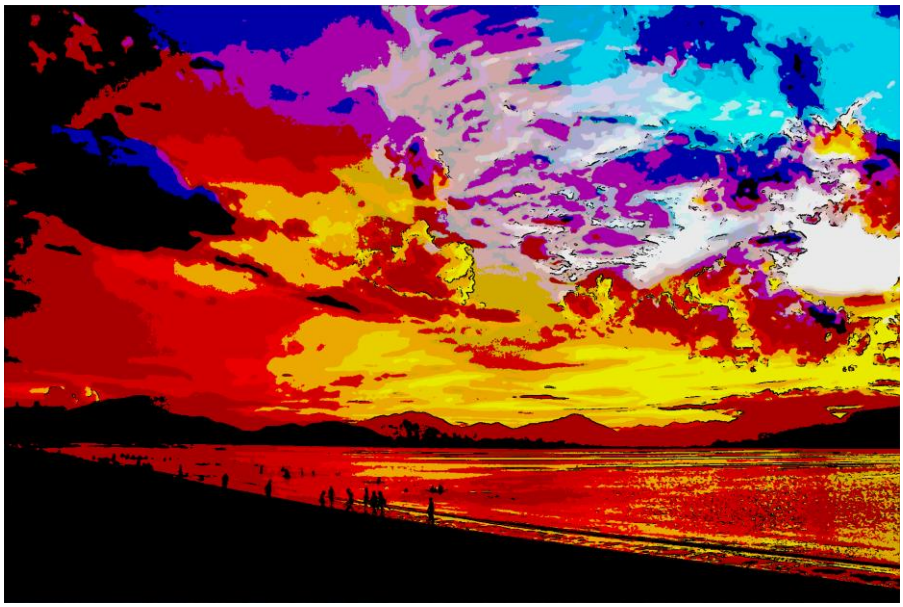
Figura 30 Resultado obtido com os parâmetros: ***CgCo:*** (8, -9, 40)



Fonte: Modificado de burning-sky PublicDomainPictures.net (2018)

Ótimos resultados podem ser encontrados com a modificação da escala de *Gamma* para imagens com baixa variação do contraste, como é demonstrado nas Figuras 30 e 31.

Figura 31 Resultado obtido com os parâmetros: ***CgCo:*** (8, -7, 20)



Fonte: Modificado de burning-sky PublicDomainPictures.net (2018)

Figura 32 Resultado obtido com os parâmetros: ***CgCo:*** (1, 10, 20)

Fonte: Modificado de burning-sky PublicDomainPictures.net (2018)

As Figuras 32 e 33 mostram outros resultados encontrados com a mesma figura fonte.

Figura 33 Resultado obtido com os parâmetros: ***CgCo:*** (1, -7, 0)

Fonte: Modificado de burning-sky PublicDomainPictures.net (2018)

Como visto anteriormente, pode-se encontrar uma alta variação de resultados modificando os valores da Escala de *Gamma* próximas ao valor máximo e diminuindo a paleta de cores com a escala de *Cartoon* como é visto na Figura 33.

4.3.3 Praia

A Figura 34, Praia foi escolhida por ter um objeto focal no centro, alta nitidez e baixa variação de contraste.

Figura 34 Praia

Fonte: hat-and-flip-flops-on-the-beach PublicDomainPictures.net (2018)

Figura 35 Resultado obtido com os parâmetros ***CgCo:*** (7, -8, -10)

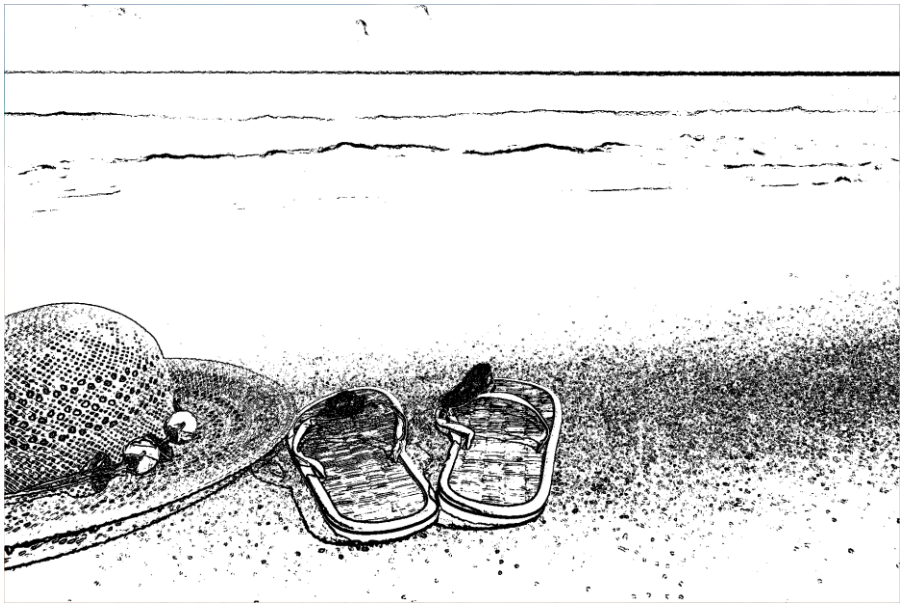
Fonte: Modificado de hat-and-flip-flops-on-the-beach PublicDomainPictures.net (2018)

A escala de contorno quando variada pode trazer resultados bem distintos, como vistos nas Figuras 35 e 36, imagens com muito ou pouco efeito de ruído ocasionados pelo filtro passa-alta podem ser encontradas com a variação da escala de contorno.

Figura 36 Resultado obtido com os parâmetros: ***CgCo:*** (7, -8, 10)

Fonte: Modificado de hat-and-flip-flops-on-the-beach PublicDomainPictures.net (2018)

Figura 37 Resultado obtido com os parâmetros *CgCo:* (1, -10, 0)



Fonte: Modificado de hat-and-flip-flops-on-the-beach PublicDomainPictures.net (2018)

Imagens monocromáticas podem ser obtidas através do *Gamma* máximo como é visto na Figura 37.

4.3.4 Woman

A Figura 38, *Woman*, foi escolhida para demonstrar os resultados com rostos, tendo alta variação de cor, brilho acentuado e alta resolução.

Figura 38 *Woman*

Fonte: *Woman* PublicDomainPictures.net (2018)

Fotografias com muito brilho são mais fáceis de se encontrar as bordas, pois onde existe muito contraste o perfil de *cartoon* pode ser mais acentuado, como é visto na Figura 39.

Figura 39 Resultado obtido com os parâmetros: ***CgCo**:* (1, 0, 0)

Fonte: Modificado de *Woman* PublicDomainPictures.net (2018)

Aumentando a escala de *gamma* para valores acima de 0, a função começa a escurecer a imagem, tonificando as cores de todo espectro luminoso como é visto na Figura 40. Em contrapartida o branco começa a parecer cinza pois a mudança das cores não é situada somente localmente e sim para toda a imagem.

Figura 40 Resultado obtido com os parâmetros: ***CgCo**:* (2, 20, 15)

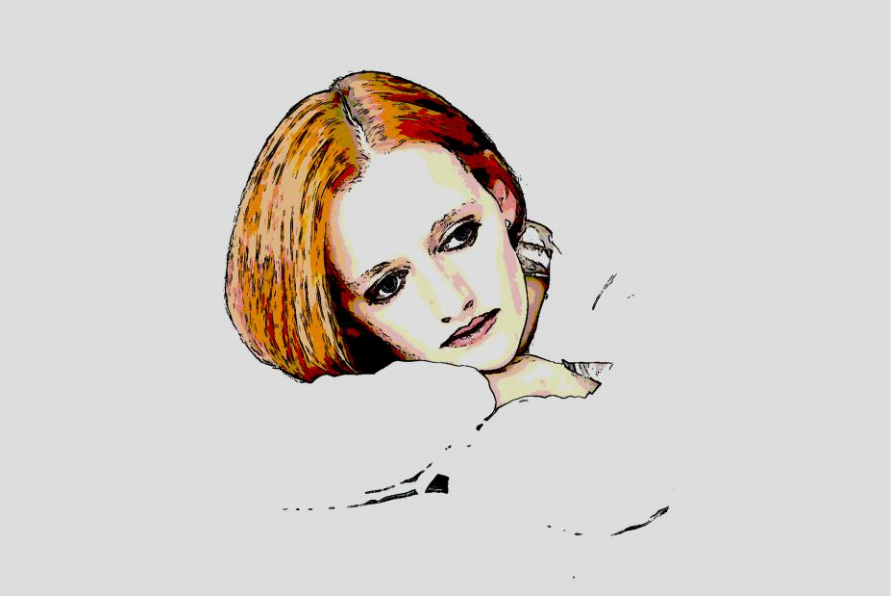
Fonte: Modificado de *Woman* PublicDomainPictures.net (2018)

Figura 41 Resultado obtido com os parâmetros: ***CgCo**:* (6, -9, 7)

Fonte: Modificado de *Woman* PublicDomainPictures.net (2018)

Mais alguns ótimos resultados obtidos pelo algoritmo demonstrados pelas Figuras 41, 42 e 43.

Figura 42 Resultado obtido com os parâmetros: ***CgCo:*** (8, -5, 15)



Fonte: Modificado de *Woman* PublicDomainPictures.net (2018)

Figura 43 Resultado obtido com os parâmetros: ***CgCo:*** (7, 0, 0)

Fonte: Modificado de *Woman* PublicDomainPictures.net (2018)

5. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma introdução a conhecimentos de fundamentos de imagens e vídeos digitais, os filtros de intensidade e de domínio espaciais utilizados, como também uma implementação de um algoritmo para cartonização de imagens e vídeos digitais.

Dependendo dos parâmetros adotados como entrada e a imagem utilizada, por ser um trabalho artístico, o resultado visual poderá agradar um indivíduo, porém não satisfazer a outro. A qualidade dos resultados obtidos pela aplicação também depende que o usuário teste os parâmetros adotados até que encontre um resultado que lhe agrade. No entanto, a tarefa de escolher uma qualidade adequada pode ser subjetiva para algumas imagens e alguns parâmetros extrapolados podem desfigurar a imagem resultante.

O algoritmo de cartonização é bastante eficiente para ser implementado em tecnologias *web* e *mobile*, como exemplo: O *Android* utiliza o *Java* como base. O algoritmo não utiliza variáveis de grande aporte como pontos flutuante de alta precisão, logo a portabilidade dele é alta, sendo fácil a implementação do mesmo em várias linguagens como *Ruby*, *JavaScript*, *TypeScript*, entre outras, a única necessidade para sua portabilidade, é a da implementação ou até mesmo a verificação da existência de uma biblioteca de *software* capaz de realizar rasterização de imagens e renderização de vídeo como ocorre com o *Java* com a existência do *FFMPEG*.

Como trabalhos futuros pretende-se elaborar novos filtros para implementar uma biblioteca de aplicações e comparar com a solução encontrada, buscar a implementação de filtros profissionais como o *bilateral filter*, o *mean shifter filter*, ou até mesmo idealizar novos filtros de filtragem. Além disso, pode-se também elaborar meios de melhorar o desempenho do algoritmo, com a utilização de processamento paralelo. Outro trabalho futuro é a extensão do algoritmo para dispositivos móveis afim de aumentar a acessibilidade de uso.

REFERÊNCIAS

CASTLEMAN K. R. **Digital Image Processing**. New York, Prentice Hall, 1996.

GONZALEZ R. C.; WOODS R. E. **Digital Image Processing**, London, PrenticeHall, 1992.

FILHO M. O.; NETO V. H. **Processamento Digital de Imagens**, Rio de Janeiro Brasport, 1999.

WILHELM B.; MARK J. B. **Digital Image Processing. An algorithmic introduction using Java**. London Springer-Verlag 2007.

CANNY J. **A Computational Approach to Edge Detection**. IEEE Transactions on Pattern Analysis and Machine Intelligence, V. 8, n. 6, 1986.

ARAD N.; GOTSMAN C. **Enhancement by Image-Dependent Warping** IEEE Transactions on Image Processing, V. 8, n. 8, 1999.

TOMASI C.; Manduchi R. **Bialteral Filtering of Gray and Color Image**. IEEE Transactions on Image Processing, 1998.

ELAD M. **On the Origin of the Bilateral Filter and Ways to Improve It**. IEEE Transactions on Image Processing, V. 11, n. 10, 2002.

KANG H.; LEE S.; CHALES K. **Flow-Based Image Abstraction**. IEEE Transactions on Visualization and Computer Graphis, V. 15, n. 1, 2009.

TUAN Q.; LUCAS J. **Separable Bilateral Filtering for Fast Video Processing**. Proceedings of International Conference on Multimedia and Expo (ICME), 2005.

DANNY B.; DORIN C. **A Common Framework for Nonlinear Diffusion, Adaptative Smoothing Bilateral Filtering and Mean Shift**. Elsevier Computer Science, 2004.

GOOCH B.; REINHARD E.; GOOCH A. **A Human Facial Illustratios: Creation and Psychophysical Evaluation** ACM Transactions of Graphis, V. 23, n. 1, pp.27-44 2004.

GEOVANE M.; ALUIR P. **Processo de Detecção de bordas de Canny,** Bol. Ciênc. Geod. sec. Artigos, Curitiba, v. 8, no 2, p.67-78, 2002.

KEVIN D. **Toonify: Cartoon Photo Effect Application**, Department of Electrical Engineering Stanford University. Disponível em:
https://stacks.stanford.edu/file/druid:yt916dh6570/Dade_Toonify.pdf
Último acesso em 30 março 2018.

SVEN H. W.; C. OLSEN; B. GOOCH. **Real time video abstraction**, Northwestern University. Disponível em: http://kucg.korea.ac.kr/seminar/2008/src/PA-07-12.pdf
Último acesso em 30 março 2018.

NARENDRAN A. **Cartoonifying an Image**, Colorado School of Mines. 2013.
Disponível em:
http://inside.mines.edu/~whoff/courses/EENG510/projects/2013/Anil_slides.pdf
Último acesso em 30 março 2018.

RUSSO F. **As 20 animações de maior bilheteria na história**. Publicação em Revista Online AdoroCinema.com. pp. 01. Disponível em:
http://www.adorocinema.com/noticias/filmes/noticia-116062/
Último acesso em 28 janeiro 2018.

BORGES L. T. **20 maiores animações da história do cinema**. Publicação em Revista Online Forbes.uol.com.br pp. 01. Disponível em:
http://forbes.uol.com.br/listas/2015/10/20-maiores-animacoes-da-historia-cinema/
Último acesso em 29 março 2018.

MediaCollege.com frame-rate
Disponível em: https://www.mediacollege.com/video/frame-rate/
Último acesso em 06 abril 2018.

Oracle.com **API Swing especificação da Oracle**
Disponível em: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
Último acesso em 29 março 2018.

Oracle.com **Classe *BufferedImage* especificação da Oracle**
Disponível em: https://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html
Último acesso em 12 fevereiro 2018.

Oracle.com **Classe Color especificação da Oracle**
Disponível em: https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html
Último acesso em 12 fevereiro 2018.

publicdomainpictures.net blue-morpho-butterfly Disponível em:
http://www.publicdomainpictures.net/view-image.php?image=7275&picture=blue-morpho-butterfly
Último acesso em 19 fevereiro 2018.

publicdomainpictures.net burning-sky Disponível em:
http://www.publicdomainpictures.net/view-image.php?image=36842&picture=burning-sky
Último acesso em 19 fevereiro 2018.

publicDomainPictures.net hat-and-flip-flops-on-the-beach Disponível em: http://www.publicdomainpictures.net/view-image.php?image=21489&picture=hat-and-flip-flops-on-the-beach
Último acesso em 19 fevereiro 2018.

publicdomainpictures.net woman Disponível em: http://www.publicdomainpictures.net/view-image.php?image=47527&picture=woman
Último acesso em 19 fevereiro 2018.

publicdomainpictures.net village Disponível em: https://www.publicdomainpictures.net/en/view-image.php?image=15785&picture=village
Último acesso em 19 fevereiro 2018.