Universidade Estadual de Campinas

Instituto de computação

Introdução ao processamento de imagem digital

MC920

Trabalho #2

Nome: Leonardo de Alencar Lopes **RA:** 171928

1. Introdução

O objetivo deste trabalho é reduzir a quantidade de cores utilizadas para exibir uma imagem através de técnicas de meios-tons(pontilhados por difusão de erro, neste caso). Para isso, foi utilizada a linguagem Python 3 e a biblioteca de tratamento de imagens OpenCV.

Junto a este documento está o arquivo comprimido **171928-Trabalho-2.zip**. Nele há todos os scripts que foram implementados e serão citados ao longo deste texto, além das pastas **img**, que contém as imagens utilizadas pelos algorítmos, e **resultados**, que contém as imagens de *outputs*.

2. O programa

Para este projeto, foram implementadas funções que realizam a diminuição da quantidade de cores através de diferentes abordagens para a difusão de erro. Para cada uma das maneiras de distribuição do erro, foram implementadas funções que recebem como entrada uma imagem **f(x,y)**, a percorrem alterando os níveis de cinza dos pixels, propagando o erro para os subsequentes e gerando uma imagem **g(x,y)** na saída. Os padrões de distribuição utilizados foram propostos pelo documento de especificação do problema e podem ser visualizados a seguir:

	f(x,y)	7/16
3/16	5/16	1/16

Table a: Floyd e Steinberg

			f(x,y)		32/200	
12/200		26/200		30/200		16/200
	12/200		26/200		12/200	
5/200		12/200		12/200		5/200

Table b: Stevenson e Arce

		f(x,y)	8/32	4/32
2/32	4/32	8/32	4/32	2/32

Table c: Burkes

		f(x,y)	5/32	3/32
2/32	4/32	5/32	4/32	2/32
	2/32	3/32	2/32	

Table d: Sierra

		f(x,y)	8/42	4/42
2/42	4/42	8/42	4/42	2/42
1/42	2/42	4/42	2/42	1/42

Table e: Stucki

		f(x,y)	7/48	5/48
3/48	5/48	7/48	5/48	3/48
1/48	3/48	5/48	3/48	1/48

Table f: Jarvis, Judice e Ninke

Além dos diferentes padrões de distribuição de erros, também foram implementadas 2 ordens diferentes de realizar a varredura na imagem. A **ordem 1** realiza o calculo sobre os pixels exclusivamente da esquerda para a direita em cada linha. Essa abordagem pode levar a padrões indesejados ou direcionalidades na imagem de saída. Assim, a **ordem 2** alterna a forma como os pixels de cada linha são varridos, fazendo com que em linhas pares a ordem seja da esquerda para a direita e nas ímpares o oposto.

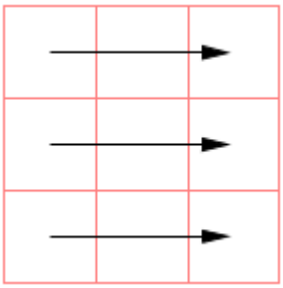

Figure 1: Ordem 1

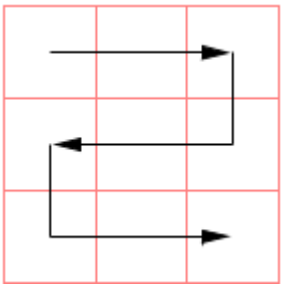

Figure 2: Ordem 2

Os scripts fornecidos junto com este documento aplicam as técnicas de meios-tons sobre uma imagem de entrada indicada pelo usuário(esta imagem deve estar presente na pasta **img**). Eles varrem a imagem das duas formas descritas anteriormente, mostram a imagem original, as duas imagens de saída(uma para cada ordem de varredura) e as salvam na pasta de resultados. Durante a execução, é mostrado no terminal a porção da imagem que já foi processada. Exemplo de execução de um script: *python3 stucki.py monalisa.png*.

Esses scripts foram desenvolvidos com base no pseudo-código apresentado em aula. Assim, os calculos são feitos de forma não vetorizada, o que pode gerar uma certa demora para serem totalmente executados. Dependendo da imagem, pode demorar algumas dezenas de segundos.

Além disso, foram criados os scripts *todos_com_monalisa.sh* e *todos_com_baboon.sh*, que automatizam a execução de todos os programas para as imagens *monalisa.png* e *baboon.png*, respectivamente.

3. Resultados

Figure 3: Original monalisa.png

Floyd e Steinberg

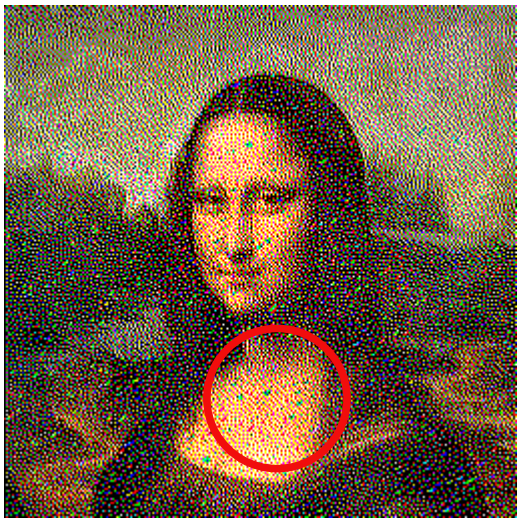

Figure 4: Floyd e Steinberg - Ordem 1

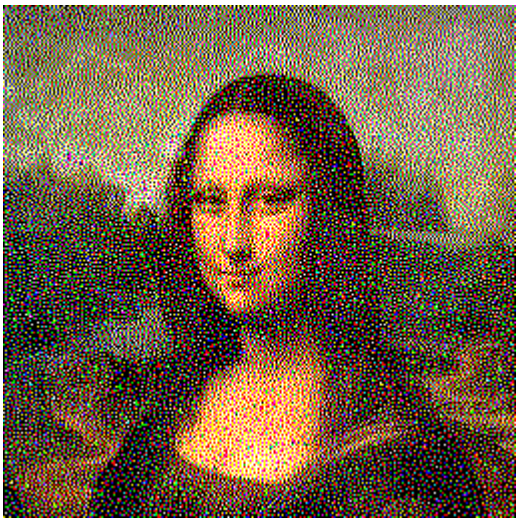

Figure 5: Floyd e Steinberg - Ordem 2

Aplicando a abordagem proposta por Floyd e Steinberg, é possível verificar que a transformação de uma imagem de tons de cores RGB para valores binários gera um certo padrão de mosaico, sendo nítida a diferença entre os pixels. A parte de vegetação, do lado esquerdo e do direito, possuem pontos verdes muito destacados, evidenciando a perda de contraste da imagem. Apesar disso, é possível identificar a imagem com facilidade. Esse problema deve ocorrer por conta da alta densidade de propagação de erros, que são divididos entre pixels muito próximos.

Com relação as duas diferentes formas de percorrer a imagem, a **Ordem 1** gerou muitos pontos verdes em um região sem essa cor(como pode ser notado pelo círculo vermelho). Como esse comportamento não foi notado na **Ordem 2**, provavelmente foi decorrido de erros acumulados provenientes da região de vegetação na parte esquerda, já na pela **Ordem 1** os erros são propagados da esquerda para baixo.

Stevenson e Arce

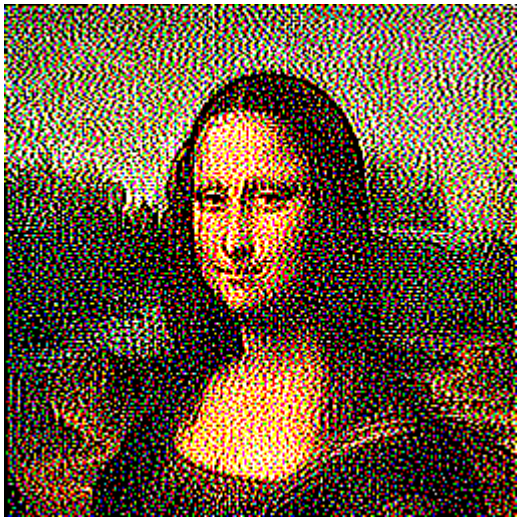

Figure 6: Stevenson e Arce - Ordem 1

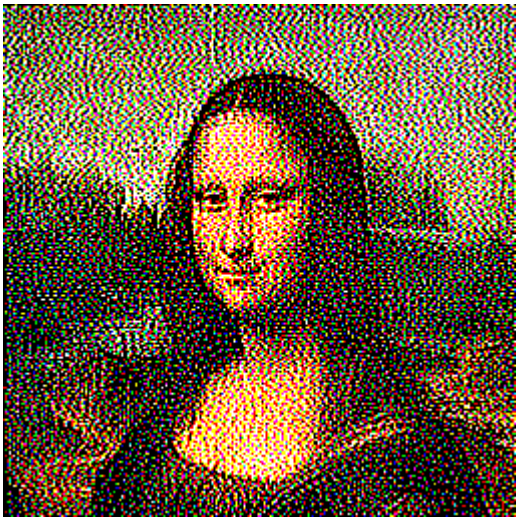

Figure 7: Stevenson e Arce - Ordem 2

Em Stevenson e Arce, o efeito de mosaico aumentou, dando a impressão de que a resolução de imagem diminui. Porém, não há diferenças notáveis na forma de percorrer a imagem. Essas caracteríscas provavelmente são provenientes do fato desta abordagem propagar seus erros para pixels mais distantes, que serão também mais diferentes entre sí, minimizando o carregamento do erro para distâncias mais longas.

Burkes

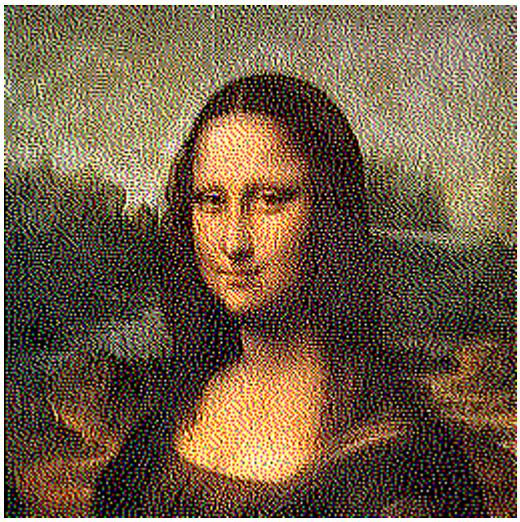

Figure 8: Burkes - Ordem 1

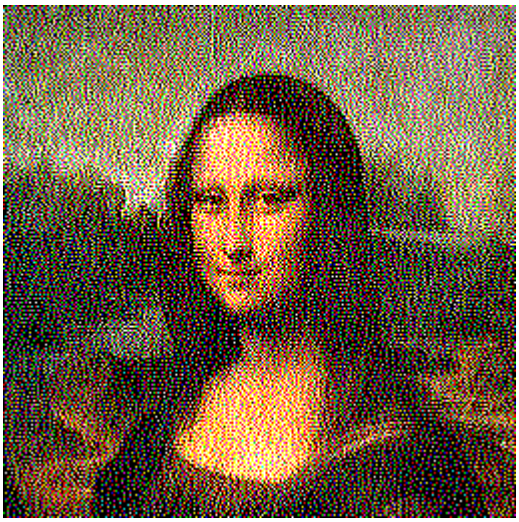

Figure 9: Burkes - Ordem 2

Burkes parece resolver os problemas de mosaico, presente em Stevenson e Arce, e o problema de excesso de propagação, em Floyd e Steinberg. Além disso, não é possível notar nenhum tipo de padrão muito acentuado gerado pelas difererentes ordens de percorrer as linhas da imagem.

Sierra

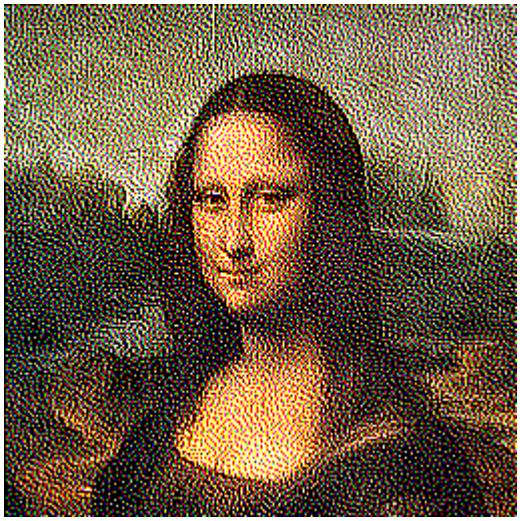

Figure 10: Sierra - Ordem 1

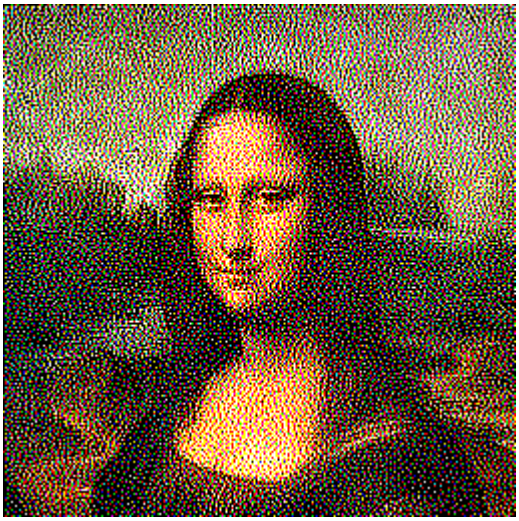

Figure 11: Sierra - Ordem 2

Stucki

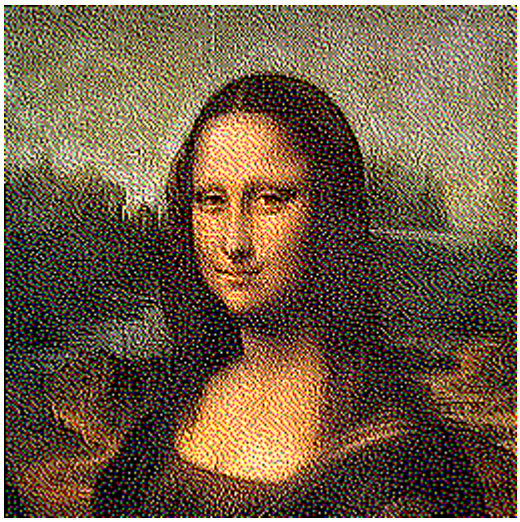

Figure 12: Stucki - Ordem 1

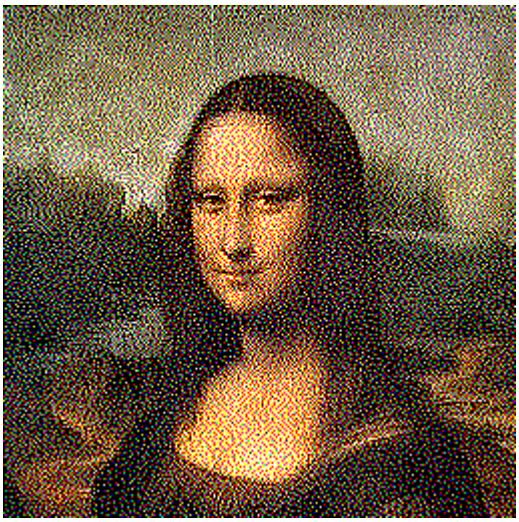

Figure 13: Stucki - Ordem 2

Jarvis, Judice e Ninke

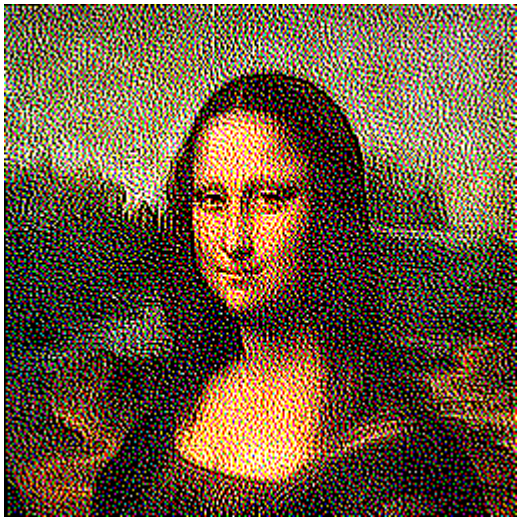

Figure 14: Jarvis, Judice e Ninke - Ordem 1

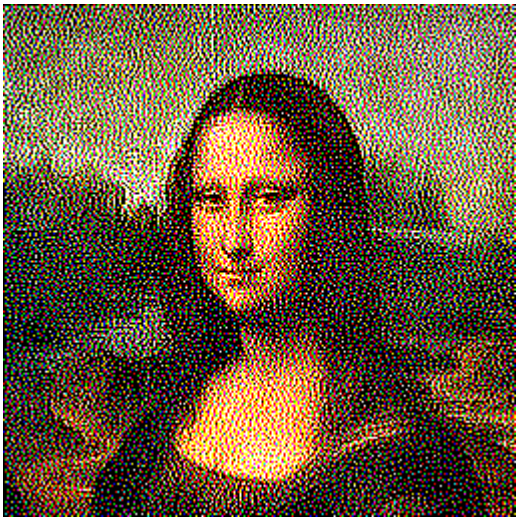

Figure 15: Jarvis, Judice e Ninke - Ordem 2

A partir de Burkes, todas as abordagens de propagação de erros apresentam resultados semelhantes. Todas resolvem bem o problema de mosaico e o excesso de propagação de erros. Também conseguem ter resultados equivalentes entre as diferentes ordem de varredura de pixels para esta imagem.

4. Conclusões

As abordagens iniciais serviram como uma boa forma de evidenciar falhas no processo de propagação de erros, mostrando que a escolha correta da maneira como se percorre a imagem e a própria forma de distribuição dos erros fazem diferença na imagem de saída. Entretando, nos testes realizados, as outras abordagens se mostraram eficientes no que se propõem, não sendo de fácil visualização diferenças entre elas tanto na matriz de propagação utilizada quanto na ordem de varredura.

Apesar disso, as matrizes que se mostraram eficientes nesta técnica operam de maneiras diferentes. Isso pode ser evidênciado no tempo de execução de cada uma, que depende da quantidade de calculos realizados a cada interação. Dentre elas, Burkes se destaca como a abordagem mais eficiente para a resolução deste problema, de acordo com teste realizado em que se aquisitou o tempo necessário para processar (nas duas diferentes ordens) a imagem *watch.png* para cada matriz de propagação de erros.

Floyd e Steinberg	1 minuto e 24,74 segundos
Stevenson e Arce	3 minutos e 9,18 segundos
Burckes	2 minutos e 3,93 segundos
Sierra	2 minutos e 53,21 segundos
Stuckis	3 minutos e 25,99 segundos
Jarvis, Judice e Ninke	3 minnutos e 12,34 segundos

Table g: Tempo de execução de cada abordagem

As abordagens aborgens que resolveram os problemas encontrados mas tiverem um tempo maior de processamento devem resolver também outros problemas não encontrados durante os testes realizados para este relatório. Portanto, tendo em mente o custo benefício, a abordagem **Burckes** une as melhores características para realizar a redução da quantidade de cores pelas técnicas de pontilhado por difusão de erro, unindo um *output* de qualidade com um tempop de execução menor.