MÁSCARA ALEATÓRIA PARA REPRESENTAÇÃO MONOCROMÁTICA EM ALTA QUALIDADE DE IMAGENS EM TONS DE CINZA

Sarina Lustosa da Costa Santos

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Ricardo Cordeiro de Farias

Rio de Janeiro
Outubro de 2013

MÁSCARA ALEATÓRIA PARA REPRESENTAÇÃO MONOCROMÁTICA EM ALTA QUALIDADE DE IMAGENS EM TONS DE CINZA

Sarina Lustosa da Costa Santos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof. Ricardo Cordeiro de Farias, Ph.D.

_____
Prof. Alexandre de Assis Bento Lima, D.Sc.

_____
Prof. Esteban Walter Gonzalez Clua, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 2013

Santos, Sarina Lustosa da Costa

Máscara aleatória para representação monocromática em alta qualidade de imagens em tons de cinza/ Sarina Lustosa da Costa Santos. – Rio de Janeiro: UFRJ/COPPE, 2013.

XI, 63 p.: il.; 29,7 cm.

Orientador: Ricardo Cordeiro de Farias

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2013.

Referências Bibliográficas: p. 42.

1. *Halftoning*. 2. Máscara Aleatória. 3. Representação Monocromática em Alta Qualidade. I. Farias, Ricardo Cordeiro de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A minha família e amigos, em especial,
aos meus pais que sempre me apoiaram
durante essa jornada.

Agradecimentos

Gostaria de agradecer aos meus pais Mauricio Rocha da Costa Santos e Maria Eline Marques Lustosa da Costa Santos, por todo seu apoio e incentivo antes e durante esta pesquisa. Ao meu orientador, Ricardo Farias (*Phd*) por seus conselhos, explicações, dicas e ajuda. Ao amigo e colega da linha de Computação Gráfica Luciano de Paula (*M.Sc.*) por suas valiosas contribuições durante todo o mestrado. Aos amigos e professores do LCG. À minha irmã Samila Lustosa por sempre aceitar adiar nossas viagens e passeios de acordo com o cronograma do mestrado. À minha professora de graduação Cristiana Bentes (*D.Sc.*) por me incentivar e me ajudar a entrar no mestrado. E por fim aos meus orientadores na graduação, Guilherme Abelha Mota (*D.Sc.*) e João Araujo (*D.Sc.*), por me introduzirem na linha de pesquisa de Computação Gráfica e sem os quais eu não teria voltado minha atenção à pesquisa científica.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MÁSCARA ALEATÓRIA PARA REPRESENTAÇÃO MONOCROMÁTICA EM ALTA QUALIDADE DE IMAGENS EM TONS DE CINZA

Sarina Lustosa da Costa Santos

Outubro/2013

Orientador: Ricardo Cordeiro de Farias

Programa: Engenharia de Sistemas e Computação

Apresentamos uma proposta de uma técnica para a representação monocromática em alta qualidade de imagens em tons de cinza usando a técnica de *halftoning* com máscaras aleatórias. Nosso algoritmo calcula aleatoriamente máscaras, com níveis 4x4 e 6x6, a serem utilizadas pelo algoritmo de *halftoning* para gerar uma impressão em alta qualidade de imagens em tons de cinza em impressoras a laser monocromáticas, visto que este tipo de impressora apresenta resultados de melhor qualidade gráfica quando comparada às impressoras a jato de tinta. O algoritmo de *halftoning* foi desenvolvido utilizando a correção linear e a correção *gamma*. Além disso, o *halftoning* foi implementado em duas resoluções diferentes: 300x300 *pixels* e 600x600 *pixels*. É feita uma comparação entre os resultados obtidos utilizando o algoritmo de *halftoning* com máscara estática ou máscara aleatória e também, uma comparação entre os resultados obtidos utilizando correção linear ou correção *gamma*.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

RANDOM MASK FOR MONOCHROMATIC REPRESENTATION OF GRAYSCALE IMAGES IN HIGH QUALITY

Sarina Lustosa da Costa Santos

October/2013

Advisor: Ricardo Cordeiro de Farias

Department: Systems Engineering and Computer Science

We present a new approach for printing gray scale images in high quality using the *halftoning* technique with random masks. Our algorithm randomly calculate masks, with 4x4 and 6x6 levels, that will be used by the *halftoning* algorithm to reproduce a high quality print of gray scale images using laser printers, since this kind of printer presents better graphics results when compared to ink jet printers. The *halftoning* algorithm was developed using linear and *gamma* correction. Besides that, the *halftoning* was implemented in two different resolutions: 300x300 pixels and 600x600 pixels. A comparison is made between the results obtained using the *halftoning* algorithm with static mask or random mask and also, a comparison between the results obtained using linear or *gamma* correction.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

1. Introdução

1.1. Introdução

Para se obter uma impressão em alta qualidade de imagens utilizam-se equipamentos de impressão como a impressora a laser [12]. Este tipo de impressora foi inventado no ano de 1969 pelo pesquisador Gary Starkweather, na empresa Xerox [11], modificando uma máquina copiadora. A impressora a laser utiliza como o próprio nome já diz, a tecnologia do raio laser modulado.

O funcionamento de uma impressora a laser se dá da seguinte forma: a impressora carrega a imagem a ser impressa em sua memória e a processa de forma que ela saiba quais partes serão pintadas e quais serão deixadas em branco. Em seguida, um feixe de laser projeta a imagem em um tambor ou cilindro eletricamente revestido com Selênio ou fotocondutores orgânicos, resultando assim em uma imagem eletrostática. O *toner*, então, joga uma película de pó (partículas de tinta) sobre o cilindro e através do calor e do contato direto esse pó é impregnado ao papel, resultando na imagem impressa.

Para este trabalho, o mais importante em relação ao funcionamento das impressoras a laser é a parte onde a impressora processa a imagem a ser impressa em sua memória, isto é, como a impressora identifica quais pontos devem ser pintados e quais devem ser deixados em branco de forma a ter o mínimo de perda de qualidade possível.

A ideia de que a justaposição de vários pequenos pontos de algumas cores possa ser visualmente notada como uma imagem colorida de tom contínuo foi explorada inicialmente por artistas, como Georges Seurat, para criar o estilo conhecido como "pontilhismo". O mesmo principio é utilizado hoje em dia em monitores de tubo de raios catódicos (*CRT*) e cristal líquido (*LCD*), que usam vermelho, verde e azul para produzir um espectro de cores. São também utilizados para impressão de livros e na maioria das impressoras [1].

A resolução de pontos por polegada (*DPI - Dots Per Inch*) em uma impressora a laser é o número de pequenos pontos pretos que a impressora pode depositar no papel por polegada. Em geral, essa resolução é a mesma nas duas direções do papel, tanto na linha (horizontal) quanto na altura (vertical). Entretanto, os pontos colocados no papel por estas impressoras são pretos e tais impressoras não possuem um ajuste de tom de cinza para imprimir as várias tonalidades que uma imagem pode ter. Para criar uma escala de tons de cinza, é necessário usar uma amostragem desses pontos, sendo uma técnica geralmente conhecida como *halftoning*. Ela é comumente usada em jornais, revistas, livros e pode ser utilizada para imagens coloridas e também imagens monocromáticas. A diferença na qualidade do *halftoning* está no número de *DPI* que pode ser colocado no papel e a forma como estes pontos foram organizados para produzir o resultado de tom de cinza [1].

Normalmente, uma imagem digital colorida é armazenada em computadores utilizando o modelo de cores *RGB* (*Red Green Blue*) [2], onde cada *pixel* armazena três valores de intensidade, um para o tom vermelho, outro para o verde e por último para o azul. Já uma imagem digital em tons de cinza é uma imagem onde o valor de cada *pixel* armazena apenas a informação da intensidade de preto, resultando em uma imagem comumente chamada de imagem preta e branca, onde os *pixels* variam de tons totalmente brancos até totalmente pretos. Entretanto, não se deve confundir uma imagem em tons de cinza com uma imagem binária, que consiste em uma imagem que possui apenas *pixels* nas cores branca e preta, ou seja, ela não possui *pixels* em tons de cinza que variam entre o branco e o preto.

Há várias formas para se converter uma imagem colorida em uma imagem em tons de cinza. Uma estratégia comumente utilizada é combinar a luminância da imagem em tons de cinza com a luminância da imagem colorida. A luminância de uma imagem pode ser considerada como o indicador de brilho de uma imagem. Para isso, é necessário obter os valores de vermelho, verde e azul de cada *pixel* da imagem em intensidade linear através da expansão *gamma* a seguir [2]:

$$C_{linear} = \begin{cases} \left(\dfrac{C_{rgb} + 0.055}{1.055} \right)^{2.4}, C_{rgb} > 0.04045 \\ \dfrac{C_{rgb}}{12.92}, C_{rgb} \leq 0.04045 \end{cases}$$

Onde:

C_{linear} = intensidade linear variando de 0 a 1

C_{rgb} = valores de cada cor RGB variando de 0 a 1

A luminância é calculada a partir da soma ponderada dos três valores de intensidade linear, conforme a fórmula a seguir:

$$Y = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B$$

Onde:

Y = luminância linear

R = intensidade linear de vermelho

G = intensidade linear de verde

B = intensidade linear de azul

É necessário ainda comprimir a luminância linear através da compressão *gamma* para obter o tom de cinza na escala convencional, como a escala *RGB*. Para isso, deve-se calcular a intensidade do tom de cinza da seguinte forma:

$$C_{rgb} = \begin{cases} 12.92 \times C_{linear}, C_{linear} \leq 0.0031308 \\ 1.055 \times C_{linear}^{1/2.4} - 0.055, C_{linear} > 0.0031308 \end{cases}$$

Uma boa aproximação dessa fórmula de conversão de imagem colorida em tons de cinza seria:

$$Cinza = \left(0.2126 \times R^{2.2} + 0.7152 \times G^{2.2} + 0.0722 \times B^{2.2}\right)^{1/2.2}$$

Um exemplo de conversão de uma imagem colorida para tons de cinza através do *software* de edição de imagens *GIMP* pode ser visualizado na Figura 1.1.

Figura 1.1: Conversão de uma imagem colorida para tons de cinza

1.2. A Proposta do Trabalho

Este trabalho tem como objetivo estudar e propor uma forma de representar monocromaticamente em alta qualidade imagens em tons de cinza em impressoras a laser utilizando a técnica de *halftoning* com o uso de máscaras aleatórias e também o uso da correção *gamma* para melhorar o resultado obtido.

Esta pesquisa foi desenvolvida em ambiente *Linux* [10], utilizando a linguagem de programação C++ [3] [9] e as bibliotecas *OpenCV* [5], *OpenGL* [4] [8] e *GLUT* [6]. A primeira biblioteca, *OpenCV*, foi utilizada na parte de leitura e escrita de imagens e conversão de imagens. Já a biblioteca *OpenGL* foi utilizada para operações no âmbito da computação gráfica e por fim, a biblioteca *GLUT* foi utilizada para gerenciar o sistema de janelas e interface com o usuário, como *mouse* e teclado.

1.3. Organização da Dissertação

Esta dissertação está organizada em cinco capítulos, onde o primeiro introduz o tema desta pesquisa, representação monocromática em alta qualidade de imagens em tons de cinza utilizando a técnica de *halftoning* com máscaras aleatórias, e explica os objetivos do trabalho realizado.

O segundo capítulo aborda os trabalhos relacionados ao tema da pesquisa e também faz uma revisão bibliográfica acerca do assunto, explicando a técnica de *halftoning,* o uso da correção *gamma* nessa técnica, as máscaras de *halftoning* e a linguagem *Postscript*.

No terceiro capítulo é descrito como foi desenvolvido e implementado o algoritmo de *halftoning* e também as máscaras estáticas e aleatórias utilizadas no algoritmo.

O quarto capítulo mostra os resultados obtidos a partir dessa pesquisa e faz também uma comparação entre os resultados obtidos utilizando a técnica de *halftoning* com máscaras estáticas e com máscaras aleatórias. Mostra ainda as resultados da aplicação da correção *gamma* no algoritmo de *halftoning*.

Por fim, no quinto capítulo é feita uma análise e conclusão do que foi obtido neste trabalho.

Capítulo 2

2. Trabalhos Relacionados

Neste capítulo serão abordados os trabalhos relacionados ao tema de impressão em alta qualidade de imagens em tons de cinza em impressoras a laser utilizando a técnica de *halftoning*. Além de fazer uma revisão bibliográfica acerca dos assuntos relacionados ao tema, como uso de correção *gamma*, máscaras de *halftone* e também a linguagem de programação para descrição de páginas para impressoras, conhecida como *Postscript*.

2.1. *Halftoning*

Entende-se por *halftoning* a técnica de agrupar individualmente os pontos pretos produzidos pela impressora para representar uma imagem. Normalmente uma imagem contém uma variação contínua de tons de cinza ou cores e a técnica de *halftoning* converte essa imagem e a imprime utilizando apenas uma cor de tinta, no caso a cor preta, e utiliza esse grupo de pontos pretos para gerar uma ilusão de óptica para o observador de forma que ele não veja esse grupo de pontos pretos e sim uma mistura deles, que para o olho humano parecerá uma suave transição entre os tons de cinza. Esta forma de impressão foi idealizada por *William Fox Talbot* por volta do ano de 1850, sendo umas das primeiras fotografias impressas utilizando a técnica de *halftoning*. A imagem de *Steinway Hall* em *Manhattan* publicada em dezembro de 1873 (Figura 2.1) [14].

Figura 2.1: Primeira fotografia impressa utilizando a técnica de *halftoning* em dezembro de 1873

Em processamento de imagens, máscaras são matrizes que são aplicadas sobre uma imagem para se filtrar ou realizar transformações *pixel* a *pixel*, onde essas transformações não dependem apenas do nível de cinza de um *pixel*, mas também do nível de cinza dos *pixels* vizinhos. Com isso, cada grupo de pontos pretos utilizado na técnica de *halftoning* é chamado de célula ou máscara de *halftone* e pode ser considerada como sendo um conjunto de 16 pontos em uma matriz 4x4. Nesta célula, todos ou apenas alguns desses pontos podem ser impressos. Assim sendo, caso esta célula seja razoavelmente pequena, o olho humano não irá perceber os pontos individualmente, e sim uma mistura dessas células que gerarão o efeito de um tom de cinza, ou seja, uma espécie de ilusão de óptica [1]. Exemplos de escalas de cinza geradas a partir de uma célula 4x4 podem ser visualizados na Figura 2.1.

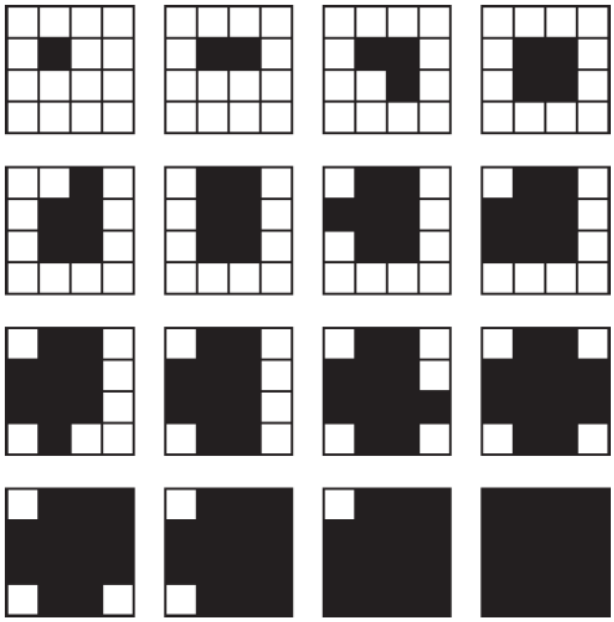


Figura 2.2: Escala de cinza gerada usando uma célula de *halftone* 4x4 [1]

Monitores de vídeo normalmente possuem uma resolução que varia de 72 a 90 *pixels* por polegada, sendo assim se cada *pixel* corresponder a uma célula de *halftone*, a imagem pode ser impressa com a mesma dimensão que ela aparece na tela. No caso de uma célula de *halftone* 4x4 e uma impressora de 300 *dpi*, isso gerará uma resolução de 75 células por polegada e cada célula usará um dos 17 possíveis níveis de cinza para representar o nível de cinza de cada *pixel* [1].

Geralmente, o número máximo de tons de cinza disponível utilizando a técnica de *halftoning* segue a regra a seguir:

$$TonsdeCinza = 1 + \left(\frac{dpi}{lpi} \right)^2$$

Onde:

dpi = resolução da impressora em pontos por polegada

lpi = resolução do *halftone* em linhas por polegada

Normalmente as impressoras a laser possuem uma resolução de 300 ou 600 *dpi* e como no exemplo citado anteriormente, para um *halftone* com célula 4x4, tem-se:

$$lpi = \frac{300}{4} = 75$$

Logo, o número máximo de tons de cinza que poderão ser utilizados para este *halftone* será:

$$1+\left(\frac{300}{75}\right)^2 = 17$$

Uma desvantagem da técnica de *halftoning* é que normalmente uma imagem possui 256 tons de cinza e utilizar apenas 17 tons parece um tanto fraco, entretanto existem outras desvantagens e limitações, como o tamanho dos pontos e das células, que pode ser visualmente percebido pelo observador caso não seja pequeno o suficiente. Além disso, cada método de impressão produz pontos de uma forma diferente, isto é, impressoras matriciais usam pequenos pinos para pressionar uma fita com tinta sobre o papel, impressoras a jato de tinta produzem pequenas gotículas de tinta, impressoras termais, usadas em máquinas de fax, usam um pino para passar uma corrente elétrica através de uma camada no papel. Impressoras a laser trabalham como uma máquina de copiar, conforme explicado anteriormente [1]. Então nem sempre é possível obter uma impressão tão boa quanto a original.

2.1.1.Correção *Gamma*

O olho humano não responde linearmente a escala de cinza, e sim logaritmicamente. Desta forma é necessário aplicar uma correção *gamma* para se obter uma imagem na qual a impressão visual de brilho varia linearmente com o valor do *pixel*, ou seja, um ajuste através de uma curva *gamma* para comprimir ainda mais os valores de tons de cinza mais escuros e expandir os valores mais claros [1]. Sendo assim, uma imagem sem a correção *gamma* poderia aparecer esbranquiçada ou muito escura.

A curva *gamma* que irá compensar a resposta não linear do olho humano pode ser definida como uma relação proporcional entre o valor de tom de cinza de entrada elevado a potencia do valor de *gamma*, ou seja, para valores de tom de cinza representados no modelo *RGB* de cores variando no intervalo de 0 a 255, tem-se:

$$I_{out} = \left(\frac{I_{in}}{255}\right)^{\gamma}$$

Onde:

I_{out} = valor do tom de cinza de saída

I_{in} = valor do tom de cinza de entrada

γ = valor do *gamma*

Entretanto, para se calcular a correção *gamma*, é necessário elevar o valor de entrada ao inverso do valor de *gamma*, então:

$$I_{out} = \left(\frac{I_{in}}{255} \right)^{1/\gamma}$$

Na Figura 2.3 tem-se o gráfico da correção *gamma* calculada para valores de tons de cinza, no modelo *RGB* de cores, variando de 0 a 255, ou seja, 256 tons de cinza e o *gamma* variando em três valores: 0.5, 1.0 e 1.5.

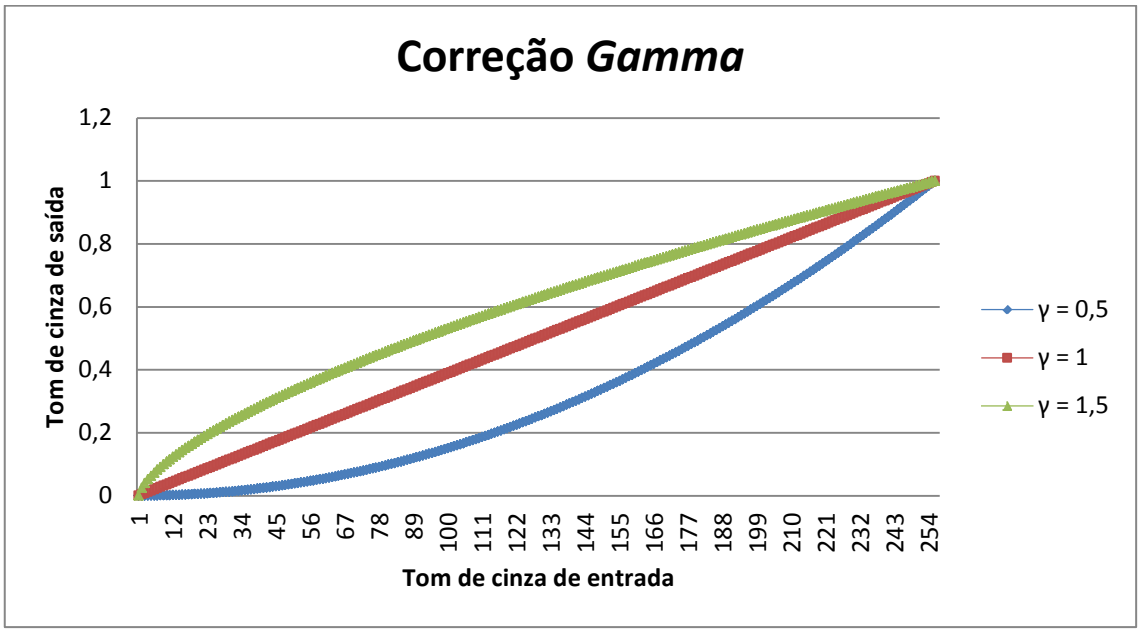


Figura 2.3: Gráfico da correção *gamma* para três valores distintos de *gamma*

Pode-se observar que para γ = 1, tem-se uma curva linear e para valores onde γ < 1, a correção *gamma* possui um arco com curvatura para baixo e para valores onde γ > 1, a curvatura do arco fica voltada para cima.

Essa variação no valor do *gamma* na correção afeta, como dito anteriormente, no nível de brilho de uma imagem. Esse efeito pode ser visualizado na Tabela 2.1, onde a correção foi aplicada em uma imagem variando o valor de *gamma* para γ = 0.5. γ = 1.0 e γ = 1.5 respectivamente.

Tabela 2.1: Aplicação da correção *gamma* em uma imagem variando os valores de *gamma* em γ = 0.5, γ = 1.0 e γ = 1.5

		
γ = 0.5	γ = 1.0	γ = 1.5

2.1.2. Máscaras de *Halftone*

Como dito no início deste capítulo, a técnica de *halftoning* usa células ou máscaras para agrupar individualmente os pontos pretos para imprimir uma imagem em tons de cinza. A forma como essas máscaras são construídas afeta o resultado do *halftoning*, e por isso, é importante a escolha da melhor máscara possível para se obter os melhores resultados.

Um fator importante na escolha de qual máscara utilizar é a resolução em linhas por polegada (*lpi*) que ela permitirá que o *halftoning* seja feito. Geralmente, jornais são impressos com uma resolução que varia de 50 a 85 *lpi*, revistas de 100 a 120 *lpi* e uma reprodução em alta qualidade exige uma resolução de 120 a 150 *lpi*.

Outro fator importante é a forma como os pontos pretos são distribuídos na máscara, pois dependendo dessa forma de distribuição dos pontos, podem ocorrer padrões na imagem, como por exemplo, regiões arredondadas de preto dentro de uma moldura branca, padrões circulares, padrões em linhas horizontais ou diagonais, padrões em cruz, entre outros. A maioria das impressoras *Postscript* usa o padrão diamante, pois ele permite uma aproximação no arredondamento dos pontos fazendo com que os pontos com uma densidade de preto menor do que 50% apareçam diferentes dos pontos com densidade de preto maior do que 50% [1]. Um exemplo de uma sequência de dez máscaras de *halftone* com dimensão 6x6 pode ser visualizado na Figura 2.4 a seguir.

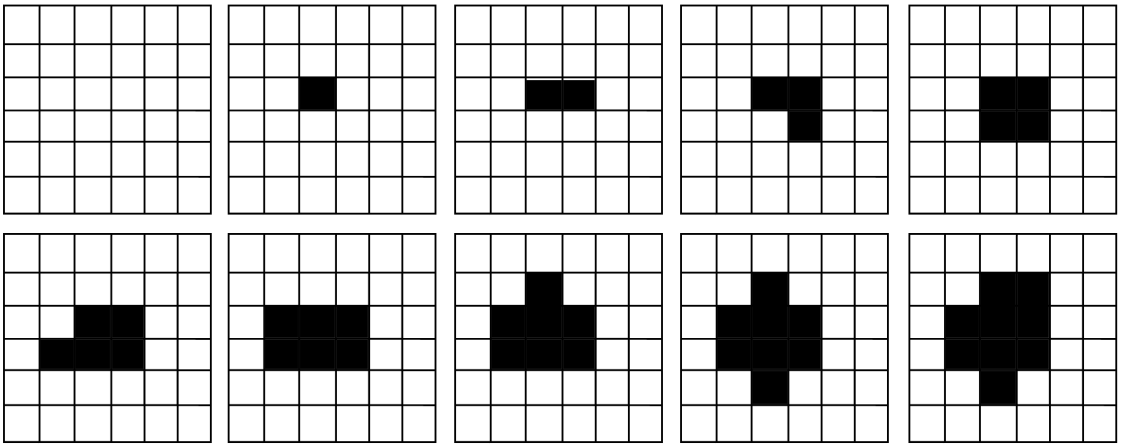


Figura 2.4: Exemplo de uma sequência de dez máscaras de *halftone* com dimensão 6x6

Uma forma de diminuir esses padrões visuais é distribuir os pontos aleatoriamente na máscara. Entretanto isso exige um processamento computacional adicional com a impressora. Além disso, impressoras com baixa resolução *dpi* não podem produzir máscaras de *halftone* que sejam pequenas e ainda assim representar vários tons de cinza sem gerar padrões que distraiam visualmente a imagem. Uma forma de melhorar essa situação é usar o método de *dithering* para representar a imagem. Normalmente imagens resultantes de um processo de *dithering* possuem uma boa visualização na transição gradual de variação do gradiente de brilho, já que o olho humano responde a uma média da densidade e espaçamento dos pontos [1]. Na Figura 2.5 têm-se dois exemplos de um dos vários métodos de *dithering* aplicado em duas imagens em tons de cinza. Neste caso foi aplicado o método *Floyd-Steinberg* nas duas imagens.

Essa busca pela diminuição dos padrões visuais ou padrões de impressão gerados pelas máscaras de *halftone* é um tema que pode ser encontrado em alguns trabalhos, como o artigo publicado por *Kang* [18], onde foi feita uma pesquisa utilizando *halftoning* com microcluster, que consiste em uma abordagem híbrida entre *ditherings* ordenados com cluster de pontos e pontos dispersos. Tendo como objetivo superar a troca entre frequência de monitores e o tom de cinza dos pontos clusterizados, para minimizar estruturas distintas de pontos dispersos e também reduzir a granularidade. Assim sendo, uma célula de *halftone* é dividida em pequenas subcélulas, onde a modulação de frequência é utilizada para dispersar o núcleo nessas subcélulas. Além disso, uma randomização estocástica é feita no núcleo da subcélula para prover uma dispersão mais homogênea. Quando os núcleos estão estabelecidos um *dither* ordenado por pontos clusterizados é aplicado para o crescimento dos pontos.

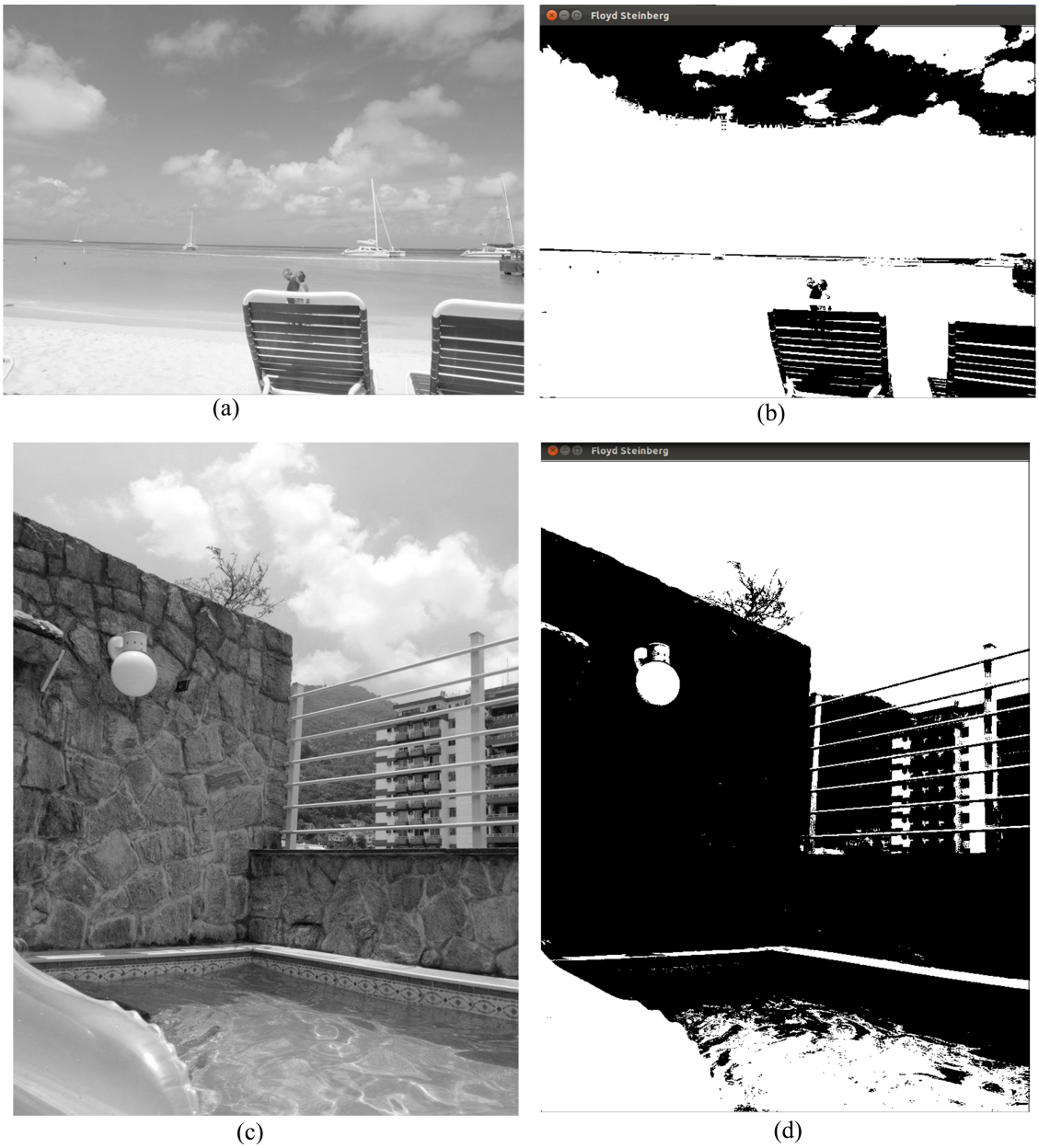


Figura 2.5: Algoritmo de *Floyd-Steinberg* aplicado em duas imagens: (a) primeira imagem em tons de cinza, (b) primeira imagem com o algoritmo de *Floyd-Steinberg* aplicado, (c) segunda imagem em tons de cinza e (d) segunda imagem com o algoritmo de *Floyd-Steinberg* aplicado

Outro trabalho relacionado à esse tema, é o artigo publicado por *Chih-Ching Lai* e *Din-Chang Tseng* [17]. Nele foi feita uma pesquisa utilizando uma técnica de *halftoning* baseada no modelo do método dos mínimos quadrados e usando um algoritmo genético para produzir uma imagem gerada por *halftone*, visando a minimização da diferença na refletância percebida entre a imagem original e a imagem gerada por *halftone*. O método dos mínimos quadrados foi incorporado às propriedades da visão humana para medir a diferença entre as duas imagens. E o algoritmo genético foi utilizado para investigar os problemas comumente encontrados nesse tema. Nesse estudo, foi utilizado um modelo modificado de impressora de sobreposição de ponto para compensar a

distorção do nível de cinza. O modelo de impressora foi combinado com um algoritmo de medição para estimar o raio do ponto impresso e fazer com que o *halftoning* proposto se adapte para a grande variedade de impressoras e papéis que existem hoje em dia.

Em dois artigos publicados por *Anderson* [15][16] é proposto um algoritmo genético que gera automaticamente máscaras de *halftone* otimizadas para o uso em sistemas específicos de impressão. Essa pesquisa foi baseada no modelo detalhado do processo de impressão e no sistema de visão humana. Os experimentos mostraram que os algoritmos genéticos são eficazes em achar máscaras de *halftone* melhores. Além disso, foi utilizado um método de medição que pode determinar a qualidade de uma máscara de *halftone* para um tipo específico de impressora. Outra melhora significativa apresentada neste trabalho são dois métodos de redução do espaço de busca para um pequeno subconjunto de máscaras de *halftone* que aumentam significativamente o desempenho do algoritmo genético.

2.2. *PostScript*

PostScript (*PS*) [7] é uma linguagem de programação interpretativa de propósito geral que foi criada por *John Warnock* e *Charles Geschke* em 1985, pela empresa *Adobe* [13], para descrição de páginas e impressão de documentos. Esta linguagem permite criar vetores gráficos, ou seja, ela possui comandos gráficos específicos para desenhar letras, formas gráficas, figuras, tipos de traçados e formas de representação de imagens. Atualmente a linguagem está na versão três.

Um programa escrito nesta linguagem pode comunicar a descrição de um documento para um sistema de impressão ou até controlar a aparência do texto e dos gráficos em um monitor. A descrição de uma página *PostScript* pode ser renderizada em uma impressora, monitor ou qualquer outro tipo de saída que possua o interpretador de *PostScript*. O interpretador converte a descrição da página, que está descrita em uma linguagem de alto nível como a *PostScript*, para o formato baixo nível do equipamento de saída, em linguagem de máquina [7].

A principal característica da linguagem *PostScript* é a sua capacidade de lidar com a maioria dos formatos de rasterização de equipamentos de saída. Esses formatos englobam tecnologias como filmadoras digitais, monitores e impressoras a laser, matriciais e jato de tinta.

Um elemento gráfico, como uma linha ou um círculo, é renderizado em um equipamento de saída através de um processo chamado *scan conversion*. Dada uma descrição matemática do elemento gráfico, este processo determina quais *pixels* desenhar e quais valores esses *pixels* receberão para atingir a melhor impressão possível no limite da resolução do dispositivo de saída. Quando a interpretação da descrição da página está completa, os valores dos *pixels* na memória representam a aparência da página. Neste momento, um processo de rasterização de saída já é capaz de fazer esta representação estar visível em uma página impressa ou em um monitor [7].

Para se renderizar elementos em escala de cinza em equipamentos que os *pixels* só podem ser branco ou preto é necessária a técnica de *halftoning*, explicada anteriormente. Isto permite que equipamentos de saída monocromáticos possam reproduzir vários tons de cinza aproximando o resultado impresso o mais próximo possível da imagem original.

Teoricamente, um programa poderia descrever uma página como um vetor de *pixels*, entretanto, isto seria insatisfatório e volumoso, pois o tamanho deste vetor dependeria dos requisitos dos equipamentos envolvidos, como por exemplo, o tamanho da memória. Visto que este vetor de *pixels* poderia extrapolar a capacidade de armazenamento da maioria dos computadores pessoais, principalmente no caso da impressão de imagens em alta resolução, que exigiria um longo vetor de *pixels* para armazenar este tipo de imagem. Uma linguagem de descrição de página, como a *PostScript*, permite produzir arquivos compactos que facilitam no armazenamento e transmissão dos mesmos, tornando o processo de impressão independente de requisitos particulares dos aparelhos de saída [7].

Um modelo de imagem em alto nível permite uma aplicação descrever a aparência de páginas contendo textos, formas gráficas e imagens em termos de elementos gráficos abstratos. Este modelo pode ser utilizado para produzir saídas em alta qualidade em diferentes impressoras e monitores. Este modelo de imagem é expresso através de linguagens de descrição de página, como *PostScript* [7].

Sendo assim, um programa que produz uma saída que será impressa, gera a mesma em basicamente dois estágios: primeiramente, a aplicação gera a descrição da página na linguagem *PostScript* ou similar, e por fim, o programa que controla o aparelho de saída interpreta a descrição e a renderiza neste aparelho. Estes dois estágios podem ser executados em lugares e momentos distintos, ou seja, a linguagem de descrição de

página serve como um padrão de intercâmbio para a transmissão e armazenamento dos documentos para impressão [7].

A linguagem *PostScript* é uma linguagem de formato dinâmico, ou seja, permite uma flexibilidade maior, pois o conjunto de operadores pode ser extensível e o significado exato do operador pode ser desconhecido até o momento em que ele é encontrado. Além disso, o formato dinâmico possui elementos como procedimentos, variáveis e estruturas de controle. Outro formato existente para linguagens de descrição de página é o formato estático, que possui um conjunto fixo de operações e uma sintaxe fixa para especificar operações e argumentos [7].

O interpretador de *PostScript* executa o programa e produz a saída para a impressora ou monitor. Existem três formas que o interpretador e a aplicação podem interagir: modo convencional, onde uma descrição de página é enviada para o interpretador, que produz a saída; modo integrado, onde a aplicação interage com o interpretador controlando o monitor ou o sistema de janelas, e assim, de acordo com as ações do usuário, a aplicação envia comandos para o interpretador ou lê informações contidas nele; e por fim, o modo interativo, onde o programador envia comandos para o interpretador, que serão executados imediatamente [7].

Capítulo 3

3. Máscara Aleatória

Neste capítulo será descrito como o algoritmo de *halftoning* utilizando máscaras aleatórias foi implementado para se obter uma impressão em alta qualidade de imagens em tons de cinza em impressoras a laser. Como dito anteriormente, este algoritmo foi desenvolvido na linguagem de programação C++ e utilizando as bibliotecas gráficas *OpenCV*, *OpenGL* e *Glut*.

3.1. O Algoritmo

O algoritmo de *halftoning* implementado neste trabalho foi desenvolvido de forma que ele é aplicado apenas em uma região em volta do ponteiro do mouse sobre a imagem e não em toda a imagem. Foi adotada uma região na forma quadrada onde o tamanho dessa região varia de acordo com o número de *pixels* que o *halftone* utiliza em linhas por polegada (*LPI*). Sendo assim, calcula-se o número de *pixels* do *halftone* dividindo a resolução do *halftone* em pontos por polegada (*DPI*) pela dimensão da máscara de *halftone* a ser utilizada:

$$htNumPxls = \frac{ht\,\mathrm{Re}\,s}{htLevel}$$

Onde:

htNumPxls = número de *pixels* do *halftone* em *LPI*

htRes = resolução do *halftone* em *DPI*

htLevel = dimensão da máscara de *halftone*

Foram adotadas resoluções de *halftone* nos valores de 300 *dpi* e 600 *dpi*. Para as máscaras foram implementadas células de *halftone* com dimensões de 4x4 e 6x6.

Em seguida, calcula-se a região da imagem onde será aplicado o *halftone*. Para isso, adota-se que o ponteiro do mouse é o centro desta região e o lado do quadrado mede o valor do número de *pixels* do *halftone*. Então, calcula-se o canto inferior esquerdo dessa

região através da posição do ponteiro do mouse e subtraindo dela metade do número de *pixels* do *halftone*:

$$region = mousePos - \frac{htNumPxls}{2}$$

Onde:

region = região da imagem onde o *halftone* será aplicado

mousePos = posição do ponteiro do *mouse* sobre a imagem

htNumPxls = número de *pixels* do *halftone* em *LPI*

Na Figura 3.1 pode-se visualizar essa região da imagem em que será aplicada a técnica de *halftoning*, esta região está sinalizada pelo quadrado vermelho em volta do ponteiro do *mouse*.

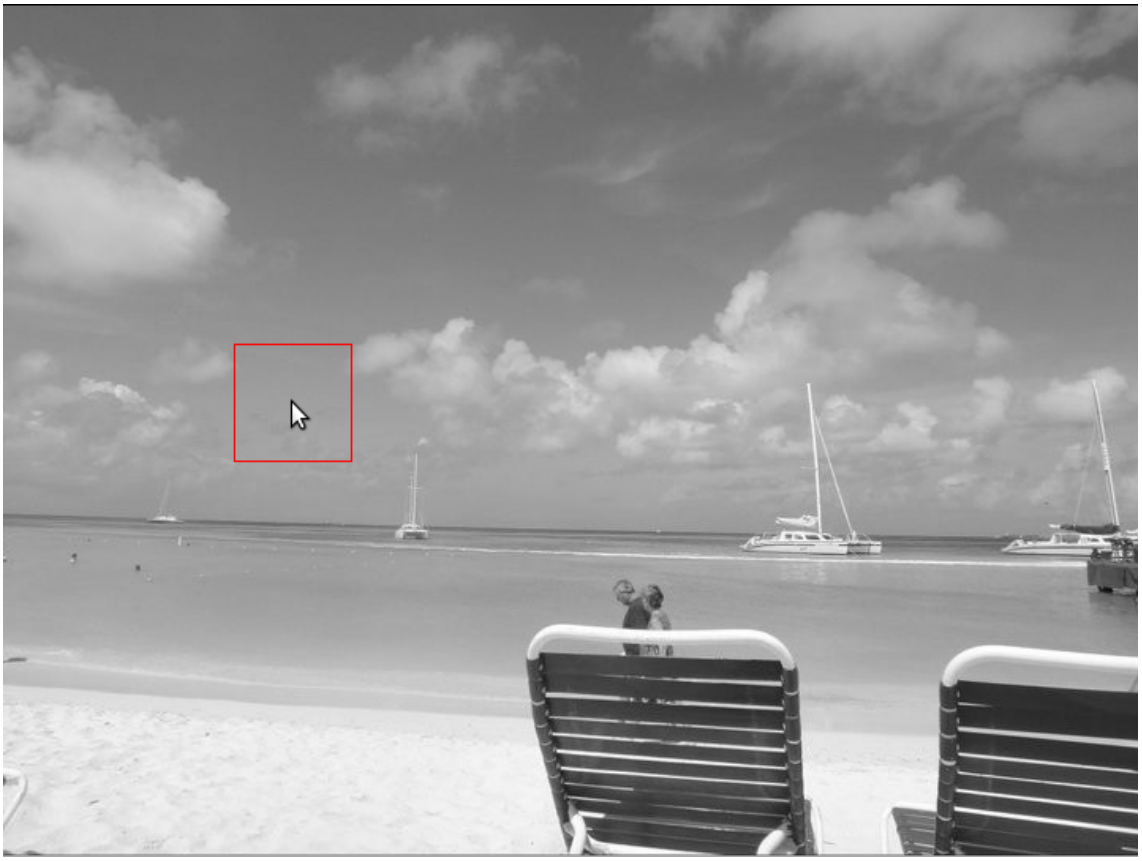


Figura 3.1: Visualização da região da imagem em que será aplicada a técnica de *halftoning* através do quadrado vermelho em volta do ponteiro do *mouse*.

Feito isso, percorre-se a imagem dentro dessa região calculada e para cada *pixel*, calcula-se o índice do *halftone*, podendo ser calculado usando a correção linear (Figura 3.2) ou a correção *gamma* (Figura 3.3). O índice do *halftone* é utilizado para saber qual

máscara dentro do *buffer* de máscaras será utilizada para o *pixel* em questão. O *buffer* de máscaras é um vetor onde são armazenadas as máscaras de *halftone* que poderão ser utilizadas durante a execução do algoritmo de *halftoning*.

Para se calcular o índice do *halftone* através da correção linear, basta dividir o valor de entrada do *pixel* por 255, visto que a imagem foi armazenada utilizando o sistema de cores *RGB*, onde cada *pixel* varia no intervalo de 0 a 255. Em seguida, multiplica-se este valor pela dimensão da máscara de *halftone* para se obter o índice do *halftone*.

Para se calcular o índice do *halftone* através da correção *gamma*, utiliza-se a fórmula citada e explicada anteriormente no capítulo dois (item 2.1.1). Sendo assim, foi adotado o valor inicial de $\gamma = 0.25$ e é permitido que o usuário varie esse valor durante a execução da aplicação. Primeiramente, calcula-se o inverso do valor de *gamma* e em seguida, divide-se o valor de entrada do *pixel* por 255. Eleva-se o resultado dessa divisão pelo inverso do valor de *gamma*, e por fim, multiplica-se o resultado pela dimensão da máscara de *halftone*.

A escolha entre utilizar a correção linear ou a correção *gamma* é feita pelo usuário.

```
int linearLevel( int cinza ) {

    return (int)(((float)cinza/255.0f) * (float)htLevels*(float)htLevels );

}
```

Figura 3.2: Cálculo da correção linear

```
float _gamma = 0.25;
int gammaCorrection( int cinza ){

    float gammaCorrection = 1.0f/_gamma;
    float temp = pow(((float)cinza/255.0f), gammaCorrection) * (float)htLevels*(float)htLevels;

    return (int)temp;
}
```

Figura 3.3: Cálculo da correção *gamma*

Como dito anteriormente, com o índice calculado, pode-se saber qual máscara, no *buffer* de máscaras, será utilizada para o *pixel* em questão. Neste momento, o usuário pode escolher entre utilizar o *buffer* de máscaras estáticas ou o *buffer* de máscaras aleatórias. O *buffer* de máscaras estáticas são máscaras pré-definidas em um arquivo, já o *buffer* de máscaras aleatórias é preenchido por máscaras calculadas aleatoriamente no momento em que se ativa o uso da máscara aleatória.

Então, aplica-se a máscara selecionada no *pixel* em questão e na sua vizinhança, isto é, caso a posição na máscara possua valor igual a um, o *pixel* em questão receberá o valor 255 no modelo de cores *RGB* (Figura 3.4).

Na Figura 3.4 pode-se visualizar a parte principal do algoritmo de *halftoning*. Primeiro, tem-se as duas iterações mais externas (uma iteração no eixo vertical da imagem e a outra no eixo horizontal), que representam a varredura dentro da região calculada a partir da posição do ponteiro do mouse sobre a imagem. Em seguida, lê-se o valor do *pixel* na imagem através da função *cvGet2D*() da biblioteca *OpenCV*. Então, verifica-se se o usuário ativou o uso da correção *gamma*. Se sim, calcula-se o índice do *halftone* utilizando a correção *gamma* através da função g*ammaCorrection*(), se não, calcula-se o índice utilizando a correção linear através da função *linearLevel*(). Com o índice calculado, seleciona-se a máscara correspondente ao índice dentro do *buffer* de máscaras. Por fim, são feitas duas iterações que percorrem tanto no eixo vertical como horizontal da imagem, aplicando a máscara selecionada no *pixel* em questão e em sua vizinhança.

```
for( int y = 0 ; y < htNumPxlsY ; y++ ) {

    for( int x = 0 ; x < htNumPxlsX ; x++ ) {

        CvScalar pixel = cvGet2D( img, img->height-1-(minRangeY+y), minRangeX+x );

        // Calculo do indice do halftone a ser usado
        if( useGamma )
            idx = gammaCorrection( pixel.val[0] );
        else
            idx = linearLevel( pixel.val[0] );

        //Seleciona a mascara escolhida no buffer de máscaras
        unsigned char *htp = htPtr+idx*htLevels*htLevels;

        int htyy, ly;
        for( htyy = y*htLevels, ly = 0 ; ly < htLevels ; ly++, htyy++ ) {

            int htxx, lx;
            for( htxx = x*htLevels, lx = 0 ; lx < htLevels ; lx++, htxx++ ) {

                if( htp[ lx + ly*htLevels ] == 1 )
                    buf[ htxx + (htRes-1-htyy)*htRes ] = 255;
```

Figura 3.4: Aplicação da máscara de *halftone* na região da imagem

3.2. As Máscaras

As máscaras estáticas utilizadas na implementação do algoritmo de *halftoning* neste trabalho foram obtidas a partir dos exemplos citados na referência [1]. Para representar computacionalmente estas máscaras, adotou-se a seguinte representação: se na máscara o *pixel* estiver pintado, ou seja, na cor preta, este *pixel* será representado com o valor um; e se o *pixel* estiver na cor branca, este será representado com o valor zero. Na

Tabela 3.1 tem-se a escala de cinza utilizada para máscaras estáticas de dimensão 4x4, gerando o total de 17 tons de cinza.

Tabela 3.1: Tabela contendo as máscaras de *halftone* de dimensão 4x4 utilizadas no *buffer* de máscaras estáticas

0	0	0	0												
0	0	0	0												
0	0	0	0												
0	0	0	0												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0	1	1	1	0	1	1	1	0
0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1

Para as máscaras de dimensão 6x6 segue-se a mesma ideia de preenchimento dos pontos pretos, porém usando uma célula com seis linhas e seis colunas, gerando uma escala de cinza com 37 tons de cinza.

No caso do *buffer* de máscaras aleatórias, no momento em que o usuário ativa esta funcionalidade, este *buffer* é preenchido com máscaras onde o preenchimento dos pontos pretos é feito de forma aleatória (Figura 3.5), permitindo assim a diminuição de padrões de impressão no resultado final. A construção de uma máscara aleatória é feita através do procedimento *getRandomMask*() que pode ser visualizado na Figura 3.5. Neste procedimento, primeiro inicializa-se a máscara com zeros, em seguida, inicializa-

se o *buffer* auxiliar com valores de zero até o limite do *buffer* menos um, onde este limite é o número de elementos da matriz que representa a máscara (por exemplo, caso a máscara possua uma resolução de 4x4, o limite do *buffer* auxiliar será de 16-1 = 15). Este *buffer* auxiliar é um vetor que permite a geração de números aleatórios distintos no intervalo de zero até o limite do *buffer*, otimizando assim, a construção da máscara, pois o algoritmo não gastará tempo gerando números repetidos. Para isso, quando um número aleatório é gerado, a posição no vetor equivalente ao valor do número aleatório é eliminada e assim este número não poderá ser gerado novamente.

Por fim, a partir do índice do *halftone* calculado anteriormente, gera-se uma quantidade de números aleatórios igual ao valor do índice do *halftone* e assim, a máscara é preenchida com o valor um em cada posição indicada pelo valor dos números aleatórios gerados. Como por exemplo, caso o número aleatório seja igual a dois, o segundo elemento da matriz que representa a máscara receberá o valor um.

A construção do *buffer* de máscaras aleatórias é feita através do procedimento *generateRandomMaskBuffer*() que pode ser visualizado na Figura 3.5. Neste procedimento, primeiro calcula-se o tamanho do *buffer*, que é obtido a partir da resolução da máscara através da seguinte fórmula:

$$bufferSize = \big((htLevel \times htLevel) + 1\big) \times htLevel \times htLevel$$

Onde:

bufferSize = tamanho do *buffer* de máscaras aleatórias

htLevel = resolução da máscara aleatória

Com o tamanho do *buffer* calculado, inicia-se o preenchimento do mesmo com as máscaras aleatórias. O *buffer* é preenchido com o número de máscaras igual ao número máximo de tons de cinza que a resolução da máscara permite, sendo assim, cada máscara equivale a um tom cinza na escala. E esses tons de cinza variam do totalmente branco (máscara com todos os elementos igual a zero) ao totalmente preto (máscara com todos os elementos igual a um). Por fim, o *buffer* de máscaras aleatórias é salvo em um arquivo de texto através do procedimento *saveMask*().

```
void generateRandomMaskBuffer(int htLevels){

    int size = htLevels*htLevels;
    unsigned char mask[size];
    unsigned char buffer[((htLevels*htLevels)+1)*htLevels*htLevels];

    for (int i = 0; i < size+1;i++)
    {
        getRandomMask(i, htLevels, mask);
        for (int j = 0; j < size; j++)
        {
            buffer[(i*size)+j] = mask[j];
        }
    }
    saveMask(buffer, htLevels);
}
void getRandomMask(int idx, int htLevels, unsigned char randomMask[]){

    int size = htLevels*htLevels;
    unsigned char buffer[size];
    int bufferLimit = size;
    int maskLimit = 0;
    int n;
    unsigned char randomNbr;

    //Initialize randomMask with zeros
    for (int i = 0; i < size; i++){
        randomMask[i] = 0;
    }

    //Initialize buffer with values from 0 to buffer limit
    for (int i = 0; i  < size; i++){
        buffer[i] = i;
    }

    //Initialize seed
    my_rand_init();

    //Calculate distincts random positions to populate the mask
    while (maskLimit < idx){
        n = my_distinct_rand(bufferLimit);
        randomNbr = buffer[n];
        buffer[n] = buffer[bufferLimit-1];
        //Set random position calculated to 1
        randomMask[randomNbr] = 1;
        bufferLimit--;
        maskLimit++;
    }
}
```

Figura 3.5: Geração do *buffer* de máscaras aleatórias

Na Tabela 3.2 tem-se um exemplo de várias outras combinações possíveis de uma escala de cinza gerada a partir de máscaras aleatórias com dimensão 4x4 através do algoritmo mostrado anteriormente na Figura 3.5.

Tabela 3.2: Tabela contendo exemplos de máscaras aleatórias com dimensão 4x4

0	0	0	0												
0	0	0	0												
0	0	0	0												
0	0	0	0												
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0
0	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1
0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	1
1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Capítulo 4

4. Resultados

Neste capítulo iremos mostrar e discutir os resultados obtidos a partir da implementação do algoritmo de *halftoning* utilizando máscaras aleatórias para imprimir imagens em tons de cinza em alta qualidade em impressoras a laser que imprimam somente em preto e branco.

Primeiramente iremos comparar os resultados obtidos a partir da aplicação da técnica de *halftoning* em uma imagem em tons de cinza utilizando o *buffer* de máscaras estáticas com os resultados obtidos da aplicação do *halftoning* na mesma imagem utilizando o *buffer* de máscaras aleatórias. Ambos utilizando somente a correção linear. Em seguida, compararemos os resultados obtidos a partir da aplicação do *halftoning* com máscaras aleatórias utilizando a correção linear com os resultados obtidos da aplicação do *halftoning* com máscaras aleatórias utilizando a correção *gamma* para suavizar a transição entre os tons de cinza da imagem resultante.

Sendo assim, à direita da Figura 4.1 podemos observar o primeiro resultado utilizando um *halftoning* com resolução de 300 *pixels* e uma escala de cinza gerada a partir de máscaras estáticas com dimensão 4x4, descritas anteriormente. Fica visível a presença de um padrão de impressão em linhas horizontais e verticais na imagem resultante. Já na Figura 4.2 podemos observar o resultado da aplicação do *halftoning* com uma resolução de 300 *pixels* e utilizando máscaras aleatórias com uma dimensão de 4x4. Ao compararmos este resultado com o mostrado na Figura 4.1, onde o *halftoning* foi aplicado utilizando a mesma resolução e a mesma dimensão de máscara, porém estática, podemos observar uma ligeira diminuição na presença de padrões de impressão na imagem resultante, gerando assim um resultado melhor, isto é, mais próximo da imagem original em tons de cinza.

Figura 4.1: A esquerda, a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado utilizando uma resolução de 300 *pixels* e uma máscara estática com dimensão 4x4.

Figura 4.2: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde foi aplicado o *halftoning* com uma resolução de 300 *pixels* e máscaras aleatórias com dimensão 4x4.

Na Figura 4.3 podemos visualizar o resultado da aplicação do algoritmo de *halftoning* com uma resolução de 600 *pixels* e também uma escala de cinza gerada a partir do *buffer* de máscaras estáticas com dimensão 4x4. É possível visualizar uma melhora significativa na impressão dos detalhes da imagem, se comparado ao resultado anterior, devido à mudança de resolução de 300 *pixels* para 600 *pixels*, entretanto ainda é visível a presença de padrões de impressões em linhas verticais e horizontais na imagem resultante. Já na Figura 4.4 temos o resultado obtido a partir do algoritmo de *halftoning* aplicado utilizando uma resolução de 600 *pixels* e máscaras aleatórias com dimensão 4x4. Ao compararmos este resultado com o mostrado na Figura 4.3, onde o *halftoning* foi aplicado utilizando a mesma resolução e a mesma dimensão de máscara estática, podemos observar uma melhora na diminuição de padrões de impressão na imagem resultante.

Figura 4.3: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde foi aplicado o *halftoning* com uma resolução de 600 *pixels* e máscaras estáticas com dimensão de 4x4.

Figura 4.4: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde foi aplicado o *halftoning* com uma resolução de 600 *pixels* e máscaras aleatórias com dimensão 4x4.

Já na Figura 4.5 podemos observar o resultado obtido da aplicação do *halftoning* com uma resolução de 300 *pixels* e uma escala de cinza gerada a partir de máscaras estáticas com dimensão 6x6. Podemos perceber uma leve melhora na impressão dos detalhes da imagem se comparado ao resultado mostrado na Figura 4.1, onde se tinha a mesma resolução de *halftoning*, porém utilizava máscaras com uma dimensão menor, 4x4. Entretanto, fica mais visível a presença de padrões de impressão no formato de pontos. Agora na Figura 4.6 podemos visualizar o resultado do *halftoning* aplicado em uma imagem em tons de cinza utilizando uma resolução de 300 *pixels* e uma escala de cinza gerada a partir do uso de máscaras aleatórias com uma dimensão de 6x6. Se compararmos este resultado com o mostrado na Figura 4.5, onde o *halftoning* foi aplicado com a mesma resolução e utilizando máscaras estáticas com a mesma dimensão, podemos perceber uma melhora significativa na diminuição dos padrões de impressão na imagem resultante, gerando assim uma imagem mais próxima da original.

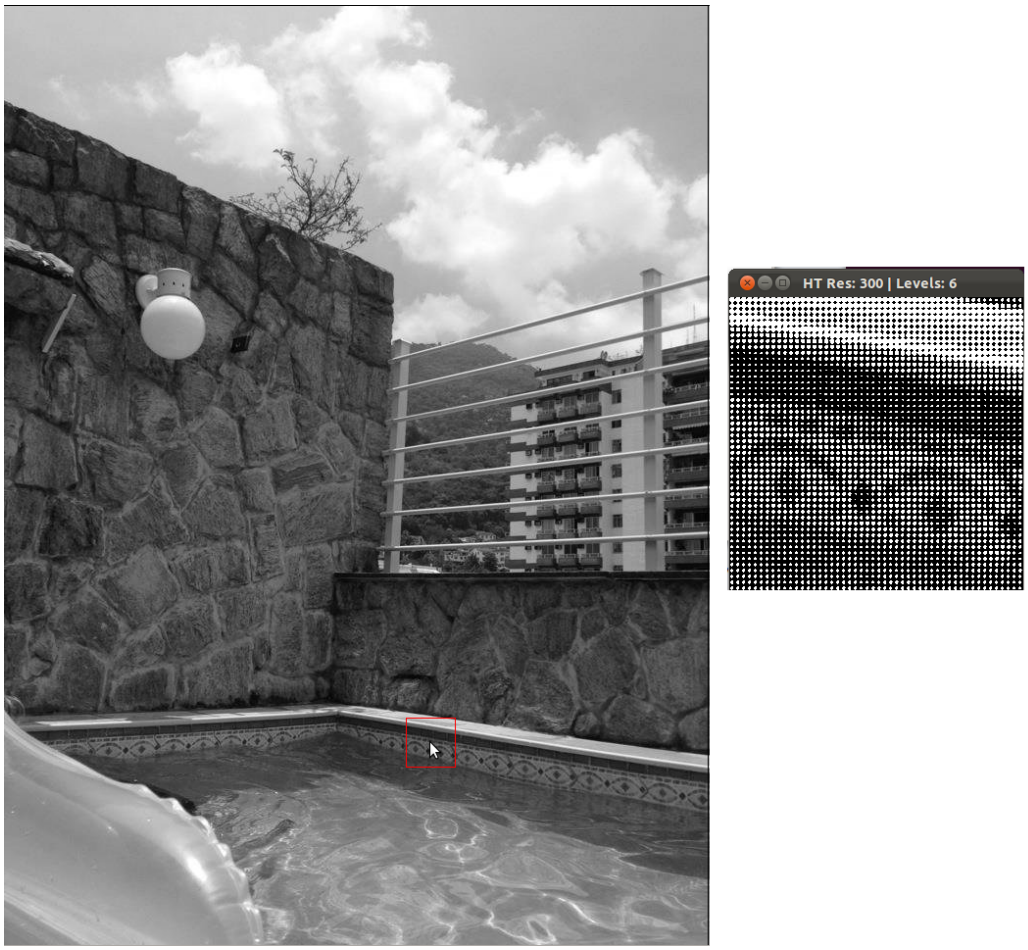


Figura 4.5: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde foi aplicado o *halftoning* com uma resolução de 300 *pixels* e utilizando máscaras estáticas com dimensão de 6x6

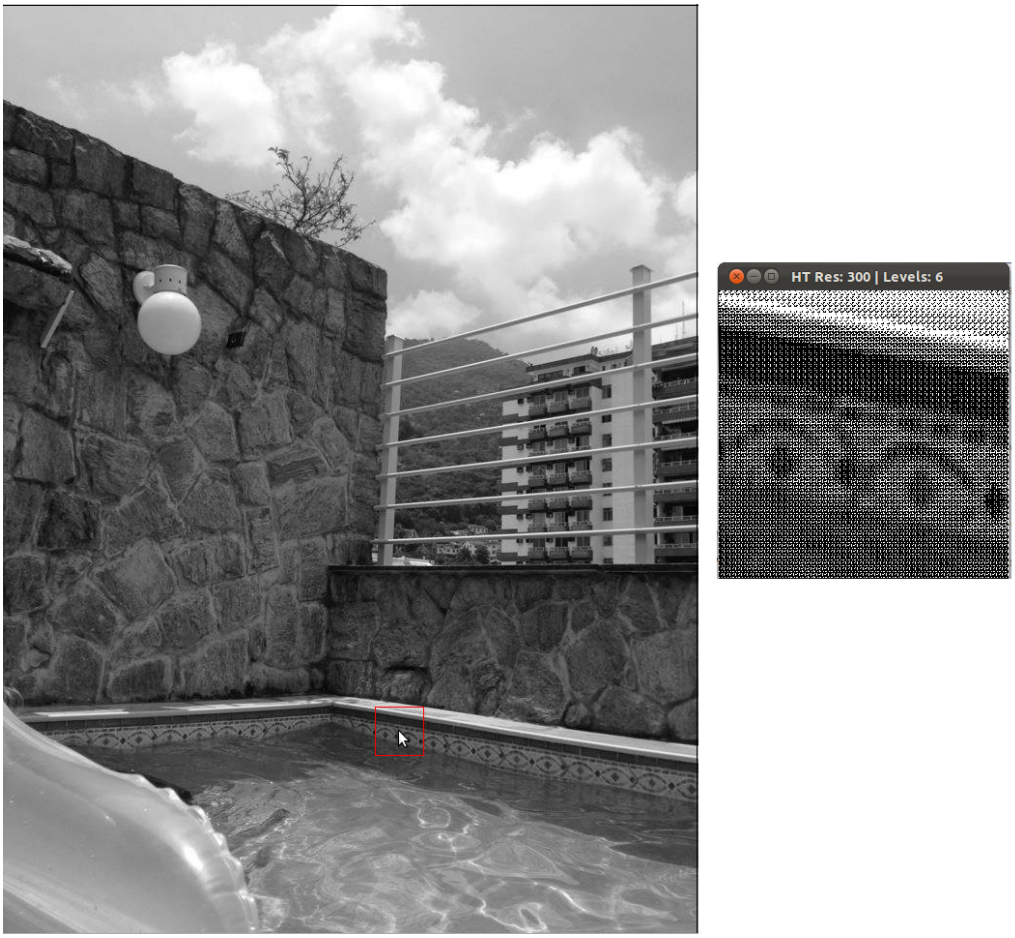


Figura 4.6: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado utilizando uma resolução de 300 *pixels* e máscaras aleatórias com dimensão 6x6.

E na Figura 4.7 visualizamos o resultado do algoritmo de *halftoning* com uma resolução de 600 *pixels* e utilizando uma escala de cinza gerada a partir do *buffer* de máscaras estáticas com uma dimensão de 6x6. Se comparado ao resultado mostrado na Figura 4.3, que utiliza a mesma resolução de *halftoning*, 600 *pixels*, tem-se uma leve melhora na qualidade da impressão dos detalhes da imagem, porém fica mais visível a presença de padrões de impressão na imagem resultante. Na Figura 4.8 temos o resultado obtido da aplicação da técnica de *halftoning* com uma resolução de 600 *pixels* e utilizando máscaras aleatórias com dimensão 6x6 para gerar uma escala de cinza. Ao compararmos este resultado com o visualizado na Figura 4.7, onde foi aplicado um *halftoning* utilizando a mesma resolução e máscaras estáticas com a mesma dimensão, podemos observar uma melhora significativa na qualidade da imagem resultante e também uma diminuição na presença de padrões de impressão na mesma.

Figura 4.7: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde foi aplicado o *halftoning* utilizando uma resolução de 600 *pixels* e máscaras estáticas com dimensão de 6x6.

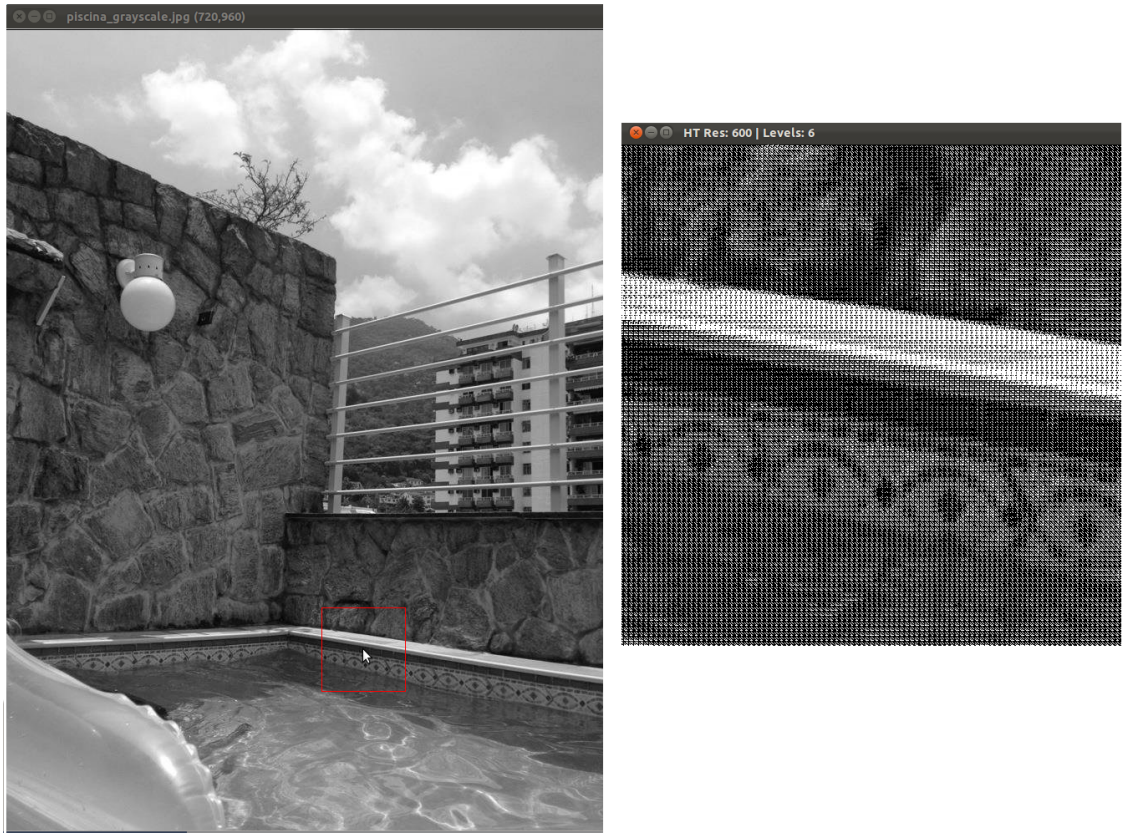


Figura 4.8: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado com uma resolução de 600 *pixels* e utilizando máscaras aleatórias com dimensão 6x6.

Mostraremos a seguir os resultados da aplicação do algoritmo de *halftoning* utilizando uma escala de cinza gerada a partir de máscaras aleatórias e utilizando também a correção *gamma* para suavizar a transição entre os tons de cinza. Na Figura 4.9 podemos visualizar uma comparação entre os tons de cinza gerados a partir do uso da correção linear e da correção *gamma* no algoritmo de *halftoning*. Podemos observar que na parte superior da Figura 4.9, que demonstra a escala de cinza gerada com o uso da correção *gamma*, a transição entre os tons de cinza ocorre de forma mais suave do que na parte inferior da Figura 4.9, que é a escala de cinza gerada com o uso da correção linear. Isso ocorre devido ao fato de que na escala de tons de cinza gerada utilizando a correção *gamma*, a transição entre os tons de cinza mais escuros se dá de forma mais comprimida. E nos tons de cinza mais claros a transição entre esses tons é expandida.

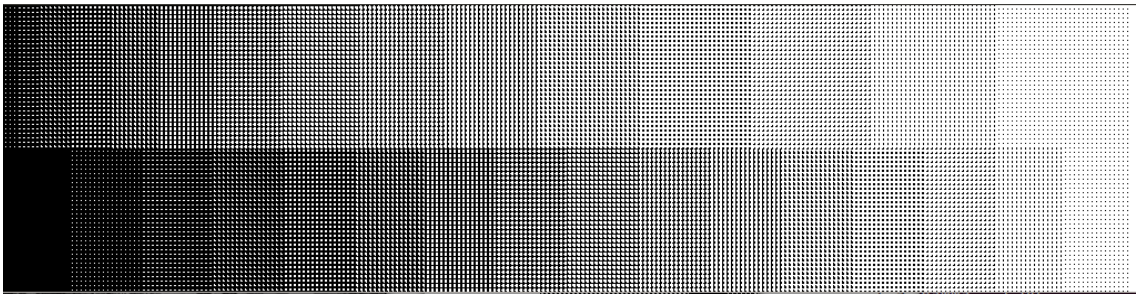


Figura 4.9: Comparação entre as escalas de cinza geradas a partir do uso da correção *gamma* (metade superior da imagem) e da correção linear (metade inferior da imagem) no algoritmo de *halftoning*.

Para os resultados que serão mostrados a seguir foi utilizada a correção *gamma* com $\gamma = 1.5$.

Sendo assim, primeiramente temos na Figura 4.10 o resultado obtido a partir da aplicação do algoritmo de *halftoning* com uma resolução de 300 *pixels*, utilizando máscaras aleatórias com dimensão 4x4 e a correção *gamma*. Se compararmos este resultado com o resultado mostrado anteriormente na Figura 4.2, onde o *halftoning* foi aplicado utilizando a mesma resolução e a mesma dimensão de máscara, podemos observar uma suavização na transição entre os tons de cinza.

Figura 4.10: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado com uma resolução de 300 *pixels*, máscaras aleatórias com dimensão 4x4 e aplicação da correção *gamma*, onde $\gamma = 1.5$.

Na Figura 4.11 temos o resultado da aplicação do *halftoning* com uma resolução de 600 *pixels* utilizando uma escala de cinza gerada a partir de máscaras aleatórias com dimensão 4x4 e o uso da correção *gamma*. Comparando este resultado com o resultado mostrado na Figura 4.4, onde o *halftoning* aplicado utiliza a mesma resolução e dimensão de máscara aleatória, podemos perceber uma melhora na suavização na transição entre os tons de cinza da imagem resultante, gerando assim um resultado mais próximo da imagem original.

Figura 4.11: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado utilizando uma resolução de 600 *pixels*, máscaras aleatórias com dimensão de 4x4 e uso da correção *gamma*, com $\gamma = 1.5$.

Agora na Figura 4.12 temos o resultado da aplicação do algoritmo de *halftoning* com uma resolução de 300 *pixels*, utilizando máscaras aleatórias com uma dimensão de 6x6 e o uso da correção *gamma*. Se compararmos este resultado com o resultado mostrado na Figura 4.6, onde o *halftoning* foi aplicado utilizando a mesma resolução e a mesma dimensão de máscara aleatória, podemos observar uma suavização na transição entre os tons de cinza da imagem resultante. Entretanto houve uma leve piora na qualidade da impressão dos detalhes da imagem, devido à máscara aleatória gerada ser pior do que a utilizada no resultado visualizado na Figura 4.6. Por isso se faz necessária a realização de testes exaustivos para se escolher as melhores máscaras aleatórias.

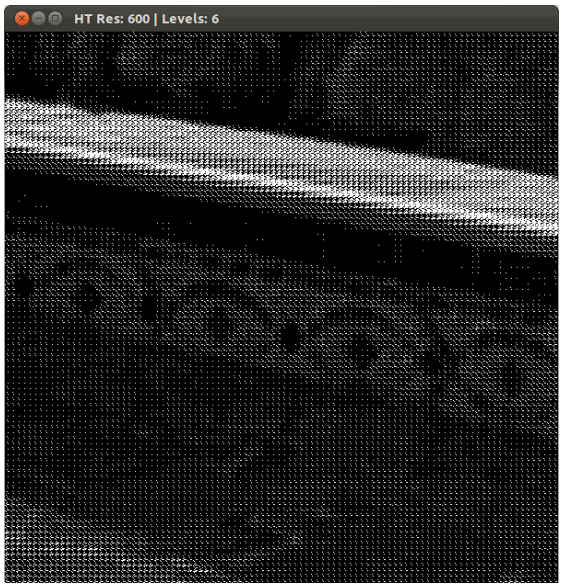
Figura 4.12: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado utilizando uma resolução de 300 *pixels*, máscaras aleatórias com dimensão 6x6 e correção *gamma* com $\gamma = 1.5$.

Por fim, temos na Figura 4.13 o resultado obtido a partir da aplicação da técnica de *halftoning* com uma resolução de 600 *pixels*, utilizando uma escala de cinza gerada a partir do uso do *buffer* de máscaras aleatórias com dimensão 6x6 e a correção *gamma*. Comparando este resultado com o resultado mostrado na Figura 4.8, onde o *halftoning* foi aplicado utilizando a mesma resolução e a mesma dimensão de máscara aleatória, podemos observar que houve uma melhora na transição entre os tons de cinza da imagem resultante.

Figura 4.13: À esquerda a imagem original em tons de cinza e à direita a região da imagem onde o *halftoning* foi aplicado com uma resolução de 600 *pixels*, máscaras aleatórias com uma dimensão de 6x6 e uso da correção *gamma* com $\gamma = 1.5$.

A seguir, na Tabela 4.1 temos o resultado da variação do fator γ durante o uso da correção *gamma* enquanto o algoritmo de *halftoning* é aplicado em uma imagem em tons de cinza. Para isso, foram utilizados os seguintes valores: $\gamma = 0.5$, $\gamma = 1.0$, $\gamma = 1.5$, $\gamma = 2.0$. Podemos observar o efeito da suavização na transição entre os tons de cinza da imagem resultante. Além disso, note que para $\gamma = 1.0$, imagem (b) da Tabela 4.1, equivale a utilizar a correção linear, visto que a curva da correção *gamma* se torna uma linha reta. Para valores de *gamma* inferiores a um, temos o efeito de escurecimento da imagem, como pode ser notado na imagem (a) da Tabela 4.1. Já para valores superiores a um, temos o efeito de clareamento da imagem, como pode ser visualizado nas imagens (c) e (d) da Tabela 4.1.

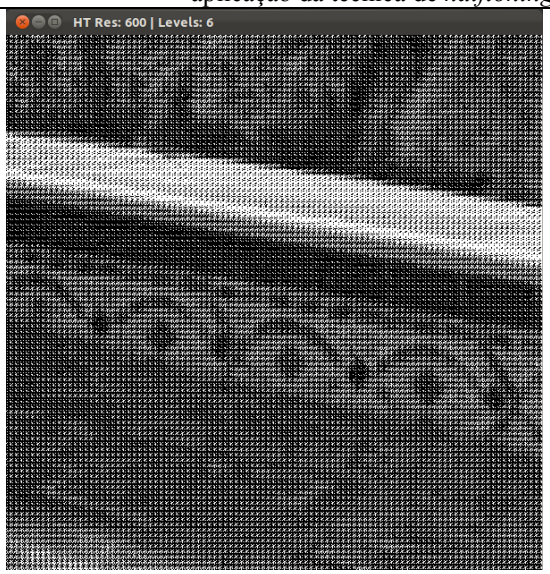
Tabela 4.1: Comparação dos resultados da variação do valor de *gamma* no uso da correção *gamma* durante a aplicação do algoritmo de *halftoning* em uma imagem em tons de cinza. Valores *gamma* utilizados: 0.5, 1.0, 1.5 e 2.0.

	
(a) $\gamma = 0.5$	(b) $\gamma = 1.0$
	
(c) $\gamma = 1.5$	(d) $\gamma = 2.0$

Agora, na Tabela 4.2, temos o resultado da variação de máscaras aleatórias no *buffer* durante a aplicação da técnica de *halftoning* em uma imagem em tons de cinza com o intuito de diminuir a presença de padrões de impressão na imagem resultante. Note que são apresentados seis resultados diferentes, sendo que em todos eles foram utilizadas as mesmas características, isto é, resolução de *halftoning* de 600 *pixels*, dimensão de

máscara aleatória de 6x6 e correção linear. Podemos observar que os resultados visualizados nas imagens (a), (c), (d) e (e) da Tabela 4.2 são os que apresentaram melhor resultado, pois a presença de padrões de impressão é bem inferior à presença de padrões visualizada nas imagens (b) e (f) da Tabela 4.2.

Tabela 4.2: Comparação entre diferentes máscaras aleatórias para gerar uma escala de cinza durante a aplicação da técnica de *halftoning* em uma imagem em tons de cinza.

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

	
(e)	(f)

Capítulo 5

5. Conclusões

Neste capítulo apresentamos as conclusões obtidas a partir do estudo e do desenvolvimento deste trabalho, como também as conclusões a partir dos resultados obtidos mostrados no capítulo anterior.

Podemos concluir a partir dos resultados obtidos que o algoritmo de *halftoning* desenvolvido pode ser considerado satisfatório, pois os resultados estão dentro do que se era esperado do uso da técnica de *halftoning*. Além disso, o uso da correção *gamma* se mostrou uma forma eficaz de melhorar os resultados, permitindo que haja uma suavização na transição entre os tons de cinza e assim gerar um resultado visualmente, ao olho humano, mais próximo da imagem original.

Podemos concluir também que o uso do *buffer* de máscaras aleatórias para gerar uma escala de cinza durante a aplicação do algoritmo de *halftoning* em uma imagem em tons de cinza é uma forma eficiente de reduzir a presença de padrões de impressão na imagem resultante e assim imprimir uma imagem com uma qualidade melhor. O algoritmo desenvolvido neste trabalho permite que o usuário escolha se quer gerar um novo *buffer* de máscaras aleatórias, salvar um *buffer* em um arquivo ou carregar um arquivo contendo um *buffer* de máscaras aleatórias. Com isso, é possível avaliar quais máscaras aleatórias foram mais eficazes e assim, reutilizá-las posteriormente.

A partir dos resultados mostrados no capitulo anterior podemos deduzir que o uso da correção *gamma* juntamente com o *buffer* de máscaras aleatórias durante a aplicação da técnica de *halftoning* em uma imagem em tons de cinza permite a obtenção de resultados melhores, ou seja, mais fiel à imagem original, pois estes resultados apresentam menos padrões de impressão e a transição entre os tons de cinza se dá de forma mais suave.

Além disso, podemos verificar que se comprovou que, conforme a teoria da técnica de *halftoning*, quanto maior for a resolução utilizada, melhor será o resultado da aplicação do algoritmo. Isso se comprovou nos resultados obtidos mostrados na Figura

4.3, Figura 4.4, Figura 4.7, Figura 4.8, Figura 4.11, Figura 4.13, pois nestes resultados foi utilizada a resolução de 600 *pixels*. Podemos concluir também que em alguns casos, o uso da dimensão 4x4 nas máscaras de *halftone* gerou resultados melhores do que as máscaras com dimensão 6x6, que permitiram uma maior percepção visual de padrões de impressão na imagem resultante. Porém isto depende da resolução do *halftoning* que está sendo utilizada.

Entretanto este fato pode ser devido ao uso de máscaras ruins com uma dimensão de 6x6, pois se observarmos os resultados mostrados na Figura 4.4 e Figura 4.8, o resultado da aplicação do *halftoning* utilizando uma resolução de 600 *pixels* e máscaras aleatórias com dimensão de 6x6 (Figura 4.8) gerou uma imagem melhor do que o resultado obtido a partir da aplicação do *halftoning* com uma resolução de 600 *pixels* e máscaras aleatórias com dimensão 4x4 (Figura 4.4).

Podemos observar também que em alguns casos o uso da correção *gamma* gerou resultados com uma qualidade inferior, como visto nos resultados visualizados na Figura 4.2 e Figura 4.10, onde apesar da transição entre os tons de cinza estar mais suave, o uso da correção *gamma* permitiu uma visualização acentuada da presença de padrões de impressão na imagem resultante. Sendo assim, podemos concluir que para se obter um resultado o mais fiel possível à imagem original em tons de cinza, se faz necessário encontrar a melhor combinação entre resolução de *halftoning*, dimensão de máscara de *halftone*, qualidade da máscara e valor do fator *gamma* usado na correção *gamma*.

Constatamos alguns pontos passíveis de melhoria no algoritmo, o primeiro seria modificar o algoritmo de forma que o *halftoning* possa ser aplicado em outras resoluções além de 300 *pixels* e 600 *pixels* para assim se obter melhores resultados. Segundo seria permitir outras dimensões de máscaras estáticas de *halftone*, como também outras máscaras estáticas, visto que não se obteve bons resultados com a máscara estática de dimensão 6x6. Para as máscaras aleatórias não há necessidade disso, pois atualmente o código constrói o *buffer* de máscaras aleatórias de acordo com os argumentos passados pelo usuário, ou seja, é possível gerar máscaras aleatórias com outras dimensões, porém foram demonstrados apenas os resultados com máscaras aleatórias de dimensão 4x4 e 6x6 para efeito de comparação aos resultados do *buffer* de máscaras estáticas.

Não obtivemos problemas de desempenho durante a execução dos testes e resultados mostrados nesse trabalho, visto que os resultados foram obtidos em tempo real. Estes

testes foram executados em uma máquina com as seguintes configurações: processador Intel Core i5, memória RAM de 4GB, placa de vídeo NVIDIA GeForce GTS 250.

Referências Bibliográficas

1. RUSS, J. C. The Image Processing Handbook. 5. ed. Boca Raton: Taylor & Francis Group, 2007.
2. A Standard Default Color Space for the Internet - sRGB. w3, 1996. Disponivel em: <http://www.w3.org/Graphics/Color/sRGB>. Acesso em: 05 Julho 2013.
3. SCHILDT, H. C Completo e Total. 1. ed. São Paulo: McGraw-Hill, 1991.
4. HEARN, D.; BAKER, M. P. Computer Graphics with OpenGL. 3. ed. Upper Sadle River: Pearson Prentice Hall, 2004.
5. BRADSKI, G.; KAEHLER, A. Learning OpenCV. 1. ed. Sebastopol: O'Reilly Media, 2008.
6. GLUT - The OpenGL Utility Toolkit. OpenGL, 1997. Disponivel em: <http://www.opengl.org/resources/libraries/glut/>. Acesso em: 08 Julho 2013.
7. ADOBE SYSTEMS INCORPORATED. PostScript language reference manual. 3. ed. New York: Addison-Wesley Publishing Company, 1999.
8. DAVE SHREINER, G. S. J. K. B. L.-K. The OpenGL Programming Guide. 8. ed. New York: Addison Wesley, 2013.
9. STROUSTRUP, B. Programming**:** Principles and Practice Using C++. 1. ed. New York: Addison-Wesley, 2009.
10. SOBELL, M. G. A Practical Guide to Ubuntu Linux. 2. ed. Boston: Pearson Education, 2008.
11. XEROX Document Management, Digital Printing Equipment, Business Process Outsourcing. Xerox. Disponivel em: <http://www.xerox.com/>. Acesso em: 02 Agosto 2013.
12. RANDALL, P. G. The Laser Printer Reference**:** the Complete Guide to Hp Laserjet Printers and Compatible Printers. 2. ed. [S.l.]: Brady, 1993.
13. ADOBE SYSTEMS INCORPORATED. Adobe. Adobe, 1982. Disponivel em: <http://www.adobe.com/>. Acesso em: 02 Agosto 2013.
14. SHARMA, G. Digital Color Imaging Handbook. [S.l.]: CRC Press, 2003.
15. ARNEY, J. S.; ANDERSON, P. G. Optimizing Halftone Masks with Genetic Algorithms and a Printer Model. Proceedings of the International Conference on

Digital Printing Technologies (NIP 19), New Orleans, 2003.

16. ANDERSON, P. G. et al. A Genetic Algorithm Search for Improved Halftone Masks. Proceeding of ANNIE 2003: Articial Neural Networks in Engineering, St. Louis, 2003.
17. LAI, C.-C.; TSENG, D.-C. Printer Model and Least-Squares Halftoning Using Genetic Algorithms. Recent Progress in Digital Halftoning II, 1999. 363-373.
18. KANG, H. R. Frequency Analysis of Microcluster Halftoning. Recent Progress in Digital Halftoning II, 1999. 339-352.

Apêndice A

A. Código Fonte

A.1 Correção Linear e Correção *Gamma*

```
int linearLevel( int cinza ) {
    return (int)(((float)cinza/255.0f) * (float)htLevels*(float)htLevels );
}

float _gamma = 0.25;
int gammaCorrection( int cinza ){
    float gammaCorrection = 1.0f/_gamma;
    float temp = pow(((float)cinza/255.0f), gammaCorrection) * (float)htLevels*(float)htLevels;
    return (int)temp;
}
```

Código A.1: Código da correção linear e da correção *Gamma*

A.2 *Halftone* com Máscara Estática

```
void displayStaticHalftone( void ) {

    int idx;

    if( !htWindow ) return;

    int halfRegionX = htNumPxlsX/2;
    int halfRegionY = htNumPxlsY/2;

    if( currMouseX <= halfRegionX || currMouseX >= img1->width -halfRegionX ||
        currMouseY <= halfRegionY || currMouseY >= img1->height-halfRegionY )
        return;

    glutSetWindow( htWindow );
    glClear( GL_COLOR_BUFFER_BIT );

    int minRangeX = currMouseX - halfRegionX;
    int minRangeY = currMouseY - halfRegionY;

    unsigned char *buf = new unsigned char[ htRes * htRes ];

    // Set buf to black
    memset( buf, htRes*htRes*sizeof(unsigned char), 0 );

    IplImage *img = img1;
```

```
void displayStaticHalftone( void ) {

    int idx;

    if( !htWindow ) return;

    int halfRegionX = htNumPxlsX/2;
    int halfRegionY = htNumPxlsY/2;

    if( currMouseX <= halfRegionX || currMouseX >= img1->width -halfRegionX ||
        currMouseY <= halfRegionY || currMouseY >= img1->height-halfRegionY )
        return;

    glutSetWindow( htWindow );
    glClear( GL_COLOR_BUFFER_BIT );

    int minRangeX = currMouseX - halfRegionX;
    int minRangeY = currMouseY - halfRegionY;

    unsigned char *buf = new unsigned char[ htRes * htRes ];

    // Set buf to black
    memset( buf, htRes*htRes*sizeof(unsigned char), 0 );

    IplImage *img = img1;

    for( int y = 0 ; y < htNumPxlsY ; y++ ) {

        for( int x = 0 ; x < htNumPxlsX ; x++ ) {

            CvScalar pixel = cvGet2D( img, img->height-1-(minRangeY+y), minRangeX+x );

            // Calculo do indice do halftone a ser usado
            if( useGamma )
                idx = gammaCorrection( pixel.val[0] );
            else
                idx = linearLevel( pixel.val[0] );

            unsigned char *htp = htPtr+idx*htLevels*htLevels;

            int htyy, ly;
            for( htyy = y*htLevels, ly = 0 ; ly < htLevels ; ly++, htyy++ ) {

                int htxx, lx;
                for( htxx = x*htLevels, lx = 0 ; lx < htLevels ; lx++, htxx++ ) {

                    if( htp[ lx + ly*htLevels ] == 1 )
                        buf[ htxx + (htRes-1-htyy)*htRes ] = 255;

                }
            }

        }

    }

    glRasterPos2i( 0, 0 );
    glDrawPixels( htRes, htRes, GL_LUMINANCE, GL_UNSIGNED_BYTE, buf );

    glutSwapBuffers();

}
```

Código A.2: Código do *halftone* utilizando máscara estática

A.3 *Halftone* com Máscara Aleatória

```
void displayRandomMask (void){

    int idx;

    if( !htWindow ) return;


    int halfRegionX = htNumPxlsX/2;
    int halfRegionY = htNumPxlsY/2;

    if( currMouseX <= halfRegionX || currMouseX >= img1->width -halfRegionX ||
        currMouseY <= halfRegionY || currMouseY >= img1->height-halfRegionY )
        return;

    glutSetWindow( htWindow );
    glClear(GL_COLOR_BUFFER_BIT);

    int minRangeX = currMouseX - halfRegionX;
    int minRangeY = currMouseY - halfRegionY;

    unsigned char *buf = new unsigned char[ htRes * htRes ];

    // Set buf to black
    memset( buf, htRes*htRes*sizeof(unsigned char), 0 );

    IplImage *img = img1;

    //random mask buffer
    unsigned char rmb[((htLevels*htLevels)+1)*htLevels*htLevels];
    generateRandomMaskBuffer(htLevels);
    if (htLevels == 4){
        readMask("RMaskBuffer4.dat", rmb, htLevels);
    }
    else if (htLevels == 6){
        readMask("RMaskBuffer6.dat", rmb, htLevels);
    }

    for( int y = 0 ; y < htNumPxlsY ; y++ ) {

        for( int x = 0 ; x < htNumPxlsX ; x++ ) {

            CvScalar pixel = cvGet2D( img, img->height-1-(minRangeY+y), minRangeX+x );

            // Calculo do indice do halftone a ser usado
            if( useGamma )
                idx = gammaCorrection( pixel.val[0] );
            else
                idx = linearLevel( pixel.val[0] );

            unsigned char *htp = rmb+idx*htLevels*htLevels;
```

```
            int htyy, ly;
            for( htyy = y*htLevels, ly = 0 ; ly < htLevels ; ly++, htyy++ ) {

                int htxx, lx;
                for( htxx = x*htLevels, lx = 0 ; lx < htLevels ; lx++, htxx++ ) {

                    if( htp[ lx + ly*htLevels ] == 1 )
                        buf[ htxx + (htRes-1-htyy)*htRes ] = 255;

                }

            }

        }

    }

    glRasterPos2i( 0, 0 );
    glDrawPixels( htRes, htRes, GL_LUMINANCE, GL_UNSIGNED_BYTE, buf );

    glutSwapBuffers();

}
```

Código A.3: Código do *halftone* utilizando o *buffer* de máscaras aleatórias

A.4 Cálculo da Máscara Aleatória

```
void getRandomMask(int idx, int htLevels, unsigned char randomMask[]){

    int size = htLevels*htLevels;
    unsigned char buffer[size];
    int bufferLimit = size;
    int maskLimit = 0;
    int n;
    unsigned char randomNbr;

    //Initialize randomMask with zeros
    for (int i = 0; i < size; i++){
        randomMask[i] = 0;
    }

    //Initialize buffer with values from 0 to buffer limit
    for (int i = 0; i  < size; i++){
        buffer[i] = i;
    }

    //Initialize seed
    my_rand_init();

    //Calculate distincts random positions to populate the mask
    while (maskLimit < idx){
        n = my_distinct_rand(bufferLimit);
        randomNbr = buffer[n];
        buffer[n] = buffer[bufferLimit-1];
        //Set random position calculated to 1
        randomMask[randomNbr] = 1;
        bufferLimit--;
        maskLimit++;
    }
}
```

Código A.4: Código do cálculo da máscara aleatória

A.5 Criação do *Buffer* de Máscaras Aleatórias

```
void generateRandomMaskBuffer(int htLevels){

    int size = htLevels*htLevels;
    unsigned char mask[size];
    unsigned char buffer[((htLevels*htLevels)+1)*htLevels*htLevels];

    for (int i = 0; i < size+1;i++)
    {
        getRandomMask(i, htLevels, mask);
        for (int j = 0; j < size; j++)
        {
            buffer[(i*size)+j] = mask[j];
        }
    }
    saveMask(buffer, htLevels);
}
```

Código A.5: Código do criação do *buffer* de máscaras de aleatórias

A.6 Leitura do *Buffer* de Máscara de um Arquivo

```
void readMask(char* filename, unsigned char buffer[], int htLevels){
    FILE * pFile;
    long lSize;
    int bufferSize = ((htLevels*htLevels)+1)*htLevels*htLevels;

    pFile = fopen (filename, "r");
    if (pFile == NULL)
    {
        printf("Erro ao abrir arquivo da mascara aleatoria!!!\n\n");
        exit(1);
    }
    int i = 0;
    int c;
    unsigned char ch;
    do
    {
        c = fgetc(pFile);
        if (c != 32 && i < bufferSize)
        {
            if (c == 48)
                ch = 0;
            else if (c == 49)
                ch = 1;
            buffer[i] = ch;
            i++;
        }
    }while (c != EOF);
    fclose (pFile);
}
```

Código A.6: Código da leitura do *buffer* de máscaras de um arquivo

A.7 Salva o *Buffer* de Máscaras em um Arquivo

```
void saveMask(unsigned char* buffer, int htLevels){

    char filename[16];
    char level[10];

    //Converte int em char
    itoa(htLevels, level);
    //Cria nome do arquivo
    strcpy(filename,"RMaskBuffer");
    strcat (filename, level);
    strcat(filename,".dat");

    FILE *pFile;

    pFile = fopen (filename, "w");

    if (pFile == NULL)
    {
        printf("Erro ao abrir arquivo da mascara aleatoria!!!\n\n");
        exit(1);
    }

    int bufferSize = ((htLevels*htLevels)+1)*htLevels*htLevels;

    for (int i = 0; i < bufferSize; i++)
    {
        unsigned char temp = buffer[i];
        fprintf( pFile, "%d ", (int)temp );
    }

    if (ferror (pFile))
        printf ("Erro ao escrever no arquivo da mascara aleatoria\n");
    fclose (pFile);
}
```

Código A.7: Código que salva um *buffer* de máscaras em um arquivo