Instituto Nacional de Matemática Pura e Aplicada

Retoque digital com o método SPH

Fernando Alves Mazzini

Dissertação de mestrado apresentada ao Instituto Nacional de Matemática Pura e Aplicada como requisito para a obtenção do título de Mestre em Matemática (Opção Matemática Computacional e Modelagem).

Orientador: Prof. Dr. Luiz H. de Figueiredo
Coorientador: Prof. Dr. Fabiano P. do Carmo

Rio de Janeiro - RJ
16 de fevereiro de 2017

Para Rover, Nina e Branca.

Sempre ao meu lado.

Agradecimentos

Quero agradecer ao amigo e coorientador Fabiano Petronetto do Carmo por sua valorosa contribuição e por todas as vezes que fez o papel de irmão mais velho, mesmo sendo o mais novo. Muito deste trabalho se deve à sua orientação e seu apoio. Agradeço também ao amigo Afonso Paiva Neto pelas preciosas sugestões. Meu muito obrigado ao meu orientador Luiz Henrique de Figueiredo e ao professor Paulo Cezar Pinto Carvalho, por toda boa vontade e consideração de ambos. Aos grandes amigos, tanto de dentro quanto de fora do ambiente acadêmico e profissional, Arnaldo Paterline Togneri, Klaus Fabian Côco, Fábio de Almeida Có e Márcio de Almeida Có, que inúmeras vezes me ajudaram ao longo dos últimos dez anos. Ao amigo Allan Lorenzoni Canal, pelo salvamento em tempo real. Aos amigos Douglas Araújo Victor, Alessandra Carvalho Gonçalves Victor e Diego Henrique Carvalho dos Santos, por tornarem mais aprazível todo este período de tensão.

Ao grande amigo, sempre presente mesmo a distância, Luis Fernando Brands Barbosa, pelo imensurável apoio ao longo de todos esses anos de amizade.

Reservo um agradecimento especial à minha namorada Bárbara Chisté Teixeira, por todo o carinho dado a mim, e por toda compreensão enquanto usei um tempo que era nosso para concluir este trabalho. E, claro, a toda minha família, por tudo que fizeram por mim, em particular à minha mãe Edlourdes Alves Penna e à minha avó Benedicta Benício Alves, pelo afeto de uma vida inteira.

Resumo

O retoque digital (*digital inpainting*) consiste na recuperação de partes degradadas da imagem com o objetivo de recuperar o aspecto original da mesma. Neste trabalho propomos a aplicação do método SPH (*Smooth Particle Hydrodynamics*) no contexto do retoque digital, explorando sua simples formulação em um algoritmo que usa a abordagem por agrupamento (*gather approach*) na discretização de sua representação integral, o que lhe confere uma satisfatória eficiência computacional ao mesmo tempo em que a qualidade dos resultados obtidos se compara a de algoritmos já consolidados encontrados na literatura.

Abstract

Digital inpainting consists in reconstructing damaged parts of an image in order to restore the original aspect. In this paper, we propose the use of SPH (*Smoothed Particle Hydrodynamics*) method into the digital inpainting context, exploring its simple formulation in an algorithm that uses the gather approach in the discretization of its integral representation, process that grants a satisfactory computing efficiency meanwhile the result data is compared to consolidated algorithms found in the current textbooks.

Sumário

Capítulo 1

Retoque digital

1.1 Introdução

A reconstrução de porções danificadas de uma imagem é uma prática largamente utilizada na restauração de pinturas antigas. A técnica, conhecida como retoque (*inpainting*) ou reintegração pictórica, foi estendida das pinturas à fotografia e ao vídeo com o mesmo propósito: reverter a deterioração por meio da adição ou remoção de elementos de imagem, preenchendo ou modificando as áreas danificadas de tal maneira que um observador não familiarizado com a imagem seja incapaz de detectar as alterações.

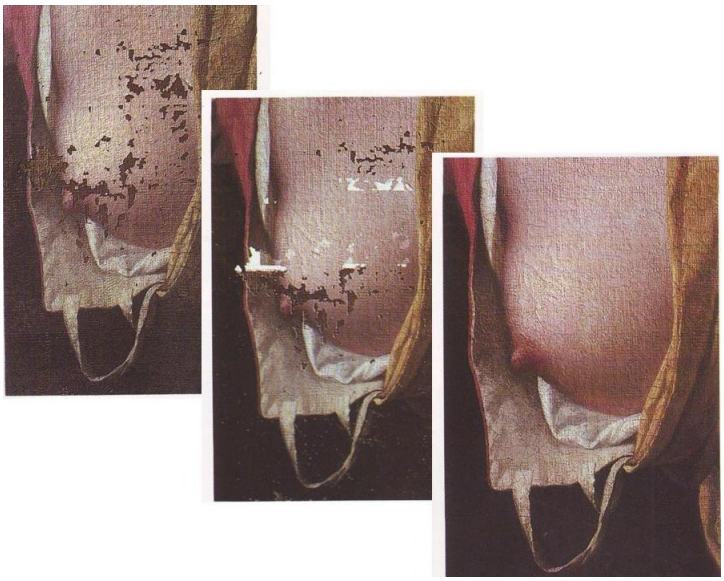


Figura 1.1: Evolução da reintegração pictórica (Técnica de reintegração mimética). Fonte: BERGEON, S. (1990). *Science et patience*. Paris: Editions des musées nationaux, p. 203. [Bai11].

O retoque manual é feito de forma subjetiva, que varia de profissional para profissional, abrange técnicas distintas [Bai11], e está ligado à arte inerente à imagem. As Figuras 1.1

e 1.2 ilustram dois trabalhos de restauração.

A Figura 1.1 ilustra uma imagem danificada (figura à esquerda), um estado intermediário do trabalho do restaurador, onde pode-se notar uma restauração parcial da imagem a partir de preenchimento de algumas das regiões danificadas (figura central), e o resultado final recuperando completamente o trabalho original (figura à direita). A figura 1.2 ilustra uma pintura antiga danificada pela ação do tempo (topo-esquerda) e sua restauração (base-esquerda). Podemos observar no *zoom* dado em uma parte restaurada da imagem (figura à direita) que a técnica utilizada no preenchimento é a mesma usada na obra original.

Em ambos os casos ilustrados acima, o objetivo principal do processo de restauração manual é obter uma imagem final de boa qualidade levando-se em conta a percepção visual humana, isto é, de forma a minimizar possíveis imperfeições e o surgimento de artefatos na imagem restaurada, de tal forma que um observador seja incapaz de detectar as alterações. De modo geral, podemos compilar os métodos adotados pelos restauradores através dos seguintes passos:

1. o aspecto global da imagem determina como preencher a área danificada;
2. a estrutura da vizinhança da área é transmitida para o seu interior, desenhando as linhas de contorno pelo prolongamento daquelas que incidem na sua borda;
3. as regiões distintas no interior da área são preenchidas com cor, a partir das informações da vizinhança;
4. os pequenos detalhes são adicionados.

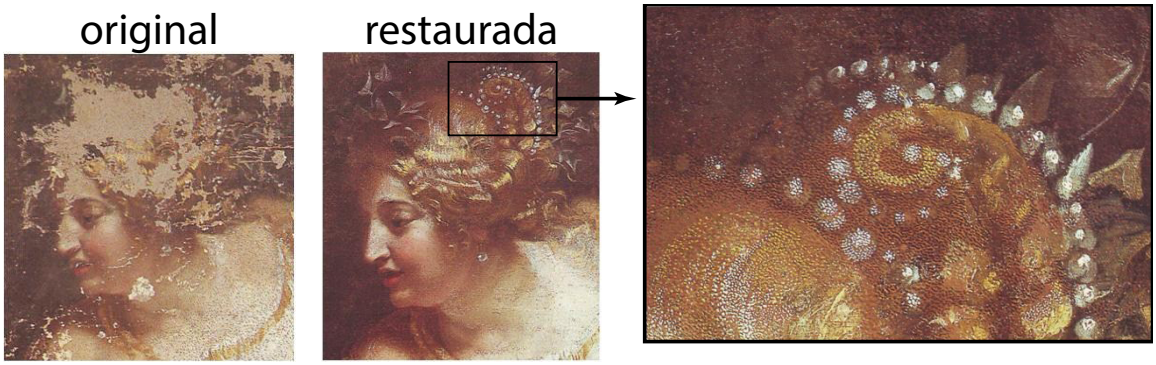


Figura 1.2: Antes e depois da reintegração pictórica em pintura antiga. À direita, detalhe da técnica utilizada na restauração (pontilhismo). Fonte: BERGEON, S. (1990). *Science et patience*. Paris: Editions des musées nationaux, p. 235. [Bai11].

O Retoque Digital, ou *digital inpainting*, busca traduzir os conceitos do retoque artístico tradicional em uma linguagem matemática e reproduzir seus resultados em imagens

digitais corrompidas. Foi proposto inicialmente por Bertalmio et al [BSCB00], onde os passos 2 e 3 do processo manual descrito acima são implementados numericamente a partir do transporte de informações de uma imagem em tons de cinza ao longo de linhas de mesmo nível de cinza na imagem. O passo 1 do processo manual, porém, não foi traduzido no algoritmo proposto por Bertalmio et al tornando o algoritmo de retoque digital mais eficiente devido a busca de informações para a recuperação da parte danificada apenas numa vizinhança desta região. Por último, o passo 4 do processo manual está diretamente relacionado à técnica de Síntese de Textura, e diversos trabalhos combinaram estas duas metodologias, retoque digital e síntese de textura, obtendo resultados onde um observador não consegue determinar as alterações em uma gama muito maior de imagens. Ressalta-se que o trabalho pioneiro de Bertalmio et al é uma das principais referências para todos os demais trabalhos na área.

A partir do trabalho seminal acima, a área de retoque digital tem sido amplamente investigada nos últimos anos. A partir do trabalho seminal de Bertalmio et al [BSCB00], diversos métodos de retoque digital foram desenvolvidos.

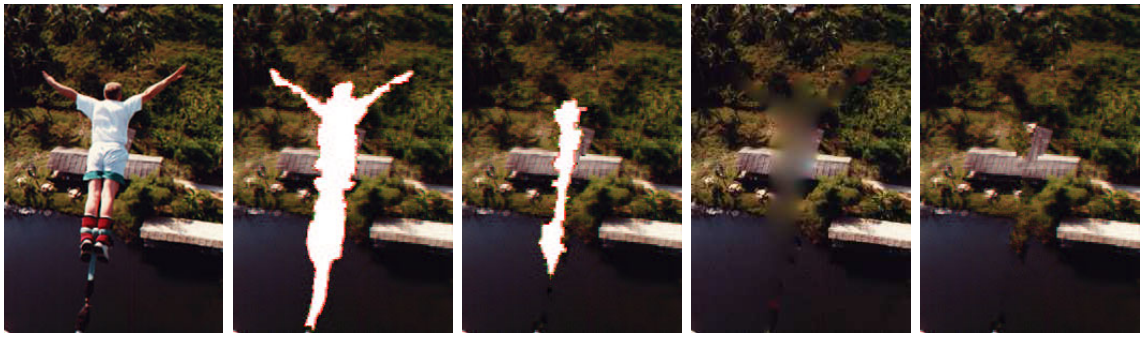
Inicialmente, o desenvolvimento de métodos de retoque digital tinha como objetivo recuperar a geometria dos objetos danificados numa imagem. Essa classe de métodos ficou conhecida como retoque digital estrutural[1] e é aplicada a situações mais simples, isto é, imagens onde a região de retoque necessita apenas da informação estrutural (ou geométrica) da imagem para que a recuperação obtida seja considerada satisfatória. Exemplos destas aplicações são imagens com regiões danificadas relativamente pequenas e imagens em preto e branco.

Posteriormente, a necessidade do aumento da capacidade do retoque digital para restaurar imagens mais complexas apoiou o desenvolvimento de métodos que acoplam a recuperação da geometria de objetos presentes na imagem a métodos de síntese de textura. Textura é compreendida como o aspecto de um objeto que pode diferenciá-lo e, portanto, distingui-lo de outro. A restauração da textura na região de retoque digital é uma meta muito mais difícil de ser obtida do que a restauração da geometria dos objetos (possivelmente) presentes nas regiões danificados. Esta classe de métodos ficou conhecida como retoque digital textural (Figura 1.3).

A remoção de ruído (ou *denoising*) é outro tópico relacionado à área de restauração de imagens. Ruído é caracterizado como uma região (em geral, um *pixel*) de alta oscilação local numa imagem. Esta caracterização é equivalente à caracterização de textura numa imagem. Portanto, métodos de remoção de ruído que dissociam ruído de textura são fundamentais para o ramo que investiga imagens com textura.

Enquanto no retoque tradicional a intervenção do artista restaurador é máxima, nos

[1]Essa classe também é conhecida como retoque digital geométrico.



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

Figura 1.3: Diferença entre o retoque estrutural e textural quando aplicados a grandes regiões: (a) Imagem original; (b, c) etapas da remoção, (d, e) resultado final via retoque estrutural e textural, respectivamente. Fonte: Criminisi et al [CPT03].

métodos de retoque digital o usuário se limita a definir apenas quais são as regiões a serem corrigidas. Em qualquer um dos métodos de retoque digital, a entrada consiste somente na imagem corrompida e uma máscara - uma imagem auxiliar que aponta as porções que devem ser restauradas[2].

Além de restaurações de imagens danificadas, algoritmos de retoque digital podem ser aplicados em outros problemas práticos, tais como, restauração de filmes (remoção de arranhões e manchas), remoção de legendas ou vandalizações (pichações), remoção de objetos indesejados, *zoom* digital e super-resolução.

1.2 Organização da dissertação

Organizamos o conteúdo deste trabalho da seguinte forma:

Para o leitor interessado, na seção 1.3 fazemos referência a trabalhos relevantes em frentes de pesquisa relacionadas ao contexto do retoque digital.

No capítulo 2 apresentamos os fundamentos teóricos dos algoritmos que foram escolhidos para fins de comparação com o nosso modelo, que será apresentado no capítulo 3. Iniciamos com o algoritmo precursor do retoque digital, proposto por Bertalmio et al [BSCB00] (e que inspirou todos os demais métodos), baseado no transporte suave de informações de cor para o domínio de retoque através da projeção do gradiente da suavidade da intensidade de imagem na direção das linhas de mesmo tom de cinza.

Em seguida mostramos o método desenvolvido por Oliveira et al [OBMC01], que unicamente filtra a imagem corrompida usando um núcleo de convolução gaussiano, reduzindo significativamente o tempo de execução, mas ainda mantendo a qualidade do retoque.

[2]Algoritmos de retoque digital podem possuir alguns parâmetros que também precisam ser definidos pelo usuário.

O terceiro algoritmo, formulado por Bertalmio et al [BBS01], trata a imagem como um fluido newtoniano bidimensional, fazendo uma relação entre o problema de retoque e o transporte de vorticidade em um fluido incompressível.

Por fim, apresentamos o algoritmo SPIR, proposto por DiBlasi et al [DBFTT11], que faz uso do método numérico SPH (*Smoothed Particle Hydrodynamics*), no tratamento de imagens com ruído. O SPH foi desenvolvido inicialmente para tratar de problemas astrofísicos, e é amplamente aplicado em problemas de dinâmica dos fluidos. No algoritmo SPIR, a execução da etapa de discretização do método SPH é feita segundo a abordagem por dispersão, possibilitando sua utilização no campo da remoção de ruídos.

No capítulo 3 destacamos um novo algoritmo, de nossa autoria, que aplica o método SPH ao contexto do retoque digital. Veremos que a abordagem por dispersão adotada no algoritmo SPIR se mostra inadequada para a recuperação de um grande número de *pixels* adjacentes, evento frequente nos problemas relacionados com o retoque digital. No nosso algoritmo, a etapa de discretização se dá através da abordagem por agrupamento. Esta escolha conduz a resultados semelhantes aos encontrados na literatura, além de trazer eficiência computacional ao método.

No capítulo 4 mostramos, de forma comparativa, os resultados numéricos obtidos com os algoritmos apresentados nos capítulos 2 e 3 em algumas aplicações usuais do retoque digital, avaliando também as limitações dos métodos e sua eficiência computacional.

Concluímos no capítulo 5 com nossas observações acerca dos resultados apresentados no capítulo 4 e algumas considerações pertinentes para estudos futuros.

1.3 Trabalhos relacionados

A seguir, propomos apresentar esta seção de trabalhos relacionados agrupando as principais referências relacionadas ao tema desta pesquisa em 3 grandes grupos.

O primeiro grupo de métodos é constituído por métodos de retoque digital estrutural. Inicialmente, alguns métodos de desoclusão, método para tornar visíveis objetos que estão obstruídos na imagem original, serão relacionados aos métodos de retoque digital pela forte influência destes métodos nos primeiros trabalhos sobre retoque digital. Em particular, veremos que os principais métodos desta classe podem ser modelados a partir de equações diferenciais parciais (EDP).

O segundo é o grupo dos métodos de retoque digital textural. Pode-se afirmar que este grupo contém métodos de retoque digital mais sofisticados buscando recuperar não somente a estrutura da região de retoque mas também a textura desta região.

Por último, o terceiro grupo descreve alguns métodos de *denoising* que de alguma maneira citaram o problema de retoque digital dentro de suas aplicações. Dentro desse grupo,

encontra-se um trabalho de *denoising* baseado no mesmo método no qual desenvolvemos o algoritmo de retoque digital proposto neste trabalho.

Retoque digital estrutural Podemos destacar alguns métodos de desoclusão como os precursores do retoque digital. Chamamos de desoclusão a recuperação de partes ocultas de objetos em uma imagem digital por interpolação a partir da proximidade da área obstruída. Das três áreas aqui mencionadas, é a significantemente menos estudada. O trabalho pioneiro nesta área se deve a Nitzberg et al [NMS93]. Os autores apresentaram uma técnica para a remoção de oclusões tendo como objetivo a segmentação de imagens, cuja ideia básica é conectar T-junções com o mesmo tom de cinza. A técnica foi desenvolvida principalmente para imagens simples, apresentando apenas alguns objetos com tons de cinza constantes.

Masnou e Morel [MM98] estenderam esta ideia, desenvolvendo uma técnica para imagens com topologia simples que realiza a desoclusão por meio da conexão das linhas de mesmo tom de cinza que incidem na fronteira da região a ser restaurada. Porém, o ângulo com o qual estas linhas de nível chegam à fronteira do domínio de retoque não é bem preservado.

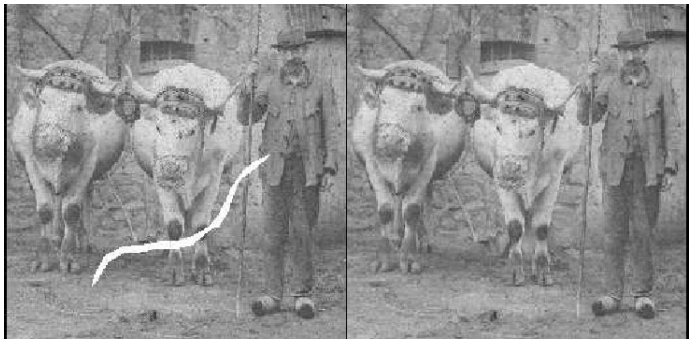


Figura 1.4: Desoclusão baseada em linhas de nível. Fonte: Masnou et al [MM98].

O algoritmo de Bertalmio et al [BSCB00] realiza o transporte do gradiente do laplaciano da imagem, tomado como uma medida da suavidade da mesma, na direção das linhas de mesmo tom de cinza. O sistema resultante é uma aproximação discreta da EDP

$$I_t = \nabla^{\perp} I \cdot \nabla \Delta I \tag{1.1}$$

para a intensidade de imagem I; onde $\nabla^{\perp}$ denota o gradiente ortogonal $(-\partial_y, \partial_x)$, e Δ denota o operador Laplaciano $\partial_x^2 + \partial_y^2$. O processo de retoque termina quando se alcança o estado estacionário, ou seja, quando não há mais informação a se propagar na direção

de $\nabla^{\perp} I$. Mais precisamente, quando temos

$$\nabla^{\perp} I \cdot \nabla \Delta I = 0. \tag{1.2}$$

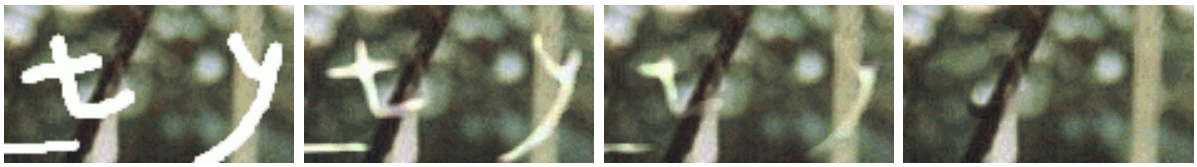


Figura 1.5: Etapas intermediárias do retoque digital. Fonte: Bertalmio et al [BSCB00].

A abordagem de Oliveira et al [OBMC01] reduz significativamente o tempo de execução simplesmente usando um filtro gaussiano G, e é baseada na ideia de que a sequência de imagens

$$I(i, j, t) = I(i, j, 0) * G(i, j, t) \tag{1.3}$$

pode ser vista como uma solução para a equação de difusão isotrópica

$$I_t = \Delta I, \tag{1.4}$$

que modela o problema.

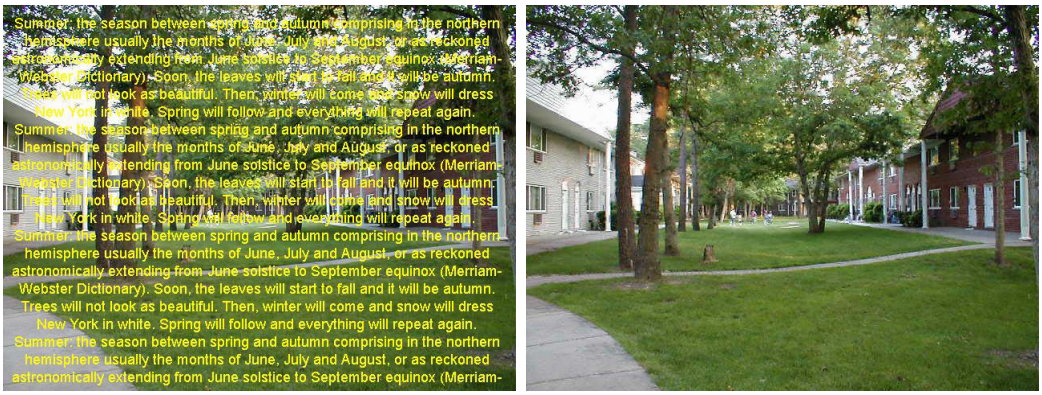


Figura 1.6: Remoção de texto sobre imagem com o algoritmo *Fast*. Fonte: Oliveira et al [OBMC01].

Bertalmio et al [BBS01] fizeram uma interessante analogia entre as equações do retoque digital e o transporte de vorticidade em fluidos incompressíveis, tratando a imagem como um fluido newtoniano bidimensional. O comportamento de um fluido newtoniano incompressível bidimensional com velocidade $V = (u, v)$, pressão p e viscosidade ν é regido pelas equações de Navier-Stokes

$$V_t + (V \cdot \nabla)V = -\nabla p + \nu \Delta V, \;\; \nabla \cdot V = 0; \tag{1.5}$$

onde p é a pressão e ν a viscosidade. Tomando o rotacional da primeira equação, obtemos a formulação corrente-vorticidade

$$\omega_t + (V \cdot \nabla)\omega = \nu \Delta \omega, \tag{1.6}$$

onde $\omega = \nabla \times V$ é a vorticidade.

Introduzindo uma função de corrente Ψ, tal que $\nabla^{\perp}\Psi = V$ (considerando que $\nabla\Psi = (v, -u)$), podemos relacioná-la com a vorticidade através do operador Laplaciano, $\Delta\Psi = \omega$. No caso de quase ausência de viscosidade ($\nu \approx 0$), temos o estado estacionário para a aproximação da solução de (1.6):

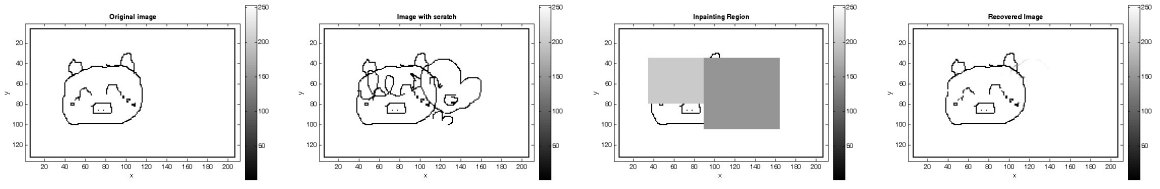
$$(V \cdot \nabla)\omega = \nabla^{\perp}\Psi \cdot \nabla \Delta \Psi \approx 0. \tag{1.7}$$

A semelhança entre as equações (1.2) e (1.7) induziu ao paralelo entre a função de corrente de um fluido incompressível bidimensional e o papel da função de intensidade da imagem no retoque digital, mostrado na tabela (1.1).

Tabela 1.1: Analogia entre a dinâmica de fluidos incompressíveis e o retoque digital.

Dinâmica dos fluidos	Processamento de imagens
função de corrente Ψ	intensidade de imagem I
velocidade do fluido $V = \nabla^{\perp}\Psi$	direção dos isófotos $\nabla^{\perp}I$
vorticidade $\omega = \Delta\Psi$	suavidade $w = \Delta I$
viscosidade ν	difusão anisotrópica ν

Au et al [AT01] desenvolveram um algoritmo de retoque baseado nas analogias descritas acima. Porém, o mesmo trabalha somente em domínios de retoque retangulares, e definir um único retângulo que contenha todo o domínio de retoque pode agregar informações de cor que estejam muito distantes da fronteira do domínio.



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

Figura 1.7: (a) Imagem original; (b) Imagem degradada; (c) determinação dos retângulos de retoque; (d) imagem restaurada pelo algoritmo NS. Fonte: Au et al [AT01].

Chan e Shen [SC02] desenvolveram um método de retoque digital que só pode ser aplicado em regiões relativamente pequenas, que empregam a difusão anisotrópica no interior

do domínio de retoque em imagens sem textura, baseados no método de *denoising* TV (*Total Variation*), de Rudin et al [RO94]. Além das aplicações em problemas de retoque digital, o algoritmo de retoque TV também se mostrou bem sucedido em problemas de *zoom* digital. Por outro lado, a conexão de linhas de contorno é feita por meio de linhas retas, prejudicando a qualidade do retoque. Além disso, o método também falha se o afastamento entre as linhas de contorno a serem conectadas for muito grande.

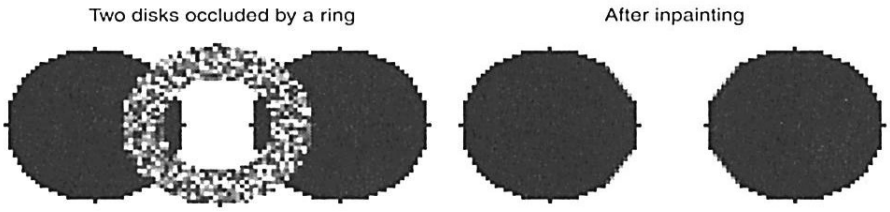


Figura 1.8: Retoque digital com o algoritmo TV. Fonte: Chan e Shen [SC02].

Chan e Shen [CS01c] desenvolveram outro método de retoque digital a partir de um novo modelo de difusão também baseado num método de restauração variacional [SC01]. O algoritmo CDD (*Curvature-Driven Diffusion*) leva em conta a curvatura das linhas de contorno (isófotos) e permite a restauração de domínios de retoque maiores. No entanto, a conexão de linhas de contorno por meio de linhas retas permanece.

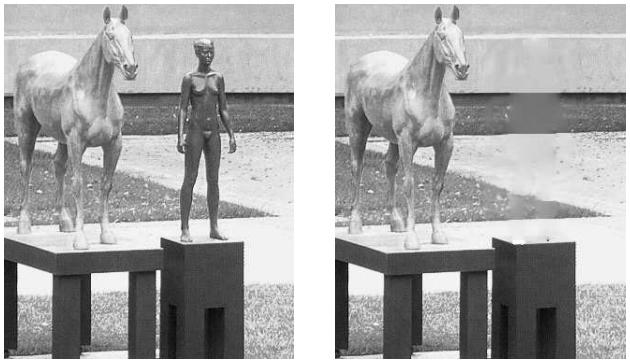


Figura 1.9: Retoque digital com o algoritmo CDD. Fonte: Chan e Shen [SC01].

Retoque digital e síntese de textura. Os algoritmos de retoque digital apresentados até aqui somente recuperam a informação estrutural/geométrica da região de retoque. Pode-se observar apenas a partir da imagem restaurada pelo algoritmo CDD (Figura 1.9) que a região de retoque é facilmente identificada devido a presença de um borrão na região de retoque e a ausência de textura na área restaurada. Este problema é inerente aos métodos de retoque digital estrutural conforme já citamos anteriormente e pode ser observado ao utilizar qualquer algoritmo de retoque digital deste grupo em uma região de retoque relativamente larga numa imagem com textura.

Algoritmos de retoque digital que além da informação estrutural também recuperam as informações de textura nas regiões de retoque foram desenvolvidos. Hirani et al [HT96] desenvolveram um método simples e eficaz para preencher regiões com uma textura selecionada combinando informações do domínio espacial com o domínio da frequência. É necessário que o usuário faça a seleção da textura, e caso a região a ser preenchida abranja várias texturas distintas, há a necessidade de segmentá-las e encontrar as substituições correspondentes por toda a imagem.

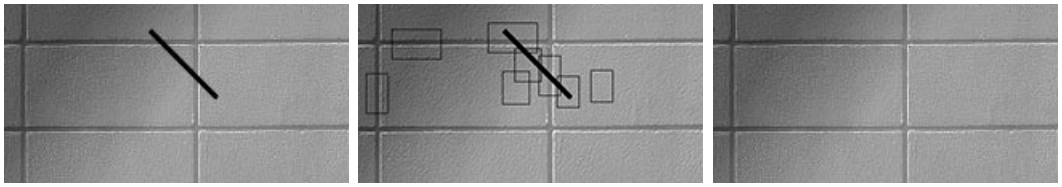


Figura 1.10: (a) Região a ser preenchida (preto); (a) seleção de textura; (c) resultado. Fonte: Hirani et al [HT96].

Bertalmio et al [BVSO03] desenvolveram um algoritmo a partir da decomposição da imagem na soma de duas imagens com diferentes características, a primeira representando a estrutura da imagem, enquanto a segunda determina somente a textura da imagem. Então, cada uma destas imagens é reconstruída separadamente. A primeira componente reconstruída pelo tradicional método de retoque digital do próprio autor [BSCB00], enquanto que o método de síntese de textura [EL99] é aplicado na segunda componente para recuperar detalhes da imagem original. Então, estas componentes são adicionadas na região de retoque definindo o resultado final.

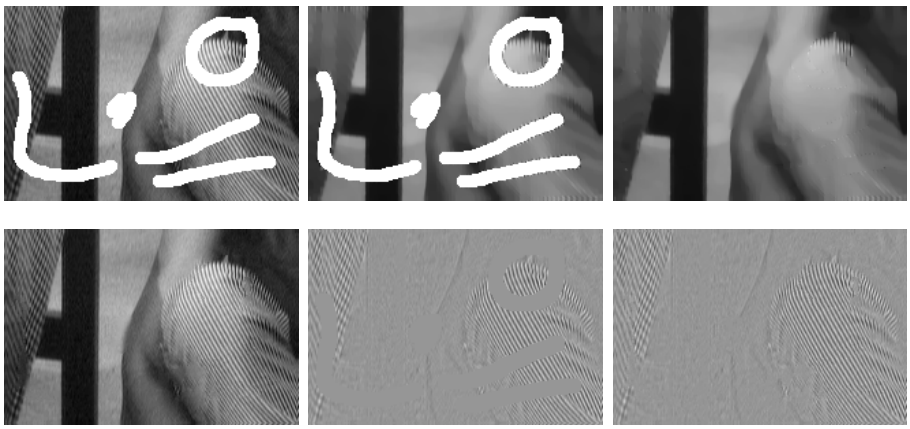
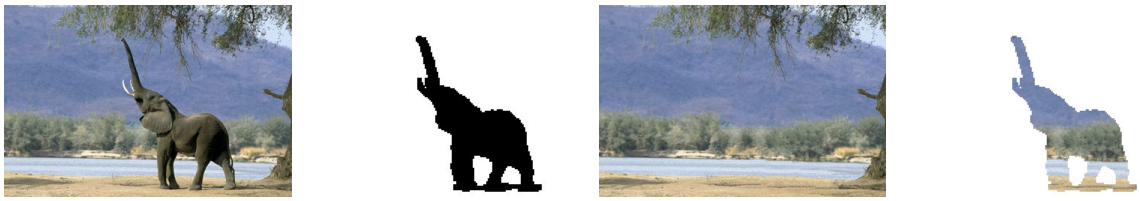


Figura 1.11: De cima para baixo, primeira coluna: imagem original e imagem retocada; segunda coluna: decomposição nas componentes estrutura e textura; terceira coluna: reconstrução via retoque digital e síntese de textura, respectivamente. Fonte: Bertalmio et al [BVSO03].

Drori et al. [DCOY03] propõem um método iterativo de retoque digital textural. A

metodologia aplicada intercala, para cada pixel da região de retoque, uma reconstrução suave que determina uma aproximação da informação estrutural com uma síntese de textura, onde um detalhe (textura) é adicionado a este pixel a partir da textura da imagem original escolhida a partir de um critério chamado de "confiança".



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

Figura 1.12: (a) Imagem original; (b) máscara que define o domínio de retoque; (c) resultado final; (d) conteúdo da região retocada. Fonte: Drori et al. [DCOY03].

Os dois trabalhos citados anteriormente propõem restaurar as informações geométricas e de textura separadamente. Criminisi et al [CPT04] propõem um algoritmo eficiente no qual as informações geométrica e de textura são propagadas simultaneamente. Fixado um pixel p da borda da região de retoque, o algoritmo busca uma janela J da imagem (9×9 por padrão) que não contém pontos da região de retoque [EL99] mais similar â janela de mesmo tamanho centrada em p, denotada por J_p. A partir daí, o valor de cada pixel da janela J_p que pertence a região de retoque é substituído pelo valor do pixel correspondente na janela J. Observe que, em um único passo, uma grande quantidade de pixels na região de retoque podem ser restaurados e, portanto, a região de retoque é atualizada.

(a) (b) (c)

Figura 1.13: (a) Imagem original; (b) determinação (manual) do domínio de retoque; (c) resultado final. Fonte: Criminisi et al [CPT04].

Remoção de ruídos. De modo geral, os algoritmos de *denoising* tendem a suavizar a textura da imagem. Casaca et al [CB10] propuseram a eliminação de ruídos em imagens reais dotadas de texturas por meio de um método numérico fundamentado em uma EDP não linear para extrair de uma imagem as componentes caracterizados por estrutura (*cartoon*), textura e ruído.

Figura 1.14: (a) Imagem original; (b) imagem restaurada. Decomposição nas três componentes: (c) *cartoon*; (d) textura e (e) ruído. Fonte: Casaca et al [CB10].

Zuo et al [ZZSZ13] desenvolveram um método de *denoising* com realce de textura através da aplicação de histogramas de gradientes orientados. Para imagens com muitas texturas distintas, o método pode gerar texturas incoerentes, havendo a necessidade de se usar algum método de segmentação da imagem, dividindo a mesma em regiões homogêneas e então aplicar o método de *denoising* a cada uma das regiões.

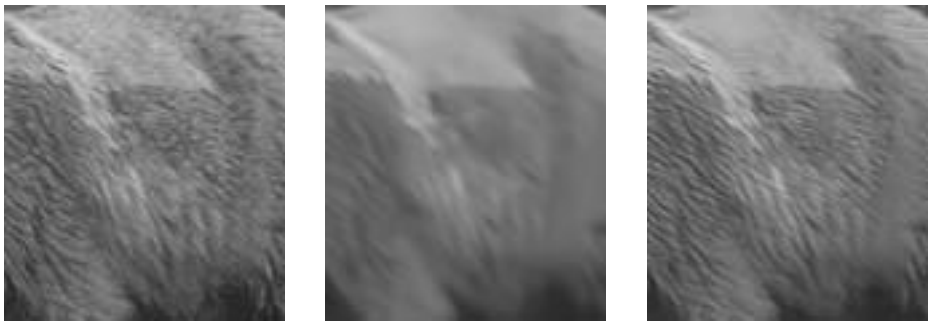


Figura 1.15: (a) Imagem original; (b) *denoising* convencional; (c) *denoising* com realce de textura. Fonte: Zuo et al [ZZSZ13].

Di Blasi et al [DBFTT11] introduziram uma nova técnica para a reconstrução de imagens com ruído usando o método SPH. Seu algoritmo, o SPIR (*Smoothed Particle Image Reconstruction*), usa a abordagem por dispersão na discretização da representação integral do SPH. Embora seja eficaz na remoção de ruído, a abordagem adotada não é adequada para o problema do retoque digital.

Figura 1.16: Remoção de ruído com o algoritmo SPIR. Fonte: Di Blasi et al [DBFTT11].

1.4 Contribuições

Nesta dissertação, apresentamos um novo método de retorque digital. Mais precisamente, desenvolvemos um método de retoque digital baseado no método *Smoothed Particle Hydrodynamics* (SPH) a partir da estratégia conhecida como abordagem por agrupamento (*gather approach*) para a discretização de sua representação integral. O método desenvolvido é chamado de G-SPIR, acrônimo de *Gather Smoothed Particle Image Reconstruction*.

Também implementamos no nosso método a estratégia de barreira de difusão. A barreira de difusão impede que informações de cor cruzem as linhas de contorno, auxiliando na preservação de bordas, e obtendo resultados mais rápidos, sem afetar a qualidade das restaurações obtidas.

O algoritmo desenvolvido é extremamente simples, só possui um parâmetro, e é computacionalmente eficiente. Aplicamos o algoritmo a diferentes situações de retoque digital e a qualidade dos resultados obtidos se compara a de outros algoritmos já consolidados presentes na literatura. Vale ressaltar que o algoritmo proposto é um método de retoque digital estrutural.

Capítulo 2

Algoritmos de restauração em imagens digitais

Neste capítulo descrevemos os algoritmos já publicados na literatura relacionada à restauração de imagens que serviram de referência para a formulação do algoritmo G-SPIR, nossa contribuição no contexto do retoque digital a ser apresentada no capítulo 3. Três deles são algoritmos de retoque digital: o BSCB, algoritmo precursor de Bertalmio et al [BSCB00]; o algoritmo de retoque rápido (*Fast*), de Oliveira et al [OBMC01]; e o algoritmo de retoque com as equações de Navier-Stokes, de Bertalmio et al [BBS01]. O quarto algoritmo, o SPIR, de Di Blasi et al [DBFTT11], usa o método SPH no problema de remoção de ruído.

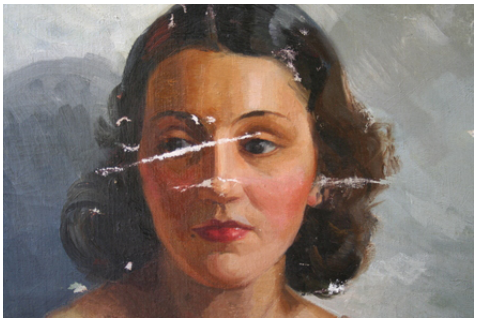
Em cada um dos algoritmos a entrada consiste, basicamente, em duas imagens de mesmo tamanho[1]. A primeira é a imagem a ser retocada. A segunda, denominada máscara, é uma imagem binária que destaca a região correspondente ao domínio de retoque da primeira, utilizada pelos algoritmos para localizar os *pixels* da imagem corrompida que sofrerão a ação do retoque (Figura 2.1). Ao final da seção de cada algoritmo será apresentado o respectivo resultado do retoque efetuado na figura 2.1(a).

2.1 Algoritmo BSCB

2.1.1 Fundamentos

Seja Ω a região da imagem a ser retocada e $\partial\Omega$ sua fronteira. O algoritmo de retoque digital (frequentemente mencionado na literatura como algoritmo BSCB), proposto por Bertalmio et al [BSCB00], consiste em, *grosso modo*, prolongar progressivamente para

[1] Além das imagens, cada algoritmo também recebe alguns parâmetros inerentes à sua formulação.



(a)

(b)

Figura 2.1: Imagens de entrada dos algoritmos de retoque digital: (a) Imagem corrompida (as regiões deterioradas correspondem ao domínio de retoque); (b) Máscara (as áreas em preto correspondem ao domínio de retoque da imagem original).

Ω as linhas de nível (ou isófotos[2]) que incidem em $\partial\Omega$ de tal maneira que o ângulo de interseção seja preservado, impedindo assim que tais linhas se cruzem. Simultaneamente, as regiões definidas pelo prolongamento destas linhas dentro de Ω são preenchidas com cor de forma a corresponder com as informações de sua vizinhança. Assim, encolhemos Ω iterativamente, transportando de forma suave as informações de sua fronteira para seu interior.

2.1.2 O algoritmo

Seja $I_0(i,j) : [0,M]\times[0,N] \to \mathbb{R}$, onde $[0,M],[0,N] \subset \mathbb{N}$, uma imagem digital em tons de cinza. O algoritmo BSCB gera uma sequência de imagens $I(i,j,n) : [0,M]\times[0,N]\times\mathbb{N} \to \mathbb{R}$ tal que $I(i,j,0) = I_0(i,j)$ e $\lim I(i,j,n) = I_R(i,j)$, onde $I_R(i,j)$ é a imagem restaurada. A partir da discretização em diferenças finitas avançadas da variação de I

$$I_t = \frac{I^{n+1} - I^n}{\Delta t},$$

o algoritmo pode ser escrito na forma

$$I^{n+1}(i,j) = I^n(i,j) + \Delta t I_t^n(i,j),\ \forall (i,j) \in \Omega. \tag{2.1}$$

O índice sobrescrito n denota a iteração, (i,j) as coordenadas do *pixel*, Δt a taxa de aprimoramento e $I_t^n(i,j)$ a atualização da imagem $I^n(i,j)$. Assim, $I^{n+1}(i,j)$ é uma versão aprimorada de $I^n(i,j)$ após nesta ser acrescida a atualização $I_t^n(i,j)$ ajustada por um fator Δt. Observe-se também que a equação (2.1) só age sobre os *pixels* de Ω.

Quanto à atualização $I_t^n(i,j)$, queremos que a informação contida na mesma possa

[2]Em uma imagem em *grayscale*, corresponde a uma fronteira definida por *pixels* com o mesmo tom de cinza.

prolongar para dentro de Ω as linhas de nível que incidem em $\partial\Omega$. Para isto utilizamos o Laplaciano $L^n(i,j) := I^n_{xx}(i,j) + I^n_{yy}(i,j)$, um estimador de suavidade da imagem, e também capaz de detectar bordas na mesma. Assim, se $\overrightarrow{N^h}(i,j)$ é a direção na qual se deseja propagar tal informação, então $I^n_t(i,j)$ é a projeção de $\nabla L^n(i,j)$ sobre $\overrightarrow{N^h}(i,j)$:

$$I^n_t(i,j) = \nabla L^n(i,j) \cdot \overrightarrow{N^h}(i,j), \tag{2.2}$$

onde o gradiente $\nabla L^n(i,j)$ mede a variação de $L^n(i,j)$. Em outras palavras, $I^n_t(i,j)$ é a variação do Laplaciano da imagem na direção de propagação. Quando a sequência de imagens se estabiliza, ou seja, quando $I^{n+1}(i,j) = I^n(i,j)$, concluímos por (2.1) e (2.2) que $\nabla L^n(i,j) \cdot \overrightarrow{N^h}(i,j) = 0$, significando que não há mais variação de $L^n(i,j)$ na direção de propagação e, assim, que toda a informação de $\partial\Omega$ foi passada para Ω.

Quanto à direção de propagação $\overrightarrow{N^h}(i,j)$, pareceria natural optar por esta ser ortogonal a $\partial\Omega$ em (i,j) (Figura 2.2), acreditando-se que tal escolha conduziria a uma bem sucedida continuação das linhas de nível. Porém a experiência mostra que estas linhas tendem a se alinhar com $\overrightarrow{N}$, produzindo resultados insatisfatórios (Figura 2.3).

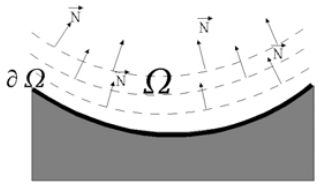
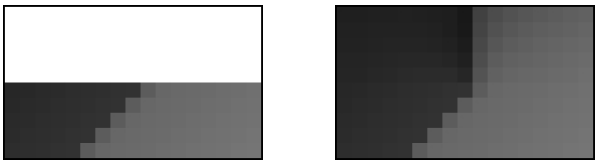


Figura 2.2: Direção de propagação normal a $\partial\Omega$.



(a) (b)

Figura 2.3: Escolha malsucedida da direção de propagação. (a) Detalhe da imagem original com a região Ω em branco. (b) Restauração de Ω com a informação propagada na direção normal a $\partial\Omega$. Fonte: Bertalmio et al [BSCB00].

Portanto, a melhor escolha para $\overrightarrow{N}$ é a própria direção das linhas de nível. A questão é que saber a direção das linhas de nível dentro de Ω equivale a ter a própria região restaurada. Usamos então uma estimativa variante no tempo do campo de direções destas linhas. Se $\nabla I^n(i,j)$ determina a direção de maior variação espacial, então esta variação é mínima na direção ortogonal $\nabla^\perp I^n(i,j)$, justamente a direção das linhas de nível, por

definição. Então tomamos $\overrightarrow{N^h}(i,j) = \nabla^{\perp} I^n(i,j)$. Embora grosseira no princípio, esta estimativa melhora progressivamente, alcançando a precisão desejada.

Em síntese, estimamos a variação da suavidade, dada pela discretização do Laplaciano, e projetamos esta variação na direção das linhas de nível. Esta projeção, por sua vez, atualiza a imagem no interior de Ω.

A difusão anisotrópica

Para assegurar a propagação correta das linhas de nível[3], evitando que as mesmas se cruzem, aplicamos um processo de difusão sobre Ω intercalado com o retoque digital. Para preservar os contornos e filtrar os ruídos simultaneamente, Perona-Malik [PM90] definiram o espaço de escala não-linear anisotrópica, modificando a equação da difusão térmica $I_t = div(c\nabla I) = c\Delta I$ através da substituição da constante c (coeficiente de difusão térmica) por uma função variante no tempo e no espaço:

$$I_t = div(c(x,y,t)\nabla I) = c(x,y,t)\Delta I + \nabla c \cdot \nabla I. \tag{2.3}$$

As duas parcelas no lado direito de (2.3) correspondem, respectivamente, à difusão isotrópica ($c(x,y,t)\Delta I$) e ao componente de ajuste difusivo ($\nabla c \cdot \nabla I$).

A intenção é promover a suavização nas áreas entre as linhas de nível, mas não sobre estas. Então o coeficiente de difusão deve ser uma função tal que seu valor seja 1 sobre as áreas e 0 sobre as linhas. Verificou-se que o gradiente $\nabla I(x,y,t)$ propicia uma boa estimativa da localização das linhas de nível. Assim, definiu-se $c(x,y,t)$ como uma função $c = g(\|\nabla I\|)$ da magnitude de ∇I, onde g é monótona decrescente, com $g(0) = 1$ (Figura 2.4). Desta forma, o processo ocorrerá principalmente no interior daquelas áreas e não afetará as linhas de nível, onde $\|\nabla I\|$ é grande.

Dentre as possíveis escolhas para g, optou-se pelas funções

$$g(\|\nabla I\|) = e^{-(\|\nabla I\|/\kappa)^2} \tag{2.4}$$

e

$$g(\|\nabla I\|) = \frac{1}{1 + (\|\nabla I\|/\kappa)^2}. \tag{2.5}$$

Os resultados obtidos com ambas no processo de difusão são similares. A constante κ pode ser ajustada manualmente, por simplicidade.

O componente de ajuste difusivo da equação (2.3) pode ser discretizado em uma

[3] A própria variação natural de L pode causar desequilíbrio no transporte de informação para Ω, principalmente se esta região for grande. Para mais detalhes, ver Chan-Shen [CS01a].

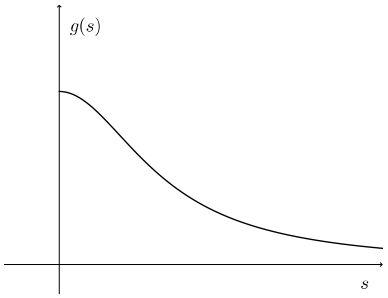


Figura 2.4: Aspecto qualitativo da não-linearidade da função g.

vizinhança 4-conectada[4] na forma

$$I^{n+1}(i,j) = I^n(i,j) + \lambda[c_N \cdot \nabla_N I + c_S \cdot \nabla_S I + c_E \cdot \nabla_E I + c_W \cdot \nabla_W I]^n_{(i,j)}, \tag{2.6}$$

onde $0 \leq \lambda \leq 1/4$ por uma questão de estabilidade, os índices subscrito (i,j) e sobrescrito n nos colchetes dizem respeito a todos os termos por eles envolvidos.

$$\begin{aligned}
\nabla_N I(i,j) &= I(i-1,j) - I(i,j) \\
\nabla_S I(i,j) &= I(i+1,j) - I(i,j) \\
\nabla_E I(i,j) &= I(i,j+1) - I(i,j) \\
\nabla_W I(i,j) &= I(i,j-1) - I(i,j)
\end{aligned} \tag{2.7}$$

são as diferenças de intensidade na vizinhança 4-conectada, e

$$\begin{aligned}
c_N^n(i,j) &= g(\|\nabla_N I^n(i,j)\|) \\
c_S^n(i,j) &= g(\|\nabla_S I^n(i,j)\|) \\
c_E^n(i,j) &= g(\|\nabla_E I^n(i,j)\|) \\
c_W^n(i,j) &= g(\|\nabla_W I^n(i,j)\|)
\end{aligned} \tag{2.8}$$

são os coeficientes de condutividade na mesma vizinhança.

Neste trabalho, usamos as equações (2.6), (2.7) e (2.8), bem como a forma (2.5) para a função g, para a implementação da difusão anisotrópica. Em [PM90] podemos encontrar aproximações numéricas alternativas, mas que produzem resultados equivalentes.

[4]Uma vizinhança 4-conectada do *pixel* (i,j) corresponde aos *pixels* de coordenadas $(i+1,j)$, $(i-1,j)$, $(i,j+1)$ e $(i,j-1)$ (Ver Figura 2.5).

2.1.3 Detalhes de implementação

As entradas do algoritmo são a imagem a ser restaurada e uma máscara que delimita as regiões que receberão o retoque. Toda a imagem original passa por um processo de suavização por difusão anisotrópica antes de entrar no processo de retoque, com o propósito de minimizar a presença de ruídos, que podem influenciar na estimativa das direções das linhas de nível que incidem em $\partial\Omega$. Em seguida tem início o processo de retoque, que só age sobre os *pixels* de Ω. Após um número prefixado de iterações do processo de retoque, é aplicada uma de difusão anisotrópica, e o processo se repete até que se atinja o estado de equilíbrio.

A discretização da equação de retoque digital é dada pela equação (2.1), onde

$$I_t^n(i,j) = \left(\nabla L^n(i,j) \cdot \frac{\overrightarrow{N}^n(i,j)}{\left\| \overrightarrow{N}^n(i,j) \right\|} \right) \left\| \nabla I^n(i,j) \right\|, \tag{2.9}$$

$$\nabla L^n(i,j) = (L^n(i+1,j) - L^n(i-1,j), L^n(i,j+1) - L^n(i,j-1)), \tag{2.10}$$

$$L^n(i,j) := I_{xx}^n(i,j) + I_{yy}^n(i,j), \tag{2.11}$$

$$\frac{\overrightarrow{N}^n(i,j)}{\left\| \overrightarrow{N}^n(i,j) \right\|} = \frac{(-I_y^n(i,j), I_x^n(i,j))}{\sqrt{(-I_y^n(i,j))^2 + (I_x^n(i,j))^2}}, \tag{2.12}$$

$$\beta^n(i,j) = \nabla L^n(i,j) \cdot \frac{\overrightarrow{N}^n(i,j)}{\left\| \overrightarrow{N}^n(i,j) \right\|}, \tag{2.13}$$

e

$$\left\| \nabla I^n(i,j) \right\| = \begin{cases} \sqrt{(I_{xbm}^n)^2 + (I_{xfM}^n)^2 + (I_{ybm}^n)^2 + (I_{yfM}^n)^2}; \text{ quando } \beta^n > 0, \\ \sqrt{(I_{xbM}^n)^2 + (I_{xfm}^n)^2 + (I_{ybM}^n)^2 + (I_{yfm}^n)^2}; \text{ quando } \beta^n < 0. \end{cases} \tag{2.14}$$

Os índices subscritos em (2.14), b e f, representam diferenças finitas atrasadas e adiantadas, respectivamente, enquanto que m e M denotam o mínimo e o máximo entre zero e a derivada parcial. As coordenadas (i,j) foram omitidas para simplificar a notação. Observe que $\|\nabla I\| = \left\|\nabla^{\perp} I\right\|$. Optou-se por estas discretizações das derivadas em lugar das diferenças centradas, bem como um campo de direções $\overrightarrow{N}$ não normalizado, por questões de estabilidade do algoritmo[5].

Seja (i,j) um *pixel* de Ω. Ao efetuarmos o transporte das informações para o interior de Ω, consideramos ponto da fronteira $\partial\Omega$ qualquer *pixel* da vizinhança 8-conectada (Figura 2.5) de (i,j) que pertença ao complementar de Ω.

Durante o processo de restauração, alternamos repetidamente A iterações de retoque

[5]Maiores detalhes podem ser vistos em [OS88], [MO00] e [ROF92].

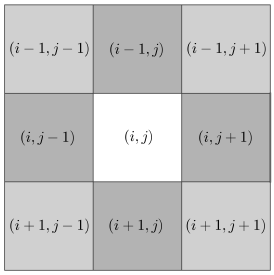
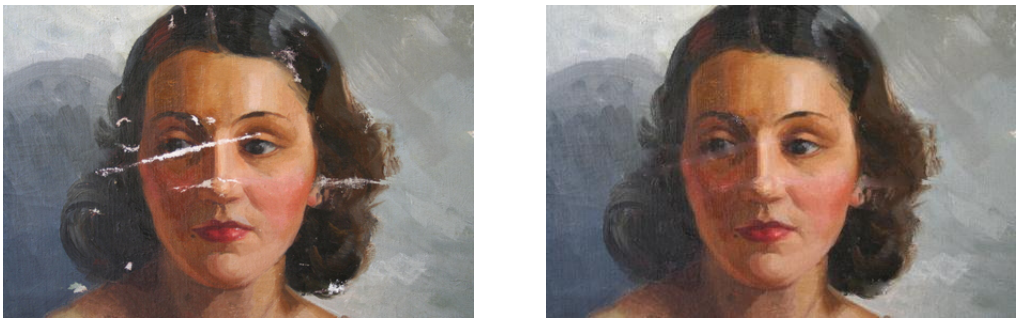


Figura 2.5: Vizinhança 8-conectada do *pixel* (i, j). Em cinza mais escuro, contidos na vizinhança 8-conectada, destacam-se os *pixels* da vizinhança 4-conectada de (i, j).

digital e B iterações de difusão anisotrópica até que um número máximo de iterações T seja atingido ou que as mudanças na imagem fiquem abaixo de um limiar preestabelecido. Neste trabalho utilizamos $A = 15$, $B = 2$ e $\Delta t = 0.1$. O valor de T depende do tamanho de Ω. Imagens coloridas foram separadas em três canais distintos, e cada um deles foi submetido ao processo de retoque digital.



(a) (b)

Figura 2.6: Restauração da imagem corrompida (Figura 2.1) obtida com o algoritmo BSCB.

2.2 Algoritmo de retoque rápido (*Fast*)

2.2.1 Fundamentos

Apesar dos métodos de restauração baseados em equações diferenciais parciais não lineares terem o potencial de preservar bordas, o retoque digital, de modo geral, não é um problema bem-posto. Implementações numéricas velozes são custosas de se obter tanto quanto é difícil encontrar modelos matemáticos adequados [CS01b]. O algoritmo BSCB (seção 2.1) produz resultados satisfatórios, porém o tempo despendido é grande. Isto motivou Oliveira et al [OBMC01] a desenvolver um algoritmo que reproduzisse resultados similares em um tempo menor.

De acordo com o teorema de amostragem [GV97], para que um sinal contínuo possa ser corretamente recuperado a partir de sua versão amostrada, a taxa de amostragem deve ser, pelo menos, duas vezes superior à maior componente de frequência presente no sinal. Então a recuperação do sinal se dá pela simples aplicação de um filtro passa-baixas [HVV01]. Assim, dada a natureza do problema de restauração (grandes áreas com toda informação perdida), só podemos esperar por uma reconstrução aproximada no lugar da reconstituição exata. Portanto, para que o procedimento de retoque seja bem sucedido, é necessário que a região a ser restaurada seja suficientemente pequena. Em regiões pequenas, podemos utilizar modelos mais simples e ainda obter resultados próximos aos dos modelos mais sofisticados. O algoritmo de retoque rápido (ou algoritmo *Fast*) também tira proveito do fato de o sistema visual humano tolerar uma certa quantidade de borramento em áreas não associadas às bordas de alto contraste [Kan79].

2.2.2 O algoritmo

Seja Ω uma pequena região da imagem a ser restaurada e $\partial\Omega$ sua fronteira. Se Ω é suficientemente pequena, o procedimento de restauração pode ser aproximado por um processo de difusão isotrópica, propagando a informação de $\partial\Omega$ para Ω. A versão mais simples do algoritmo consiste em iniciar Ω, removendo-se toda informação de cor e, em seguida, aplicar o retoque convolvendo-se repetidamente a região com um núcleo de difusão. A discretização de $\partial\Omega$ é feita como descrito na subseção 2.1.3. Como no algoritmo BSCB, o número de iterações pode ser ajustado manualmente, ou o processo pode ser interrompido assim que as mudanças na imagem fiquem abaixo de um limiar preestabelecido. À medida que o processo de difusão evolui, o retoque progride de $\partial\Omega$ para o interior de Ω.

De acordo com Koenderink [Koe84], dada a imagem original $I(i, j, 0)$, a sequência de imagens obtidas pela convolução com um núcleo gaussiano[6] $G(i, j, t)$:

$$I(i, j, t) = I(i, j, 0) * G(i, j, t) \tag{2.15}$$

pode ser vista como uma solução para a equação de difusão térmica (ou difusão isotrópica)

$$I_t = \Delta I. \tag{2.16}$$

Então convolver-se uma imagem com um núcleo gaussiano equivale a aplicar sobre a mesma uma difusão isotrópica. Na implementação do algoritmo *Fast*, utilizamos dois

[6] O núcleo gaussiano corresponde a um filtro passa-baixas cuja função é permitir a passagem de valores de baixa frequência, mas eliminar os de alta frequência, causando a suavização da imagem [Gon09].

núcleos de médias ponderadas que só levam em conta a contribuição da vizinhança 8-conectada, tomando peso zero para o *pixel* central (Figura 2.7, (b) e (c)).

```
Iniciar Ω;
for (iter =0; iter < num.iteracoes; iter=iter+1)
    efetuar convolucao do nucleo com Ω;
```

(a)

a	b	a
b	0	b
a	b	a

(b)

c	c	c
c	0	c
c	c	c

(c)

Figura 2.7: Núcleos de difusão com a = 0.073235, b = 0.176765 e c = 0.125.

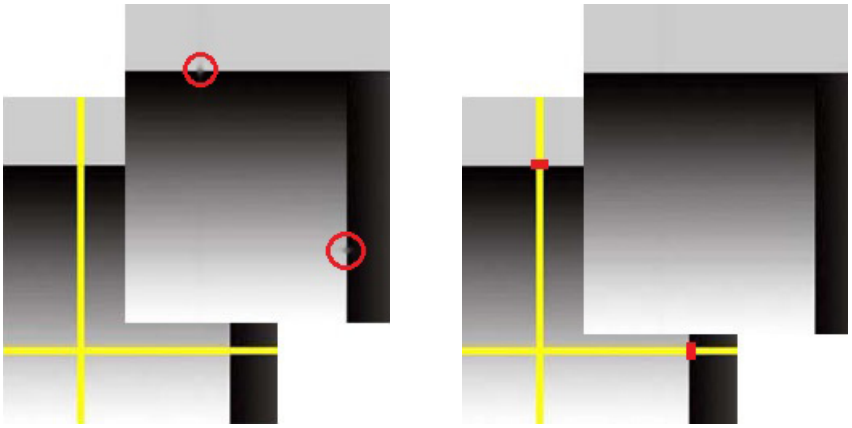
2.2.3 Detalhes de implementação

Barreiras de difusão

A versão elementar do algoritmo (Figura 2.7(a)) pode produzir um borramento indesejado onde Ω cruza bordas com alto contraste. Como o propósito do retoque rápido é gastar pouco tempo na etapa de restauração, a maior parte do tempo despendido em todo processo se dá na especificação da máscara de retoque, que é determinada pelo usuário. Então aproveitamos esta etapa para definir barreiras de difusão, que visam minimizar o efeito indesejável produzido nos cruzamentos com bordas.

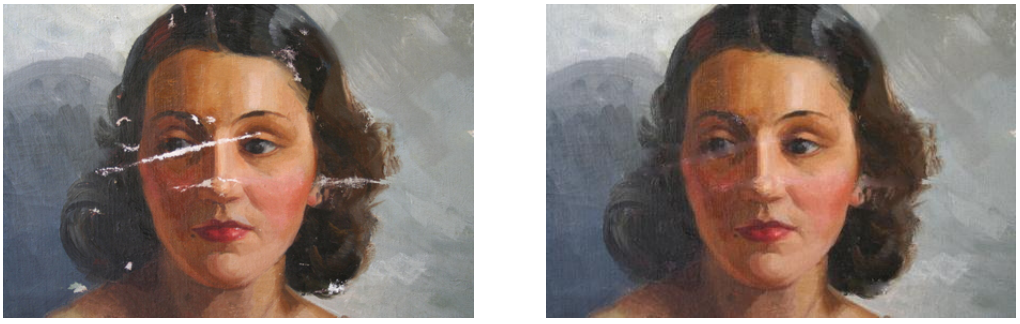
Uma barreira de difusão é, simplesmente, um segmento de dois *pixels* de largura inserido sobre uma interseção de $\partial\Omega$ com uma borda de alto contraste. Ao efetuarmos o retoque sobre estas barreiras, obtemos um resultado similar àquele obtido com a difusão anisotrópica, no que concerne à reconstrução das linhas de nível, mas sem o mesmo custo computacional. Quando um *pixel* na fronteira da barreira passa pelo processo de retoque, sua cor é definida, mas o processo de difusão é interrompido, evitando a mistura de informações presentes nos dois lados da borda e assegurando a integridade da mesma.

A figura 2.8 ilustra o procedimento. As linhas amarelas representam o domínio de retoque Ω. Em (a), vemos o borramento consequente da ação da difusão isotrópica (detalhes circulados em vermelho). Em (b), a utilização de barreiras (em vermelho) impede o avanço da difusão e preserva as bordas.



(a) (b)

Figura 2.8: Resultados distintos do algoritmo *Fast* sobre o mesmo domínio Ω (amarelo): (a) sem o uso de barreiras de difusão; (b) com o uso de barreiras de difusão. Fonte: Oliveira et al [OBMC01].



(a) (b)

Figura 2.9: Restauração da imagem corrompida (Figura 2.1) obtida com o algoritmo de retoque rápido (*Fast*).

2.3 Retoque digital com as equações de Navier-Stokes

2.3.1 Fundamentos

O algoritmo BSCB [BSCB00], descrito na seção 2.1, oferece a abordagem mais natural ao problema de retoque digital. Como visto naquela seção, o algoritmo procura imitar as técnicas da restauração manual, estendendo as linhas de nível que incidem na fronteira da região a ser restaurada para seu interior, conectando estas linhas de nível, e preenchendo com cor as regiões entre as linhas segundo as informações presentes na fronteira.

Se a variação de intensidade em uma imagem I é dada por ∇I, então as direções das linhas de nível são definidas por $\nabla^{\perp} I$, que determina a direção de menor variação de intensidade. Seja Ω a região a ser restaurada em I. A informação transportada da fronteira $\partial\Omega$ para o interior de Ω é a variação da medida de suavidade ΔI. O algoritmo determina a projeção do gradiente da suavidade na direção das linhas de nível, resultando numa discretização da equação

$$I_t = \nabla^{\perp} I \cdot \nabla \Delta I. \tag{2.17}$$

Com a adição da difusão anisotrópica, a equação acima toma a forma

$$I_t = \nabla^{\perp} I \cdot \nabla \Delta I + \nu \nabla \cdot (g(\|\nabla I\|) \nabla I). \tag{2.18}$$

O processo de retoque termina quando se alcança o estado estacionário, ou seja, quando não há mais informação a se propagar na direção de $\nabla^{\perp} I$. Mais precisamente, quando temos

$$\nabla^{\perp} I \cdot \nabla \Delta I = 0. \tag{2.19}$$

A relação entre o retoque digital e o transporte de vorticidade em fluidos incompressíveis

Seja $V = (u, v)$ a velocidade de um fluido newtoniano incompressível bidimensional. Seu comportamento é regido pelas equações de Navier-Stokes [BBS01]

$$V_t + (V \cdot \nabla) V = -\nabla p + \nu \Delta V, \;\; \nabla \cdot V = 0; \tag{2.20}$$

onde p é a pressão e ν a viscosidade. Derivando as componentes cartesianas da primeira equação em (2.20) em relação a y e a x respectivamente:

$$\frac{\partial}{\partial y} \left(u_t + (V \cdot \nabla) u = -p_x + \nu \Delta u \right), \tag{2.21}$$

$$\frac{\partial}{\partial x} \left(v_t + (V \cdot \nabla) v = -p_y + \nu \Delta v \right), \tag{2.22}$$

encontramos

$$u_{ty} + u_y u_x + u u_{xy} + v_y u_y + v u_{yy} = -p_{xy} + \nu(u_{xxy} + u_{yyy}), \tag{2.23}$$

$$v_{ty} + u_x v_x + u v_{xx} + v_x v_y + v v_{yx} = -p_{yx} + \nu(v_{xxx} + v_{yyx}). \tag{2.24}$$

Finalmente, subtraindo (2.23) de (2.24), rearranjando os termos, e levando-se em conta que $\nabla \cdot V = u_x + v_y = 0$, obtemos a formulação corrente-vorticidade de (2.20)

$$\omega_t + (V \cdot \nabla)\omega = \nu \Delta \omega, \tag{2.25}$$

onde $\omega = \nabla \times V$ é a vorticidade.

Introduzindo uma função de corrente Ψ, tal que $\nabla^\perp \Psi = V$ (considerando que $\nabla \Psi = (v, -u)$), podemos relacioná-la com a vorticidade através do operador Laplaciano, $\Delta \Psi = \omega$. No caso de quase ausência de viscosidade ($\nu \approx 0$), temos o estado estacionário para a aproximação da solução de (2.25):

$$(V \cdot \nabla)\omega = \nabla^\perp \Psi \cdot \nabla \Delta \Psi \approx 0. \tag{2.26}$$

A semelhança entre as equações (2.26) e (2.19) nos induz ao paralelo entre a função de corrente de um fluido incompressível bidimensional e o papel da função de intensidade da imagem no retoque digital, mostrado na tabela 2.1.

O propósito é utilizar um algoritmo baseado nas equações de Navier-Stokes como uma abordagem para o problema do retoque digital, e o método apresentado nesta seção busca resolver a equação do transporte de vorticidade

$$w_t + (V \cdot \nabla)w = \nu \nabla \cdot (g(\|\nabla w\|)\nabla w) \tag{2.27}$$

sobre o domínio de retoque, no lugar da equação (2.18) que, em termos de processamento de imagens, corresponde à equação (2.25). Aqui, $V = \nabla^\perp I$, e a presença da função g (ver subseção 2.1.2) leva em consideração a difusão anisotrópica da suavidade w.

Tabela 2.1: Analogia entre a dinâmica de fluidos incompressíveis e o retoque digital.

Dinâmica dos fluidos	Processamento de imagens
função de corrente Ψ	intensidade de imagem I
velocidade do fluido $V = \nabla^\perp \Psi$	direção dos isófotos $\nabla^\perp I$
vorticidade $\omega = \Delta \Psi$	suavidade $w = \Delta I$
viscosidade ν	difusão anisotrópica ν

A imagem I é então restaurada resolvendo-se o problema de Dirichlet

$$\Delta I = w,\ I|_{\partial\Omega} = I_0. \tag{2.28}$$

2.3.2 O algoritmo

Suponha, inicialmente, que o domínio de retoque Ω seja uma região retângular de dimensões $M \times N$. O algoritmo NS[7] usa as informações presentes em $\partial\Omega$ para resolver as equações (2.27) e (2.28) em Ω, simultânea e iterativamente, de tal forma que no "tempo" $t = n\Delta t$ tenhamos I^n e w^n. A iteração prossegue até que se atinja o estado estacionário.

Para aproximar as derivadas envolvidas no processo, foram utilizadas diferenças finitas centrais[8] variantes no tempo.

A iteração usada para evoluir a equação (2.27) no tempo pode ser escrita na forma

$$w_{i,j}^{n+1} = w_{i,j}^n + \Delta t[-u^n D_x^0 w_{i,j} - v^n D_y^0 w_{i,j} + \nu\{\text{discretização da difusão}\}], \tag{2.29}$$

onde D_x^0 e D_y^0 denotam as discretizações das derivadas parciais primeiras.

Para resolver a equação de $\Delta I^n = w^n$, usamos a discretização do Laplaciano

$$\Delta y^2(I_{i+1,j} + I_{i-1,j}) + \Delta x^2(I_{i,j+1} - I_{i,j-1}) - 2I_{i,j}(\Delta y^2 + \Delta x^2) = \Delta x^2 \Delta y^2 w_{i,j}^n \tag{2.30}$$

que, juntamente com a transformação de I^n e w^n em vetores de comprimento $M * N$, produz o sistema linear equivalente $A\vec{I^n} = \vec{w^n}$, onde $A_{M*N \times M*N}$ é a matriz da discretização do Laplaciano. Neste trabalho utilizamos o método de Gauss-Seidel para resolver o sistema linear.

Quanto à difusão anisotrópica, foi utilizada a forma

$$\nabla \cdot (g(\|\nabla w\|)\nabla w) = \partial_x(g(\|\nabla w\|)\nabla w_x) + \partial_y(g(\|\nabla w\|)\nabla w_y), \tag{2.31}$$

com g definida tal como na equação (2.5). A discretização de (2.31) segue as seguintes diretivas:

1. Aplicar diferenças centrais onde possível e a forma unilateral de três pontos na fronteira (ver subseção 2.3.3 a seguir) para o cálculo de w_x e w_y em cada ponto de I;

[7] A partir daqui tomaremos a liberdade de nos referirmos ao algoritmo baseado nas equações de Navier-Stokes pelas iniciais NS por uma simples questão de comodidade.

[8] Aproximações para as derivadas cujas fórmulas se baseiam na expansão do polinômio de Taylor para a função f: $\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x,y)-f(x-\Delta x,y)}{2\Delta x}$ e $\frac{\partial f}{\partial y} \approx \frac{f(x,y+\Delta y)-f(x,y-\Delta y)}{2\Delta y}$.

2. Calcular $\|\nabla w\| = \sqrt{w_x^2 + w_y^2}$ e determinar $g(\|\nabla w\|)$ em cada ponto de I;

3. Aplicar diferenças centrais onde possível e a forma unilateral de três pontos na fronteira para o cálculo de $\partial_x(g(\|\nabla w\|)\nabla w_x)$ e $\partial_y(g(\|\nabla w\|)\nabla w_y)$, e somar estas parcelas para obter o valor desejado em cada ponto de I.

2.3.3 Detalhes de implementação

Condições de contorno e o tratamento dos pontos de fronteira

Ao resolver simultaneamente as equações mencionadas na seção anterior, o algoritmo NS necessita de informações de fronteira tanto para I^n quanto para w^n. A única informação presente em $\partial\Omega$ é I, mas também levamos em conta que I é bem conhecida não só na fronteira, mas fora dela. Para determinar as condições de contorno do problema de Dirichlet (2.28), simplesmente usamos os valores de I em $\partial\Omega$. Já a determinação de w na fronteira é orientada pelas seguintes diretivas:

1. Calcular $u = -I_y$ e $v = I_x$ para uma espessura de três *pixels* a partir de $\partial\Omega$, usando diferenças centrais onde possível e a forma unilateral de três pontos, caso contrário;

2. Com os valores de u e v obtidos em (1), calcular u_y e v_x para uma espessura de um *pixel* a partir de $\partial\Omega$, usando diferenças centrais onde possível e a forma unilateral de três pontos, caso contrário;

3. Com os valores de u_y e v_x obtidos em (2), calcular $w = v_x - u_y$.

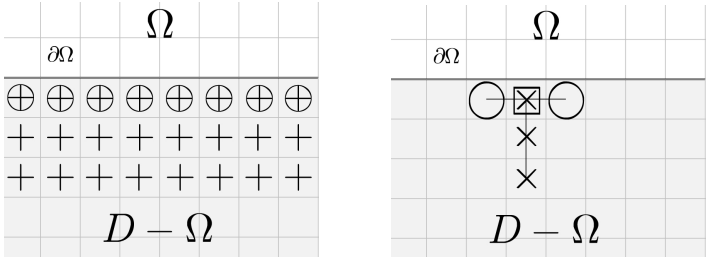
A forma forma unilateral de três pontos mencionada acima é dada por

$$f'_j = \frac{-f_{j+2} + 4f_{j+1} - 3f_j}{2\Delta x} + O(\Delta x^2),$$
$$f'_j = \frac{3f_j - 4f_{j-1} + f_{j-2}}{2\Delta x} + O(\Delta x^2).$$

Estas formas, juntamente com as diferenças centrais, garantem precisão de segunda ordem nas condições de contorno [AT01].

Domínios de retoque irregulares

Em quase a totalidade dos problemas de retoque, o domínio não é retangular. Uma vez que a formulação do algoritmo NS assume que Ω seja retangular (pois a formulação para domínios irregulares acarreta dificuldades de implementação [AT01]), mesmo que o retoque fosse realizado sobre o menor retângulo $\bar{\Omega}$ contendo Ω, a informação contida em $\bar{\Omega} - \Omega$ provavelmente seria perdida, comprometendo a qualidade do retoque. Para superar



(a) (b)

Figura 2.10: Ilustração do cálculo das condições de contorno na parte inferior fronteira. Em (a): os sinais '+' marcam onde u e v devem ser calculados numa espessura de três *pixels* e os sinais 'o' marcam onde u_y, v_x e w (espessura de um *pixel*) devem ser calculados. Em (b): No cálculo de u_y e v_x no ponto de fronteira, usa-se diferença central para v_x e uma forma unilateral de três pontos para u_y.

o problema, o que fazemos é resolver as equações (2.27) e (2.28) em $\bar{\Omega}$, mas atribuindo os resultados apenas a Ω, e preservando as informações em $\bar{\Omega} - \Omega$.

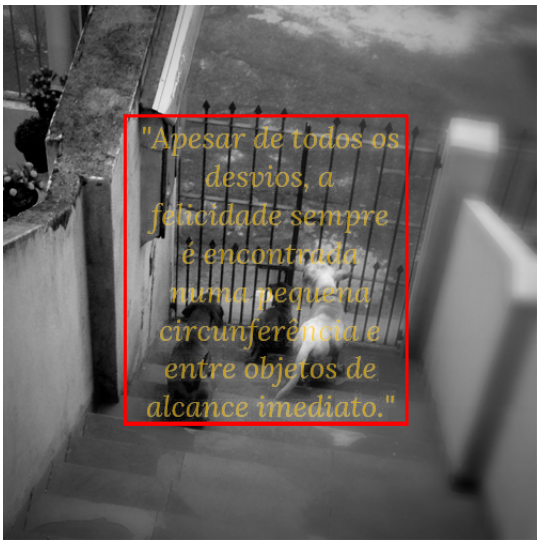


Figura 2.11: Domínio irregular: O conjunto de caracteres do texto sobreposto à fotografia corresponde ao domínio de retoque Ω, enquanto o retângulo vermelho delimita a regíão $\bar{\Omega}$.

2.4 Remoção de ruído com o algoritmo SPIR

2.4.1 Fundamentos

O método SPH

O SPH (*Smoothed Particle Hydrodynamics*) é um método numérico sem malha[9] de abordagem lagrangeana[10] desenvolvido nos anos 1970 para tratar e modelar problemas astrofísicos. Sua simples formulação e a facilidade de modelagem oferecida pelo mesmo é o que o torna eficiente e geral o bastante para ser aplicado em problemas de dinâmica de fluidos e modelagens de fenômenos naturais em geral. O método SPH determina uma aproximação para uma função e suas derivadas a partir de uma média local. Mais precisamente, a partir de um conjunto de partículas, pontos que representam o objeto da simulação e possuem propriedades inerentes ao problema, cada função e suas derivadas em um determinado ponto são aproximadas por uma média ponderada de contribuições dadas por partículas que estão próximas a este ponto. A formulação básica do método pode ser dividida em duas etapas: a representação integral e a aproximação por partículas.

A representação integral de uma função f definida num domínio $\Omega \subset \mathbb{R}^2$ é definida pela convolução da função f por uma função suave $W_h : \mathbb{R}^2 \to \mathbb{R}$, denominada núcleo. Usamos como núcleo uma função de base radial

$$W_h = W(R), \tag{2.32}$$

onde $R = \|\mathbf{x}\|/h$ com $W(r) = 0$ quando $r > \kappa$, com $\kappa \in \mathbb{R}$ associado ao núcleo W. Portanto, a representação integral é dada por

$$f^h(\mathbf{u}) = \int_\Omega f(\mathbf{x}) W_h(\mathbf{u} - \mathbf{x}) d\Omega. \tag{2.33}$$

No método SPH, o núcleo W_h é, em geral, escolhido sendo uma função diferenciável com suporte compacto e integral unitária.

A aproximação por partículas do método SPH consiste na discretização da representação integral por uma soma em uma coleção de pontos (ou partículas) distribuídos arbitrariamente sobre o domínio do problema. A condição de compacidade determina que somente um número finito de partículas são consideradas na aproximação. O domínio efetivo do núcleo W_h em um ponto $\mathbf{u} \in \Omega$ é dado por

[9] Os métodos sem malha visam obter soluções numéricas estáveis e precisas em um conjunto de partículas sem o uso de qualquer malha que defina conectividade entre as mesmas.

[10] No caso de escoamento de fluidos, este é representado por uma coleção de elementos de fluido (partículas) onde cada um destes elementos se desloca com o escoamento. Na abordagem lagrangeana o referencial se desloca junto com o escoamento.

$$V(\mathbf{u}) = \{\mathbf{x} \in \mathbb{R}^2; \|\mathbf{x} - \mathbf{u}\| \leq \kappa h\} \tag{2.34}$$

e é chamado de suporte compacto de $\mathbf{u}$. Assim, a aproximação de $f(\mathbf{u})$ é feita tomando-se uma média dos valores de $f(\mathbf{x})$ obtidos pela contribuição das partículas vizinhas (NNP[11]) $\mathbf{x}$ da partícula $\mathbf{u}$:

$$f^h(\mathbf{u}) = \sum_{\mathbf{x} \in V(\mathbf{u})} f(\mathbf{x}) W_h(\mathbf{u} - \mathbf{x}) V_x, \tag{2.35}$$

onde o volume finito V_x que substitui $d\Omega$ é a medida do suporte compacto em torno da partícula $\mathbf{x}$.

O êxito na aproximação depende diretamente do comprimento suave h, que determina o número de partículas vizinhas utilizadas na equação (2.35). Um valor pequeno de h pode resultar em um número insuficiente de partículas vizinhas no suporte compacto. Por outro lado, se for escolhido um valor grande para h, haverá suavização de propriedades locais [PPLT09]. Por este motivo, a determinação dos NNP deve ocorrer antes da computação do método SPH. Existem duas abordagens para o problema da determinação de partículas vizinhas: a abordagem por agrupamento, que será apresentada no capítulo 3, e a abordagem por dispersão, utilizada na formulação do algoritmo SPIR.

Na abordagem por dispersão, o comprimento suave h_j se refere ao raio do suporte compacto da partícula $\mathbf{x}_j$, e esta é considerada uma das NNP se a partícula $\mathbf{u}$ pertencer ao seu suporte compacto.

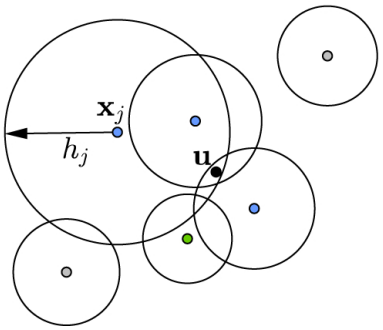


Figura 2.12: Abordagem por dispersão. A partícula $\mathbf{x}_j$ é uma NNP se a partícula $\mathbf{u}$ se encontra dentro do suporte compacto $V(\mathbf{x}_j)$. Observe que a terceira partícula mais próxima (verde) não foi classificada como uma NNP (azul).

[11] Do inglês *nearest neighboring particles*.

2.4.2 O algoritmo

No que concerne ao processamento de imagens, o algoritmo SPIR (*Smoothed Particle Image Restoration*), proposto por DiBlasi et al [DBFTT11], aplica o método SPH ao problema de remoção de ruído. Neste contexto, "partículas" são *pixels*[12], e a função $f(\mathbf{x})$ em um domínio $\Omega \subset \mathbb{R}^2$ corresponde à imagem digital

$$I(\mathbf{x}) : [1, a] \times [1, b] \subset \mathbb{N}^2 \to [0, 1]^3 \subset \mathbb{R}^3, \tag{2.36}$$

onde $a \cdot b$ corresponde ao tamanho da imagem. Assim, a equação (2.35) pode ser reescrita na forma

$$I^h(\mathbf{u}) \cong \sum_{j=1}^{N} I(\mathbf{x}_j) W(\mathbf{u} - \mathbf{x}_j, h_j) V_j. \tag{2.37}$$

Ao iniciarmos a reconstrução da imagem, é necessário ajustar os comprimentos suaves h_j segundo o modelo de abordagem por dispersão. Tal abordagem foi escolhida com o intuito de evitar grandes variações de densidade de partículas que ocorrem em certos problemas, conduzindo a núcleos suaves desequilibrados [LL03]. Para que se obtenha um número suficiente de partículas vizinhas sem que se perca informações locais tomando h_j demasiadamente grandes, um valor inicial $h_j^{(1)}$ é fixado. Então, para cada *pixel* a ser recuperado, um valor de cor é atribuído ao mesmo se um número mínimo de NNP for satisfeito. O processo continua iterativamente até que todos os *pixels* do domínio sejam restaurados.

A cada iteração, é natural a ocorrência de *pixels* para os quais o número mínimo de NNP não foi atingido. Neste caso, nenhum valor de cor é atribuído ao mesmo. Os valores dos h_j são então atualizados segundo uma recorrência:

$$h_j^{(i)} = g(i, h_j^{(1)}, \ldots, h_j^{(i-1)}),\ j = 1, \ldots, N, \tag{2.38}$$

onde g é alguma função tal que as sequências $(h_j^{(i)})_{i=1}^n,\ j = 1, \ldots, N$ são estritamente crescentes. Em seguida, os *pixels* não recuperados são reavaliados quanto ao número mínimo de NNP na próxima iteração.

2.4.3 Detalhes de implementação

O suporte compacto V_j da partícula $\mathbf{x}_j$ foi definido como

$$V_j = \min_{k=1,\ldots,N} \{V_{jk};\ V_{jk} \geq 1\},\ k \neq j, \tag{2.39}$$

[12] A partir daqui usaremos, sem o risco de gerar confusões, os termos "partícula(s)" e *pixel(s)* para fazer referência ao mesmo sujeito.

onde

$$V_{jk} = |x_{\mathbf{x}_j} - x_{\mathbf{x}_k}| \cdot |y_{\mathbf{x}_j} - y_{\mathbf{x}_k}|. \tag{2.40}$$

Uma vez calculado V_j, o valor do comprimento suave h_j correspondente é determinado por

$$h_j = \sqrt{(x_{\mathbf{x}_j} - x_{\mathbf{x}_k})^2 + (y_{\mathbf{x}_j} - y_{\mathbf{x}_k})^2}. \tag{2.41}$$

A determinação dos NNP é efetuada em uma grade, uma matriz adaptativa $n \times n$ centrada em $\mathbf{u}$, tal que $n \cdot n$ seja maior ou igual ao número mínimo de NNP. Estima-se que, no contexto (caso bidimensional do método SPH), este número mínimo deve variar em torno de 21 partículas [LL03]. O tamanho da grade aumenta iterativamente enquanto o número mínimo de NNP não for atingido.

Para o núcleo suave foram utilizadas a função gaussiana [DBFTT11]

$$W(R_j, h_j) = \frac{1}{\pi h_j^2} e^{-R_j^2}, \tag{2.42}$$

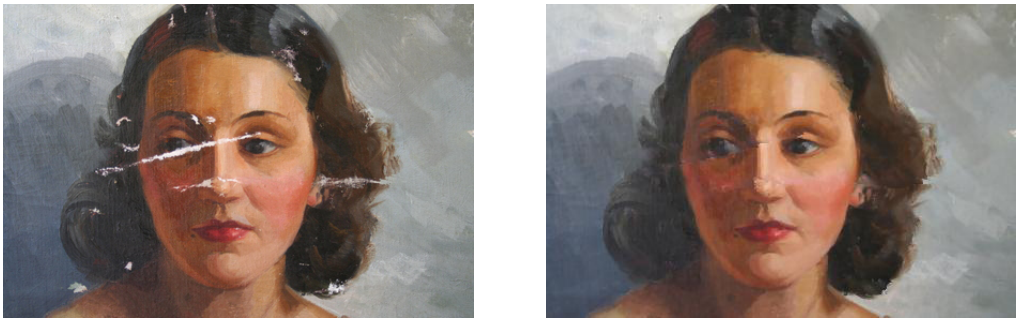
e o spline quíntico [PPLT09]

$$W(R_j, h_j) = \frac{7}{478\pi h_j^2} \cdot \begin{cases} (3 - R_j)^5 - 6(2 - R_j)^5 - 15(1 - R_j)^5, & \text{se } 0 \leq R_j < 1 \\ (3 - R_j)^5 - 6(2 - R_j)^5, & \text{se } 1 \leq R_j < 2 \\ (3 - R_j)^5, & \text{se } 2 \leq R_j < 3 \\ 0, & \text{se } R_j \geq 3 \end{cases}, \tag{2.43}$$

onde $R_j = \|\mathbf{u} - \mathbf{x}_j\|/h_j$, $j = 1, \ldots, N$. A função g de atualização dos $h_j^{(i)}$ é dada por

$$g(i, h_j^{(1)}, \ldots, h_j^{(i-1)}) = i \cdot h_j^{(1)}. \tag{2.44}$$

O algoritmo SPIR é descrito pelo pseudo-código apresentado abaixo.



(a) (b)

Figura 2.13: Restauração da imagem corrompida (Figura 2.1) obtida com o algoritmo SPIR.

input : imagem corrompida, $i = 1$, Ω
output: imagem restaurada

Determine as medidas dos suportes compactos de cada $\mathbf{x}_j$ e seus respectivos $h_j^{(1)}$, $j = 1, \ldots, N$;
while *existem pixels em* Ω *não restaurados* **do**
    for *cada pixel a restaurar* **do**
        Verifique o número mínimo de NNP;
        if *número mínimo de NNP é satisfeito* **then**
            Determine a cor do *pixel* segundo o método SPH;
        else
            Deixe o *pixel* sem cor;
        end
    end
    $i = i + 1$;
    $h_j^{(i)} = i \cdot h_j^{(1)}$, $j = 1, \ldots, N$;
end

Algoritmo 1: Algoritmo SPIR.

Capítulo 3

Retoque digital com o algoritmo G-SPIR

Neste capítulo apresentamos um novo algoritmo de retoque digital que consiste na aplicação do método SPH utilizando a abordagem por agrupamento na etapa de aproximação por partículas, escolha esta que levou a resultados muito próximos dos obtidos com os métodos conhecidos na literatura do retoque digital. Estes resultados serão apresentados no capítulo 4.

3.1 Fundamentos

SPH e a abordagem por agrupamento

A abordagem por dispersão adotada na formulação do algoritmo SPIR (seção 2.4) é eficiente na remoção de ruídos, uma vez que os *pixels* a serem restaurados estão cercados por *pixels* com informação de cor. No entanto, torna-se ineficaz na restauração de um grande número de *pixels* reunidos, evento frequente no problema do retoque digital.

Neste capítulo apresentamos o algoritmo G-SPIR, que aplica o método SPH (subseção 2.4.1) ao contexto do retoque digital. Para isto adotamos a abordagem por agrupamento na determinação das partículas vizinhas mais próximas (Os NNP mencionados na seção 2.4). Nesta abordagem, o comprimento suave h se refere ao raio do suporte compacto da partícula $\mathbf{u}$, e seus vizinhos correspondem às partículas $\mathbf{x}_j$ contidas neste suporte (Figura 3.1). Desta forma, fixado um inteiro positivo k, o valor de h é atualizado gradativamente até que se alcance k partículas vizinhas contidas no suporte de $\mathbf{u}$.

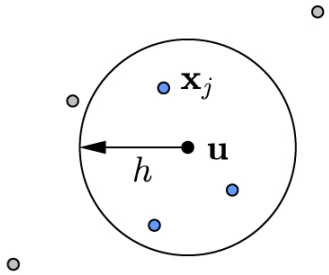


Figura 3.1: Abordagem por agrupamento. A partícula $\mathbf{x}_j$ é uma NNP (azul) se a mesma se encontra dentro do suporte compacto $V(\mathbf{u})$.

3.2 O algoritmo

Dada uma imagem degradada, o usuário fornece uma máscara que especifica as regiões (Ω) a serem restauradas e um inteiro positivo k. Para cada pixel em Ω, o algoritmo efetua a busca dos seus k vizinhos mais próximos (NNP), pela abordagem por agrupamento, e o SPH atribui uma cor ao pixel com base na interpolação das cores desses vizinhos. O processo é ilustrado abaixo (Figura 3.2).

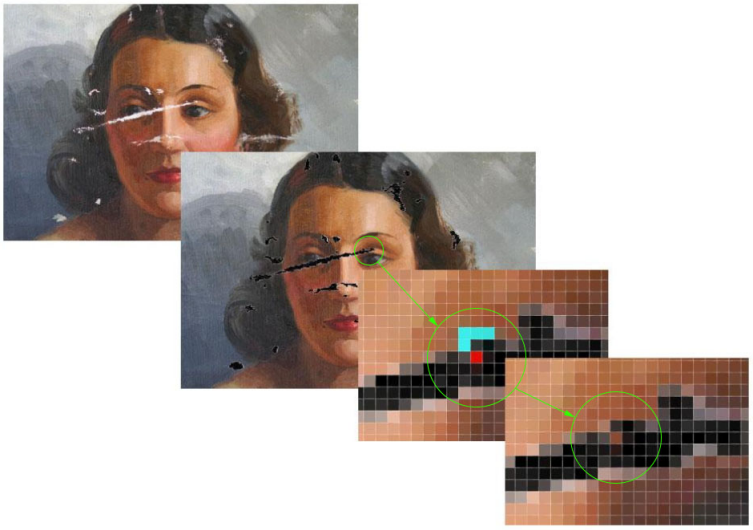


Figura 3.2: Etapas do retoque com o G-SPIR. De cima para baixo: imagem degradada; definição da máscara; determinação dos vizinhos mais próximos (azul); atribuição de cor ao pixel retocado (vermelho) pelo método SPH.

No algoritmo G-SPIR, a imagem degradada

$$I : [1, a] \times [1, b] \subset \mathbb{N}^2 \to [0, 1] \subset \mathbb{R}, \tag{3.1}$$

é discretizada em um conjunto de partículas $\{p_1, p_2, \cdots, p_n\}$, com $n = a \cdot b$, onde cada

partícula p_k representa o *pixel* de coordenadas $(i_k, j_k) \in [1, a] \times [1, b]$ e a cor $I_k = I(i_k, j_k)$ armazenada neste *pixel*.

Além disso, o domínio de retoque Ω é definido a partir de uma imagem

$$M : [1, a] \times [1, b] \subset \mathbb{N}^2 \to [0, 1] \subset \mathbb{R}, \tag{3.2}$$

na qual é atribuído valor nulo somente para os *pixels* a serem retocados. Ou seja,

$$p_k \in \Omega \Leftrightarrow M(i_k, j_k) = 0.$$

Uma partícula $p_k \in \Omega$ é restaurada a partir da equação

$$I_k = \sum_{j \in N_k} I_j W_{kj} V_j, \tag{3.3}$$

onde a soma percorre todo o conjunto de índices das partículas vizinhas mais próximas de p_k, denotado por N_k, $W_{kj} = W_h(r_{kj})$ com r_{kj} dado pela distância entre as partículas p_k e p_j, e V_j é o elemento de área da partícula p_j.

3.3 Detalhes de implementação

A determinação dos k vizinhos mais próximos deve ser feita antes de qualquer computação do método SPH. Uma vez que se tenha determinado os N_i de cada partícula p_i no domínio de retoque Ω, os comprimentos suaves h_i são dados por

$$h_i = \alpha \max\{d_{ij};\ j \in N_i\}, \tag{3.4}$$

onde d_{ij} é a distância euclidiana entre as partículas p_i e p_j, e α é um parâmetro de escala do comprimento suave, evitando que a contribuição de partículas distantes de p_i seja praticamente nula. O elemento de área V_j da partícula p_j é dado por $V_j = 1/n$.

O elemento de área V_j é o mesmo para todas as partículas p_j, e foi definido desta forma afim de evitar o desequilíbrio da equação (3.3), mesmo quando uma grande região da imagem é restaurada, permitindo assim a utilização do nosso método em casos tradicionais de retoque digital.

As funções de núcleo suave utilizadas foram a função gaussiana

$$W(R_j, h_i) = \frac{1}{\pi h_i^2} e^{-R_j^2} \tag{3.5}$$

e o spline quíntico

$$W(R_j, h_i) = \frac{7}{478\pi h_i^2} \cdot \begin{cases} (3-R_j)^5 - 6(2-R_j)^5 - 15(1-R_j)^5, & \text{se } 0 \leq R_j < 1 \\ (3-R_j)^5 - 6(2-R_j)^5, & \text{se } 1 \leq R_j < 2 \\ (3-R_j)^5, & \text{se } 2 \leq R_j < 3 \\ 0, & \text{se } R_j \geq 3 \end{cases} \tag{3.6}$$

onde $R_j = \|p_i - p_j\|$. A simples formulação da técnica é traduzida pelo pseudo-código abaixo que descreve o algoritmo G-SPIR.

input : I = imagem corrompida, k = número de vizinhos mais próximos (NNP), Ω = máscara (M)
output: R = imagem restaurada

`/* Determinar os NNP de cada pixel corrompido` p_i `*/`

for *cada pixel* p_i *a restaurar* **do**
  Determine N_i com $N_i \cap \Omega = \emptyset$
end

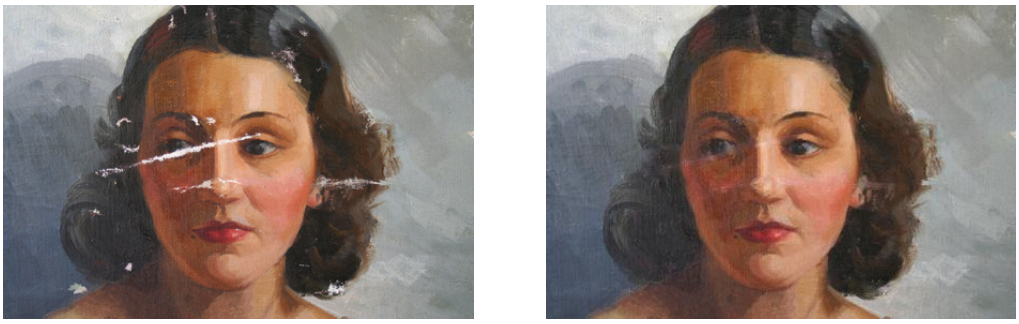
`/* Iniciar imagem restaurada: copiar pixels fora de` Ω `*/`

$R = I$;

`/* Avaliar o atributo de cor de cada pixel corrompido` p_i `*/`

for $p_i \in \Omega$ **do**
  Determine $R(p_i)$ com a equação (3.3)
end

Algoritmo 2: Algoritmo G-SPIR.



(a) (b)

Figura 3.3: Restauração da imagem corrompida (Figura 2.1) obtida com o algoritmo G-SPIR.

Capítulo 4

Resultados

Neste capítulo apresentamos alguns resultados obtidos com a aplicação dos algoritmos descritos nos Capítulos 2 e 3. Os algoritmos foram implementados no software MATLAB®, e todos os testes foram realizados com um processador Intel® Core™ i3 2.13 GHz com 4 GB de memória RAM.

Em todas as implementações foi utilizada a difusão anisotrópica com $0 < \kappa \leq 0.03$. Além da imagem corrompida e da máscara de retoque, foram usados como dados de entrada:

No algoritmo BSCB: taxa de aprimoramento Δt, número máximo de iterações e limiar[1];

No algoritmo *Fast*: número máximo de iterações e limiar;

No algoritmo NS: número máximo de iterações, taxa de aprimoramento Δt e limiar;

No algoritmo SPIR: número de vizinhos NNP;

No algoritmo G-SPIR: k, o número de vizinhos mais próximos.

O limiar foi definido como $\epsilon = 10^{-5}$ para todos os métodos iterativos. A taxa de aprimoramento foi definida como $\Delta t = 0.1$, como mencionado na subseção 2.1.3.

Nos testes 1 a 3 (com exceção das duas primeiras imagens do teste 3), os defeitos a serem removidos das imagens foram criados artificialmente com o objetivo de viabilizar análises quantitativas dos retoques por meio do cálculo do erro quadrático médio (MSE[2]) e do índice de similaridade (SSIM[3]) entre a imagem retocada e a imagem original (sem a adição do defeito).

As formas geométricas presentes nas imagens sintéticas foram usadas com o intuito de investigar a recuperação de contornos. As duas primeiras imagens do teste 3 são fotos antigas que foram usadas nos testes apresentados em Bertalmio et al [BSCB00] e Oliveira et al [OBMC01]. Nestes casos, por não haver uma imagem original, sem degradação, que

[1] O limiar corresponde a um $\epsilon > 0$ utilizado como critério de parada: $\|I_t - I_{t-1}\| \leq \epsilon$.

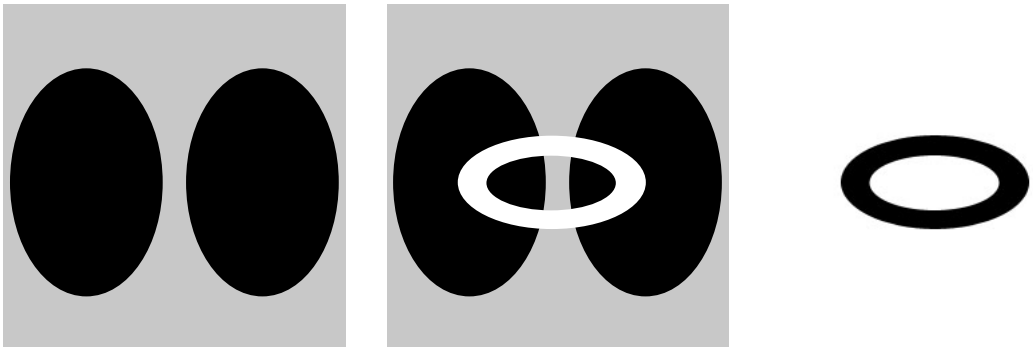
[2] Do inglês *Mean Square Error*.

[3] Do inglês *Structural Similarity Index*.

servisse de referência, o MSE e o SSIM foram calculados em relação às imagens restauradas obtidas com os respectivos algoritmos de cada artigo (BSCB e *Fast*, respectivamente). O teste 4 se refere à remoção de objetos em imagens reais, impossibilitando uma avaliação quantitativa através do MSE ou do SSIM. Nestes casos, apresentamos somente os resultados acompanhados dos respectivos tempos de execução. Em cada teste o tamanho da imagem está expresso em *pixels* e o número de pixels da região de retoque está denotado por $\#\Omega$.

4.1 Teste 1: Remoção de objetos em imagens sintéticas

O objetivo é remover o objeto em cor branca. A primeira imagem deste teste é uma reprodução da imagem sintética encontrada em Bertalmio et al [BSCB00]. O tamanho da imagem é 256×256 e $\#\Omega = 4628$.



(a) (b) (c)

Figura 4.1: (a) Imagem original, (b) imagem corrompida, (c) máscara de retoque.

O algoritmo BSCB forneceu um resultado visual satisfatório (Figura 4.2(a)), enquanto que a imagem obtida com a versão mais simples do algoritmo *Fast* apresentou um borramento sobre o domínio de retoque (Figura 4.2(b)). O algoritmo NS produziu o mesmo defeito (Figura 4.2(d)), porém mais intenso. Este defeito desapareceu no resultado do algoritmo *Fast* quando utilizamos barreiras de difusão (subseção 2.2.3) nas interseções entre a fronteira do domínio de retoque e as bordas de alto contraste (Figura 4.2(c)). A imagem resultante também apresentou o menor MSE entre todas as demais obtidas neste teste[4]. A tabela 4.1 mostra o desempenho destes algoritmos[5].

Quanto ao algoritmo SPIR (Figura 4.3), para todos os testes apresentados neste capítulo, foram realizados retoques com diferentes valores para o número de NNP (de 1 a 100). Os melhores resultados foram obtidos com valores em torno de 4 e 8 NNP. Em

[4]Nos testes seguintes, apresentaremos somente os resultados do algoritmo *Fast* com o uso de barreiras de difusão.

[5]*Fast*(BD) se refere ao retoque com o uso de barreiras de difusão.

função das semelhanças no aspecto visual e da pequena variação dos MSE e SSIM obtidos nos resultados, adotamos os valores 4 e 8 como padrão.

No teste com 4 vizinhos NNP (Figura 4.3(a)), verificamos que o preenchimento do domínio de retoque gerou bordas de alto contraste em diagonal. Isto se deve à forma como o algoritmo percorre o domínio de retoque. Esta fronteira indesejada desaparece com o aumento do número de vizinhos para 8 NNP. Infelizmente, o aumento do número de vizinhos acarreta na utilização de *pixels* muito distantes do *pixel* retocado, tornando a contribuição destes vizinhos praticamente nula (devido à compacidade do núcleo suave), e consequentemente escurecendo o domínio de retoque (Figura 4.3(b)).

Percorrer randomicamente os *pixels* domínio de retoque pode ser uma alternativa (Figura 4.3(c)), e realmente atenua o efeito indesejado obtido na figura 4.3(b). No entanto, o tempo de execução do teste aumenta drasticamente. O desempenho do algoritmo nos três casos supracitados é mostrado na tabela 4.2 (NNP = 8(*) corresponde ao teste com 8 vizinhos e varredura randômica do domínio de retoque).

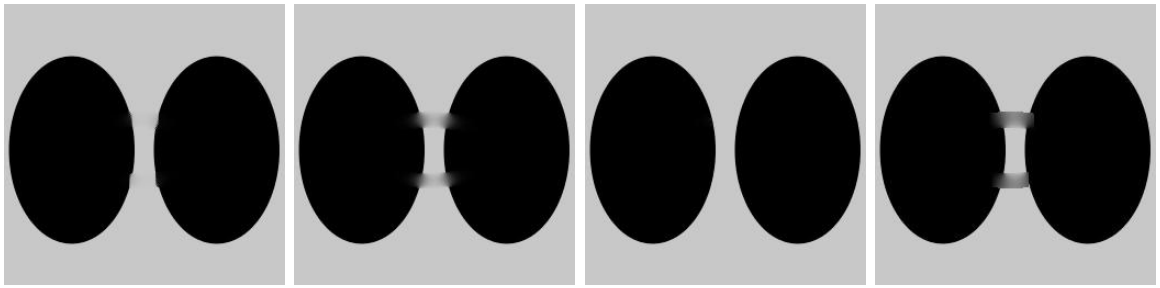
Quanto ao algoritmo G-SPIR, aqui mostramos as imagens obtidas com $k = 4$, $k = 8$ e $k = 21$, e os resultados visuais estão muito próximos do obtido pelo algoritmo BSCB (Figura 4.2(a)). Não houve variações significativas no tempo de execução para diferentes valores de k. Porém, a medida que incrementamos k, também aumentou o borramento na região de interseção do domínio de retoque com as bordas de alto contraste. O valor do MSE foi menor para $4 \leq k \leq 8$, e aumentou gradativamente para $k \geq 9$.

A presença deste efeito indesejável se assemelha àquele obtido com a versão mais simples do algoritmo *Fast*, e isto nos motivou a também utilizar barreiras de difusão na execução do algoritmo G-SPIR. Os resultados obtidos com o uso das barreiras podem ser vistos na figura 4.5.

A melhora nos resultados também é constatada pelos valores do MSE apresentados na tabela 4.4. Por este motivo, nos testes seguintes também apresentaremos somente os resultados do algoritmo G-SPIR onde foram utilizadas barreiras de difusão.

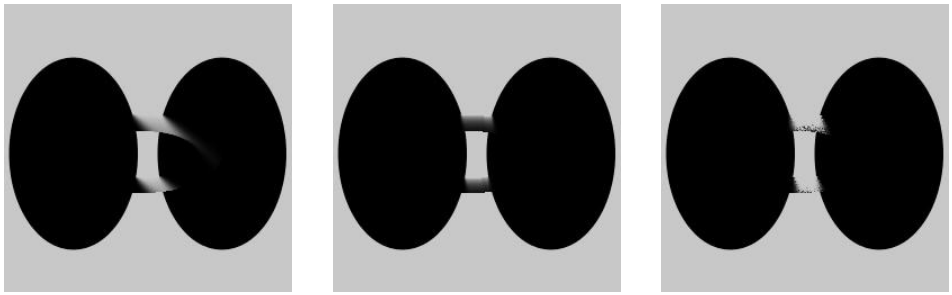
De modo análogo ao algoritmo SPIR, padronizamos $k = 4$, 8 e 21 nos testes seguintes com o algoritmo G-SPIR. Os resultados obtidos com valores de k próximos a estes conduziram a resultados visuais e valores de MSE muito próximos aos apresentados aqui.

A tabela 4.5 reúne os melhores resultados obtidos com cada algoritmo quanto ao valor do SSIM. Os resultados dos algoritmos *Fast* e G-SPIR apresentados nesta tabela são os obtidos com o uso de barreiras de difusão.



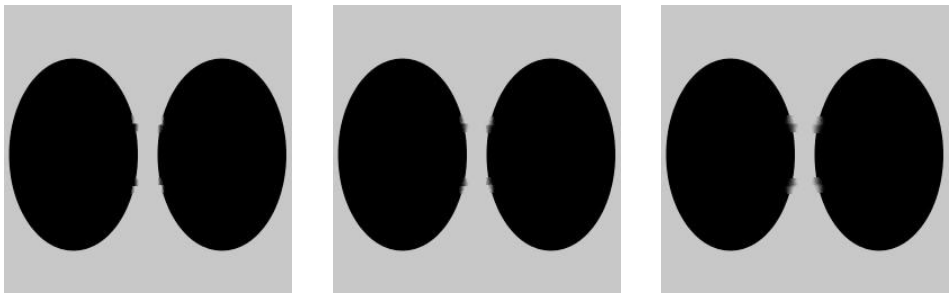
(a) (b) (c) (d)

Figura 4.2: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo *Fast* com o uso de barreiras de difusão; (d) Algoritmo NS.



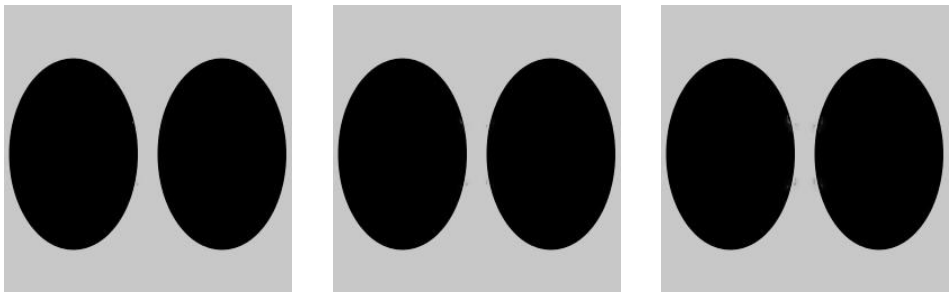
(a) (b) (c)

Figura 4.3: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.



(a) (b) (c)

Figura 4.4: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.



(a) (b) (c)

Figura 4.5: Efeito da utilização de barreiras de difusão na execução do G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.1: Desempenho dos algoritmos BSCB, *Fast* e NS.

Algoritmo	**Iterações**	**Tempo decorrido**	**MSE**
BSCB	50000	627.133883 s	16.7087
Fast	736	11.011522 s	39.3204
NS	4200	4060.799628 s	72.9722
Fast (BD)	372	30.994684 s	0.7059

Tabela 4.2: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**MSE**
4	18.191166 s	185.6311
8	45.358134 s	197.3078
8(*)	469.172638 s	81.8430

Tabela 4.3: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**MSE**
4	1.242924 s	16.9163
8	0.500294 s	16.1391
21	0.657558 s	21.8469

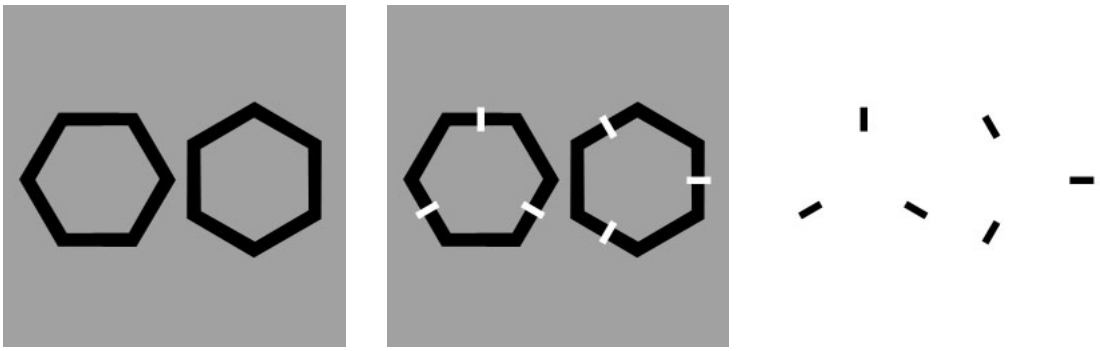
Tabela 4.4: Desempenho do algoritmo G-SPIR com o uso de barreiras de difusão.

k	**Tempo decorrido**	**MSE**
4	0.555198 s	0.9592
8	0.583374 s	1.2022
21	0.761178 s	2.4934

Tabela 4.5: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	627.133883 s	0.9806
Fast	11.011522 s	0.9847
NS	4060.799628 s	0.9676
SPIR(8(*) NNP)	469.172638 s	0.9732
G-SPIR(k=4)	0.555198 s	0.9879

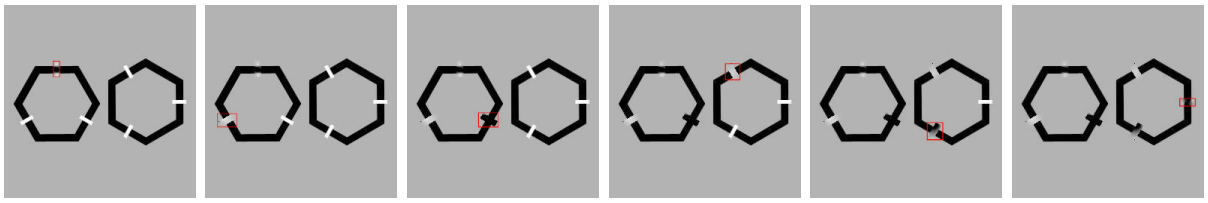
A segunda imagem do teste 1 foi idealizada para verificar a eficiência na conexão de bordas em diferentes ângulos. O tamanho da imagem é 256×256 e $\#\Omega = 1153$.



(a) (b) (c)

Figura 4.6: (a) Imagem original, (b) imagem corrompida, (c) máscara de retoque.

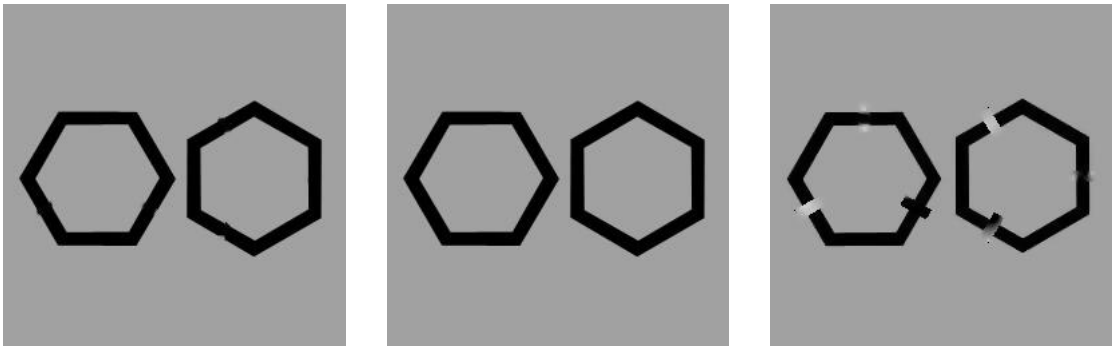
Os algoritmos *Fast* e G-SPIR apresentaram resultados visuais semelhantes, efetivamente fazendo a conexão das arestas, sendo que o menor MSE e o maior SSIM se devem ao *Fast*. O algoritmo BSCB apresentou pequenas falhas na conexão das arestas em diagonal, e os algoritmos NS e SPIR não foram bem sucedidos no retoque. O retoque com algoritmo NS só fez a conexão das arestas horizontais e verticais. Além disso, a utilização de um único retângulo $\bar{\Omega}$ contendo todo o domínio de retoque Ω (Ver seção 2.3.3) provocou estouro de memória durante a execução do algoritmo. O resultado final (Figura 4.8(c)) é a composição de seis retoques parciais com retângulos $\bar{\Omega}$ menores (Figura 4.7).



(a) (b) (c) (d) (e) (f)

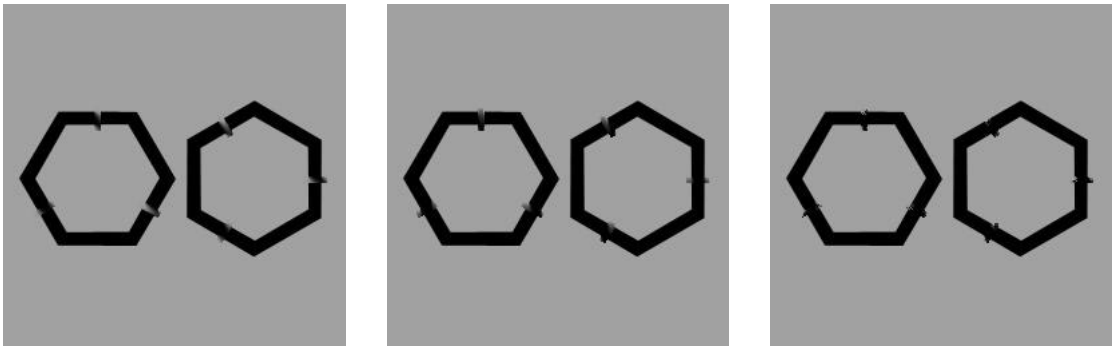
Figura 4.7: Etapas do retoque que gerou o resultado da Figura 4.8(c). As janelas em vermelho correspondem aos retângulos $\bar{\Omega}$ utilizados em cada etapa.

Os resultados são mostrados a seguir.



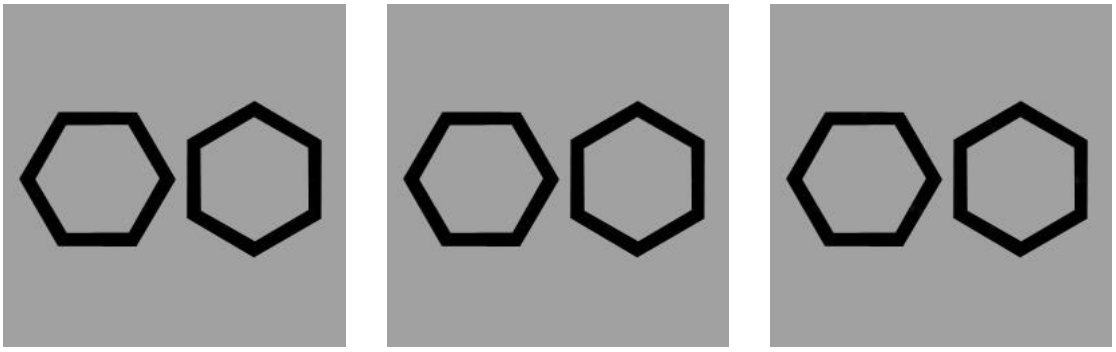
(a) (b) (c)

Figura 4.8: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo NS.



(a) (b) (c)

Figura 4.9: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.



(a) (b) (c)

Figura 4.10: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.6: Desempenho dos algoritmos BSCB, *Fast* e NS.

Algoritmo	**Iterações**	**Tempo decorrido**	**MSE**
BSCB	50000	605.114918 s	10.1264
Fast	54	1.122744 s	0.8210
NS	4200	1813.455137 s	492.5242

Tabela 4.7: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**MSE**
4	3.70510 s	36.6717
8	10.886179 s	37.5518
8(*)	19.230554 s	36.3059

Tabela 4.8: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

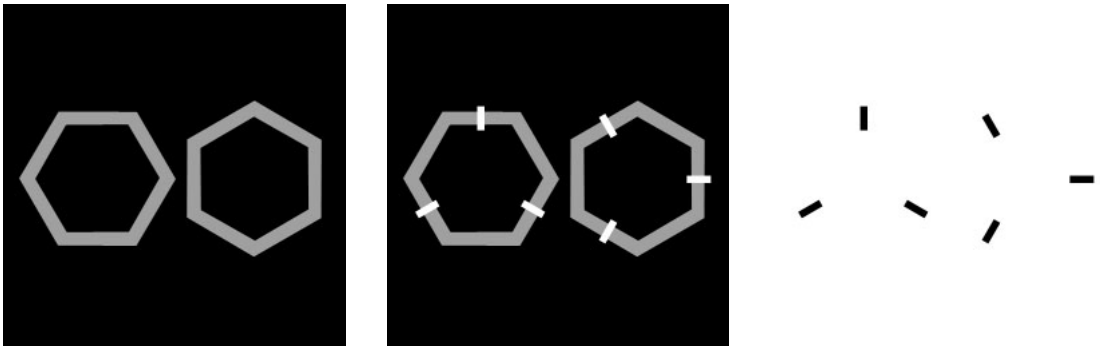
k	**Tempo decorrido**	**MSE**
4	0.440036 s	0.9215
8	0.328967 s	0.9329
21	0.398377 s	1.1737

Tabela 4.9: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	605.114918 s	0.9838
Fast	1.122744 s	0.9862
NS	1813.455137 s	0.8945
SPIR(4 NNP)	3.70510 s	0.9768
G-SPIR(k=8)	0.328967 s	0.9860

A terceira imagem do teste 1 corresponde à segunda imagem com uma inversão das cores de fundo e das figuras geométricas.

Os resultados não diferiram significativamente dos obtidos com a imagem anterior, a menos do aumento dos valores do erro para o algoritmo SPIR.



(a) (b) (c)

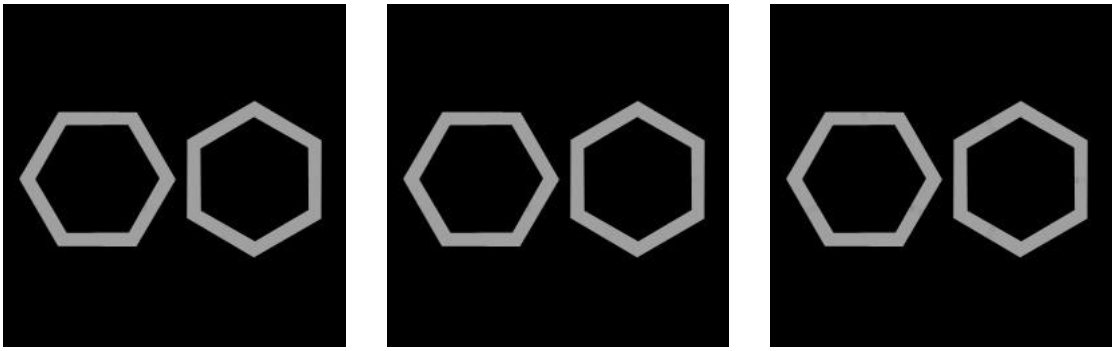
Figura 4.11: (a) Imagem original, (b) imagem corrompida, (c) máscara de retoque.

(a) (b) (c)

Figura 4.12: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo NS.

(a) (b) (c)

Figura 4.13: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.



(a) (b) (c)

Figura 4.14: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.10: Desempenho dos algoritmos BSCB, *Fast* e NS.

Algoritmo	**Iterações**	**Tempo decorrido**	**MSE**
BSCB	50000	826.723492 s	10.6795
Fast	86	1.864730 s	0.8444
NS	4200	2916.912137 s	121.3798

Tabela 4.11: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**MSE**
4	3.456865 s	37.7894
8	10.416311 s	56.9676
8(*)	19.266000 s	66.1770

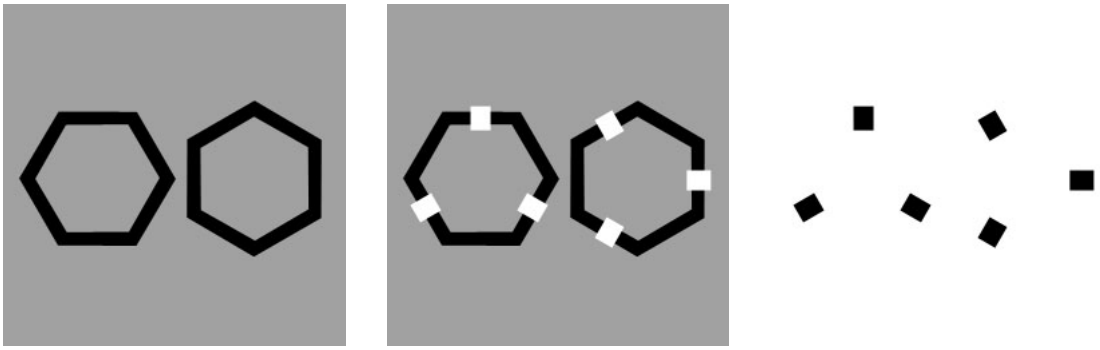
Tabela 4.12: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**MSE**
4	0.323201 s	0.9429
8	0.362361 s	0.9555
21	0.332108 s	1.1988

Tabela 4.13: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	826.723492 s	0.8868
Fast	1.864730 s	0.8873
NS	2916.912137 s	0.8818
SPIR(4 NNP)	3.456865 s	0.8855
G-SPIR(k=4)	0.323201 s	0.8873

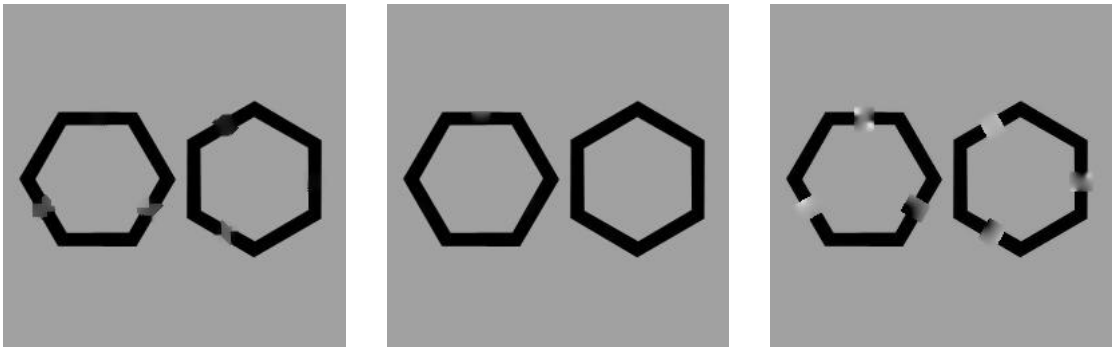
A quarta e última imagem corresponde à segunda imagem do teste 1, mas o domínio de retoque possui uma área maior ($\#\Omega = 1611$).



(a) (b) (c)

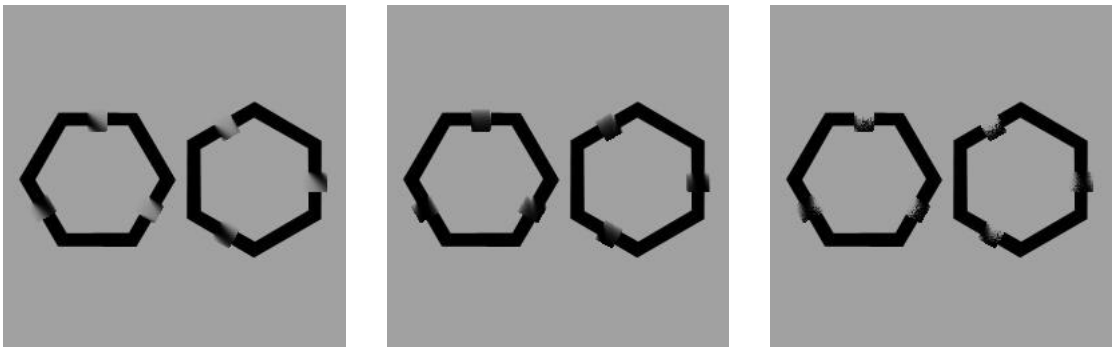
Figura 4.15: (a) Imagem original, (b) imagem corrompida, (c) máscara de retoque.

Apenas o algoritmo *Fast* fez conexão satisfatória. O algoritmo BSCB fez a conexão apenas das bordas horizontal e vertical, o algoritmo G-SPIR fez conexão incompleta em todos os ângulos, e os algoritmos NS e SPIR não fizeram conexão alguma. Quanto ao retoque com o algoritmo NS, o aumento no tamanho dos retângulos $\bar{\Omega}$ provocaram um aumento considerável no tempo de execução (Ver tabelas 4.6 e 4.14 para comparação).



(a) (b) (c)

Figura 4.16: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo NS.



(a) (b) (c)

Figura 4.17: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.

(a) (b) (c)

Figura 4.18: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.14: Desempenho dos algoritmos BSCB, *Fast* e NS.

Algoritmo	**Iterações**	**Tempo decorrido**	**MSE**
BSCB	50000	681.425273 s	61.0812
Fast	172	5.647021 s	4.2378
NS	4200	8855.968255 s	268.3397

Tabela 4.15: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**MSE**
4	6.999796 s	165.9593
8	19.776753 s	135.6589
8(*)	114.001117 s	197.0343

Tabela 4.16: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

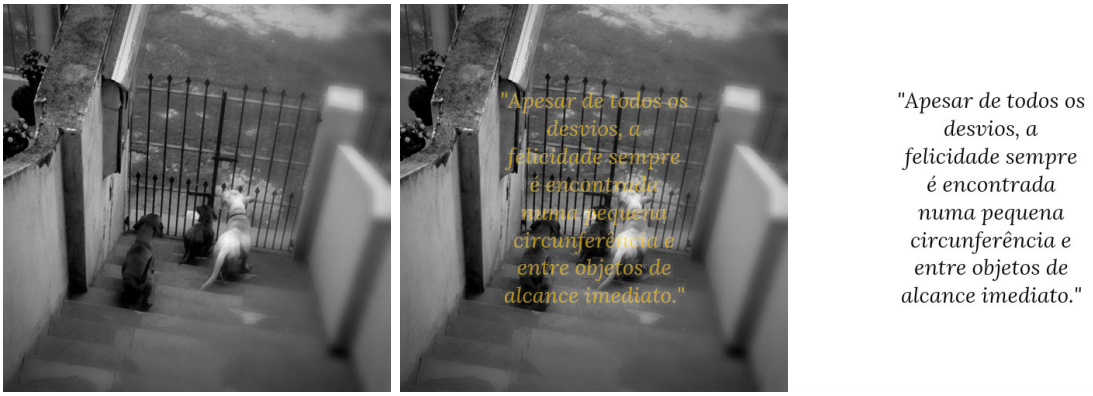
k	**Tempo decorrido**	**MSE**
4	0.365271 s	23.2302
8	0.482135 s	21.4957
21	0.468224 s	22.9062

Tabela 4.17: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	681.425273 s	0.9712
Fast	5.647021 s	0.9839
NS	8855.968255 s	0.9555
SPIR(4 NNP)	6.999796 s	0.9609
G-SPIR(k=4)	0.365271 s	0.9732

4.2 Teste 2: Remoção de texto sobre imagem

Neste teste avaliamos a eficiência na reconstrução de uma imagem por meio da remoção de texto sobreposto. Neste caso a fronteira do domínio de retoque é bastante irregular. Abaixo vemos a imagem original, a imagem com o texto sobreposto e a máscara de retoque. O tamanho das imagens é 500×500 e $\#\Omega = 6626$.



(a) (b) (c)

Figura 4.19: (a) Imagem original; (b) Imagem com o texto sobreposto; (c) Máscara de retoque.

Os algoritmos foram aplicados a cada um dos três canais de cor da imagem. Todos obtiveram resultados visuais muito próximos da imagem original, a menos do algoritmo SPIR com 8 vizinhos, que não reconstruiu a imagem satisfatoriamente. Os menores MSE foram obtidos com o algoritmo *Fast*, enquanto que o maior SSIM foi obtido tanto com o algoritmo *Fast* quanto com o algoritmo G-SPIR. O tempo de execução do algoritmo BSCB aumentou consideravelmente (triplicado por consequência do retoque nos três canais distintos). O menor tempo de execução foi obtido pelo algoritmo G-SPIR. O MSE foi calculado para cada um dos três canais de cor. Como tanto a imagem original quanto a imagem retocada estão em tons de cinza, os erros foram iguais nos três canais, exceto para o retoque com o algoritmo SPIR e busca randômica de vizinhos, pois o algoritmo percorre os *pixels* do domínio de retoque de forma distinta a cada vez que é aplicado

(tabela 4.19). O retoque com o algoritmo NS requereu o uso de uma grande quantidade de retângulos $\bar{\Omega}$, tornando inviável a sua aplicação.

(a) (b)

Figura 4.20: (a) Algoritmo BSCB; (b) Algoritmo *Fast*.

(a) (b) (c)

Figura 4.21: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.

(a) (b) (c)

Figura 4.22: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.18: Desempenho dos algoritmos de BSCB e *Fast*.

Algoritmo	**Iterações**	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
BSCB	30000	2396.918545 s	10.9594	10.9594	10.9594
Fast	129	9.743430 s	9.9674	9.9674	9.9674

Tabela 4.19: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	159.579261 s	14.5365	14.5365	14.5365
8	513.373926 s	24.9863	24.9863	24.9863
8(*)	825.284628 s	29.7125	28.7839	30.0541

Tabela 4.20: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	4.590722 s	10.3443	10.3443	10.3443
8	4.479016 s	10.9701	10.9701	10.9701
21	5.163101 s	13.3704	13.3704	13.3704

Tabela 4.21: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	2396.918545 s	0.9615
Fast	9.743430 s	0.9620
SPIR(4 NNP)	159.579261 s	0.9584
G-SPIR(k=4)	4.590722 s	0.9620

4.3 Teste 3: Restauração de imagens degradadas

O objetivo deste teste é avaliar a eficiência dos algoritmos na restauração de fotografias que foram degradadas pela ação do manuseio e do tempo. As duas primeiras imagens foram retiradas de Bertalmio et al [BSCB00] e de Oliveira et al [OBMC01], respectivamente. Como não havia a possibilidade de calcular-se o MSE e o SSIM em relação às imagens originais (antes de sofrerem degradação), usamos como referência as imagens retocadas pelos algoritmos BSCB e *Fast* que foram apresentadas em cada um dos dois artigos. As demais imagens utilizadas foram danificadas artificialmente, permitindo-nos calcular o MSE e o SSIM em relação às imagens originais. Também foi possível variar a extensão do dano e avaliar o desempenho dos algoritmos quanto ao tamanho da região de retoque sobre imagens com textura.

Restauração de imagens antigas. Resultados obtidos com a imagem[6] extraída de Bertalmio et al [BSCB00]. Seu tamanho é 483×405 e $\#\Omega = 14258$.

(a) (b) (c)

Figura 4.23: (a) Imagem de referência para o cálculo dos MSE e SSIM; (b) Imagem original; (c) máscara. Fonte: Bertalmio et al [BSCB00].

Todos os algoritmos restauraram satisfatoriamente a fotografia, com exceção do algoritmo SPIR. Como já era esperado, o menor MSE e o maior SSIM foram obtidos com o algoritmo BSCB.

Todos os algoritmos também geraram defeitos na área dos olhos da menina do centro, por causa da perda de grande quantidade de informação da imagem original naquela região. A figura 4.27 ilustra a diferença entre as restaurações dos algoritmos de BSCB e G-SPIR.

[6]Imagem e máscara disponíveis da página `www.dtic.upf.edu/~mbertalmio/restoration2.html`.

(a) (b)

Figura 4.24: (a) Algoritmo BSCB; (b) Algoritmo *Fast*.

(a) (b) (c)

Figura 4.25: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.

(a) (b) (c)

Figura 4.26: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

(a) (b)

Figura 4.27: Detalhe do retoque na região dos olhos: (a) Algoritmo BSCB; (b) Algoritmo G-SPIR com $k = 8$.

Tabela 4.22: Desempenho dos algoritmos BSCB e *Fast*.

Algoritmo	**Iterações**	**Tempo decorrido**	**MSE**
BSCB	50000	2734.3779 s	4.9270
Fast	3179	246.2673 s	5.1812

Tabela 4.23: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**MSE**
4	64.875883 s	456.6891
8	185.701571 s	1.4509e+03
8(*)	2398.080775 s	652.7292

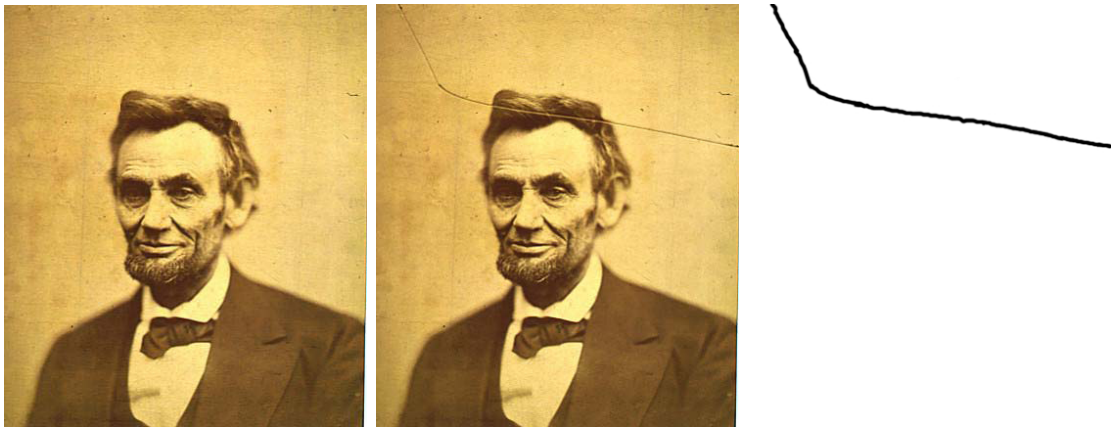
Tabela 4.24: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**MSE**
4	2.025388 s	6.0354
8	1.402278 s	6.2815
21	1.979766 s	7.1297

Tabela 4.25: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	2734.3779 s	0.9732
Fast	246.2673 s	0.9731
SPIR(4 NNP)	64.875883 s	0.9662
G-SPIR(k=4)	2.025388 s	0.9706

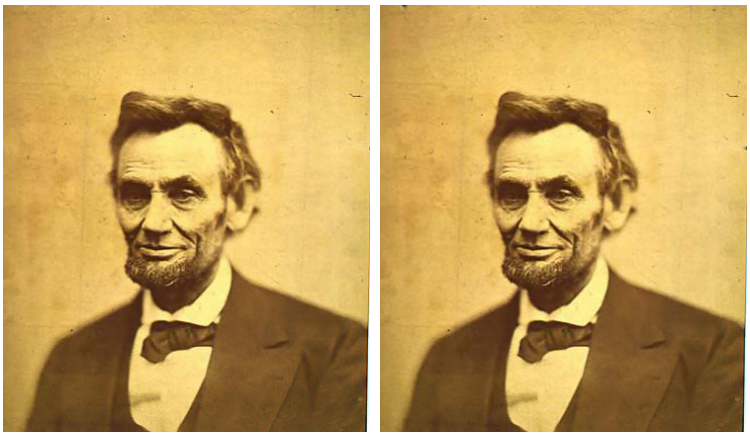
Os próximos resultados foram obtidos com a imagem extraída de Oliveira et al [OBMC01]. Seu tamanho é 371×432 e $\#\Omega = 1622$.



(a) (b) (c)

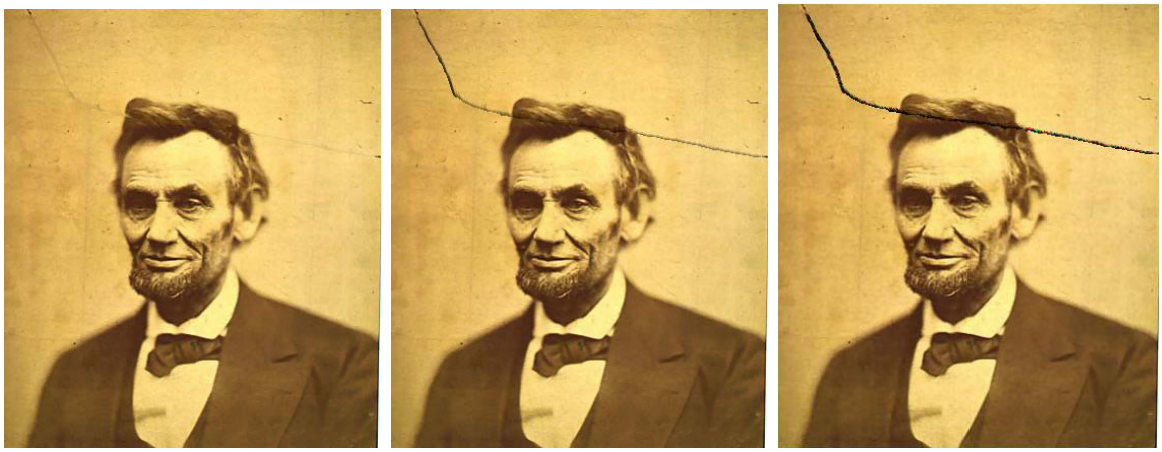
Figura 4.28: (a) Imagem de referência para o cálculo dos MSE e SSIM; (b) Imagem original; (c) máscara. Fonte: Oliveira et al [OBMC01].

Todos os algoritmos apresentaram resultados visuais satisfatórios e muito parecidos entre si, como podemos observar também pela proximidade dos valores do MSE e do SSIM. O algoritmo SPIR não obteve sucesso na restauração com 8 NNP, e podemos observar um leve defeito na fronteira do domínio de retoque na restauração com 4 NNP.



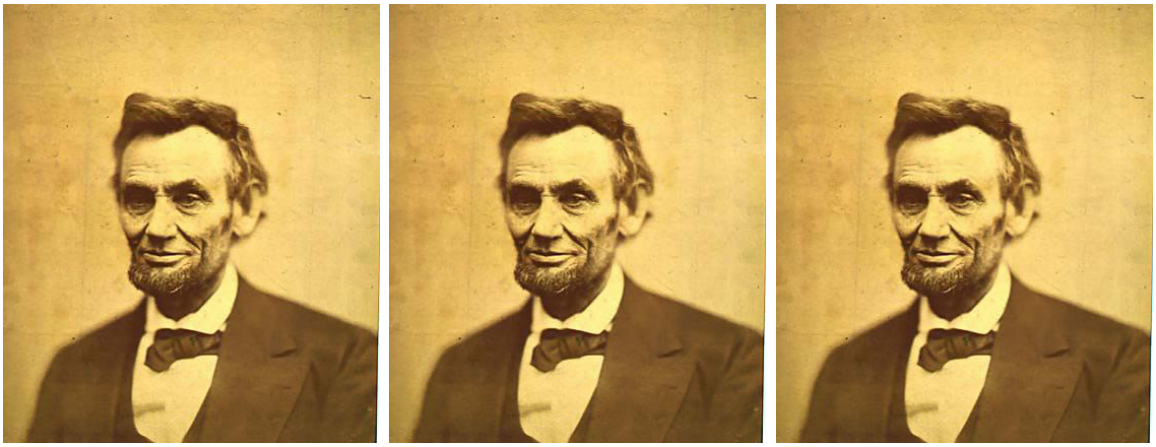
(a) (b)

Figura 4.29: (a) Algoritmo BSCB; (b) Algoritmo *Fast*.



(a) (b) (c)

Figura 4.30: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.



(a) (b) (c)

Figura 4.31: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.26: Desempenho dos algoritmos BSCB e *Fast*.

Algoritmo	**Iterações**	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
BSCB	45000	2018.257296 s	14.9438	12.9410	23.9872
Fast	319	18.336571 s	16.0177	13.6629	24.1756

Tabela 4.27: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	28.532348 s	14.5365	14.5712	24.4563
8	88.742821 s	56.3253	39.6433	30.8596
8(*)	158.015027 s	108.0913	73.6016	34.0630

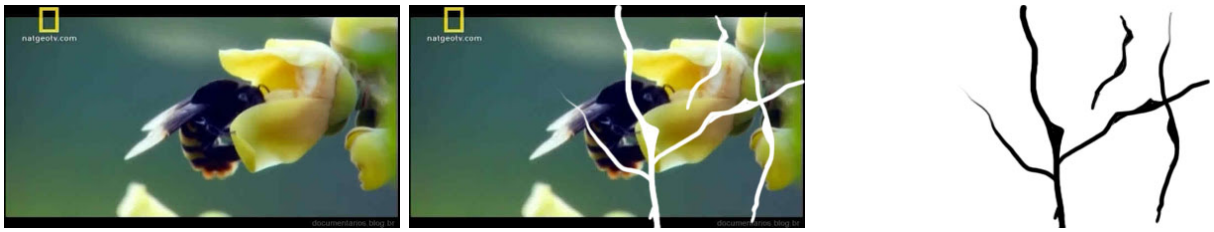
Tabela 4.28: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	3.637440 s	15.2469	13.3213	24.3777
8	2.238509 s	15.2553	13.3561	24.2616
21	2.466809 s	15.3733	13.4899	24.1927

Tabela 4.29: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	2018.257296 s	0.9954
Fast	18.336571 s	0.9953
SPIR(4 NNP)	28.532348 s	0.9952
G-SPIR(k=8)	2.238509 s	0.9953

Recuperação de linhas de contorno em imagem real. Imagem corrompida artificialmente, com desconexão moderada de linhas de contorno. Seu tamanho é 400×224 e $\#\Omega = 4256$.



(a) (b) (c)

Figura 4.32: (a) Imagem original, (b) Imagem corrompida, (c) Máscara de retoque.

O melhor resultado visual se deve ao algoritmo *Fast*. Os menores MSE são relativos ao G-SPIR, e o maior SSIM foi obtido tanto com o algoritmo G-SPIR quanto com o *Fast*. Os algoritmos BSCB e G-SPIR introduziram um leve defeito nas pétalas (próximos à abelha). Porém, estes algoritmos recuperaram satisfatoriamente as linhas de contorno interrompidas. Não foram obtidos resultados satisfatórios com o algoritmo SPIR.

(a) (b)

Figura 4.33: (a) Algoritmo BSCB, (b) Algoritmo *Fast*.

(a) (b) (c)

Figura 4.34: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.

(a) (b) (c)

Figura 4.35: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.30: Desempenho dos algoritmos de BSCB e *Fast*.

Algoritmo	**Iterações**	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
BSCB	45000	839.861163 s	12.6146	11.9710	12.3169
Fast	1472	123.5690 s	7.3996	7.5348	8.1333

Tabela 4.31: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	64.426100 s	46.8970	50.0044	29.5235
8	199.141587 s	329.2507	364.1865	145.4481
8(*)	255.862650 s	276.1546	314.8604	169.8391

Tabela 4.32: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	2.088992 s	7.3662	7.2928	7.8496
8	2.053344 s	7.9842	7.9487	8.2460
21	2.453671 s	9.1422	8.9647	9.1054

Tabela 4.33: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	839.861163 s	0.9797
Fast	123.5690 s	0.9808
SPIR(4 NNP)	64.426100 s	0.9716
G-SPIR(k=4)	2.088992 s	0.9808

Restauração em imagem com textura. Imagem corrompida artificialmente. Seu tamanho é 275×183 e $\#\Omega = 5126$.

(a) (b) (c)

Figura 4.36: (a) Imagem original; (b) Imagem corrompida; (c) máscara de retoque.

Todos os algoritmos retocaram a imagem satisfatoriamente com resultados muito próximos quanto à recuperação da textura.

(a) (b)

Figura 4.37: (a) Algoritmo BSCB; (b) Algoritmo *Fast*.

(a) (b) (c)

Figura 4.38: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.

(a) (b) (c)

Figura 4.39: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.34: Desempenho dos algoritmos BSCB e *Fast*.

Algoritmo	**Iterações**	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
BSCB	45000	450.534370 s	122.1317	121.1348	110.4484
Fast	560	7.738811 s	101.3142	99.5933	94.8378

Tabela 4.35: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	99.903306 s	139.7116	134.7855	122.0875
8	311.836496 s	193.3634	175.2189	149.2705
8(*)	852.177479 s	199.3054	187.5473	160.5186

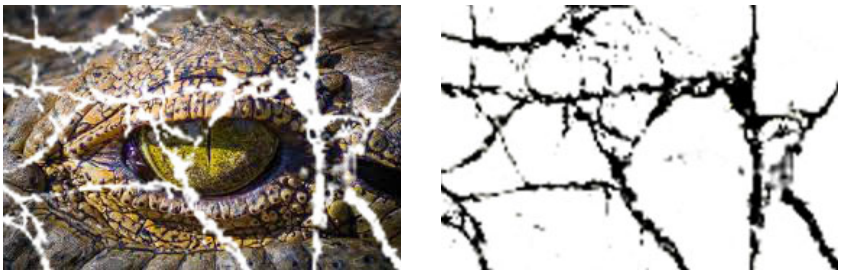
Tabela 4.36: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	1.532321 s	101.5443	99.1971	95.3361
8	1.636019 s	112.4862	110.2967	104.0975
21	2.214519 s	132.8067	129.4126	116.4264

Tabela 4.37: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	450.5343703 s	0.9511
Fast	7.738811 s	0.9555
SPIR(4 NNP)	99.903306 s	0.9016
G-SPIR(k=4)	1.532321 s	0.9563

Restauração de imagem com textura com aumento da área do domínio de retoque. Usamos a imagem anterior, mas com o domínio de retoque abrangendo uma área maior da imagem, com $\#\Omega = 7341$.



(a) (b)

Figura 4.40: (a) Imagem corrompida; (b) máscara de retoque.

Com o aumento da área, é possível perceber pequenas falhas na recuperação da textura no domínio de retoque. O algoritmo G-SPIR apresentou o melhor resultado visual.

(a) (b)

Figura 4.41: (a) Algoritmo BSCB; (b) Algoritmo *Fast*.

(a) (b) (c)

Figura 4.42: SPIR com: (a) 4 NNP, (b) 8 NNP, (c) 8 NNP e busca randômica de vizinhos.

(a) (b) (c)

Figura 4.43: G-SPIR com: (a) $k = 4$, (b) $k = 8$, (c) $k = 21$.

Tabela 4.38: Desempenho dos algoritmos BSCB e *Fast*.

Algoritmo	**Iterações**	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
BSCB	45000	619.335425 s	235.3428	222.3151	181.9130
Fast	1833	25.031135 s	193.1616	186.6176	158.5194

Tabela 4.39: Desempenho do algoritmo SPIR quanto ao número de vizinhos NNP.

NNP	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	111.434120 s	228.7571	219.3761	184.3899
8	358.338436 s	436.9963	367.4142	291.2923
8(*)	1611.694268 s	332.2179	304.9182	254.5190

Tabela 4.40: Desempenho do algoritmo G-SPIR quanto ao número k de vizinhos mais próximos.

k	**Tempo decorrido**	**(MSE)r**	**(MSE)g**	**(MSE)b**
4	2.001107 s	221.3452	213.2479	179.1630
8	2.029226 s	224.2339	213.9017	177.1285
21	2.725910 s	233.3086	219.8339	177.1774

Tabela 4.41: Desempenho dos algoritmos quanto ao SSIM.

Algoritmo	**Tempo decorrido**	**SSIM**
BSCB	619.335425 s	0.9145
Fast	25.031135 s	0.9215
SPIR(4 NNP)	111.434120 s	0.9123
G-SPIR(k=4)	2.001107 s	0.9195

4.4 Teste 4: Remoção de objetos em imagens reais

Neste teste, verificamos a eficiência dos algoritmos na remoção de um objeto presente em uma imagem real. O objetivo é avaliar se a imagem retocada apresenta um aspecto natural, de tal forma que a alteração na mesma seja imperceptível aos olhos de um observador que não esteja familiarizado com a imagem original. No que concerne aos algoritmos SPIR e G-SPIR, mostramos somente o melhor resultado de cada um, ambos obtidos com NNP= $k = 4$.

Remoção de objeto em imagens com fundo sem textura. O tamanhos da imagens abaixo é 400×562 e $\#\Omega = 3234$.

(a) (b)

Figura 4.44: (a) Imagem original, (b) Máscara.

A remoção do caule com o pássaro foi bem sucedida com todos os algoritmos, observando-se novamente um defeito na fronteira do domínio de retoque na imagem obtida com o algoritmo SPIR. O mesmo defeito está presente de forma mais branda na imagem retocada pelo algoritmo G-SPIR.

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

Figura 4.45: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo SPIR; (d) Algoritmo G-SPIR.

Tabela 4.42: Desempenho dos algoritmos.

Algoritmo	**Tempo decorrido**
BSCB	2356.772733 s
Fast	190.358850 s
SPIR	42.053284 s
G-SPIR	4.347986 s

A próxima imagem tem tamanho 500×336 e $\#\Omega = 6158$.

(a) (b)

Figura 4.46: (a) Imagem original, (b) Máscara de retoque.

Os algoritmos *Fast* e G-SPIR removeram o objeto com sucesso, apesar de o domínio de retoque ocupar uma área maior da imagem. Os algoritmos de BSCB e SPIR não fizeram o transporte adequado das informações da fronteira para o domínio, produzindo defeitos na imagem.

(a) (b)

(c) (d)

Figura 4.47: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo SPIR; (d) Algoritmo G-SPIR.

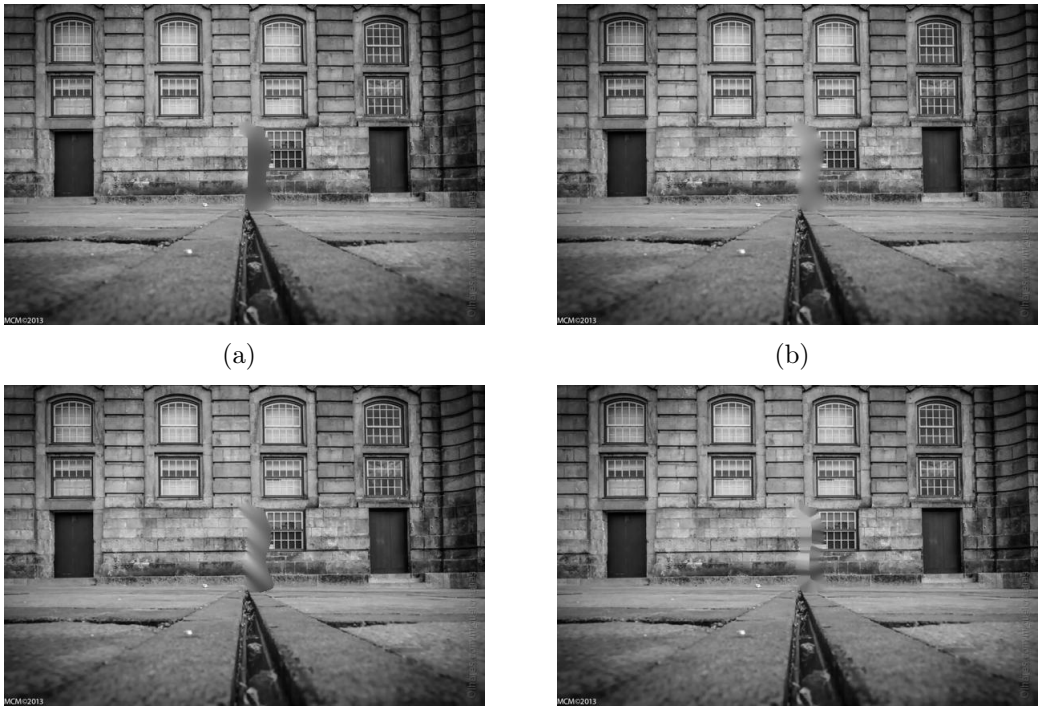
Tabela 4.43: Desempenho dos algoritmos.

Algoritmo	**Tempo decorrido**
BSCB	2075.860330 s
Fast	339.896076 s
SPIR	84.323751 s
G-SPIR	3.434111 s

Remoção de objeto sobre fundo com textura. O objeto (ancião) também ocupa uma grande área da imagem. O tamanho da imagem é 600×400 e $\#\Omega = 3198$.

(a) (b)

Figura 4.48: (a) Imagem original, (b) Máscara de retoque.



(a) (b)

(c) (d)

Figura 4.49: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo SPIR; (d) Algoritmo G-SPIR.

Nenhum dos algoritmos foi capaz de reproduzir a textura no interior do domínio de retoque. Todos os algoritmos também foram malsucedidos na remoção de objetos em imagens de alta resolução (Figuras 4.50 a 4.52).

Tabela 4.44: Desempenho dos algoritmos quanto aos resultados da figura 4.49.

Algoritmo	**Tempo decorrido**
BSCB	1017.201598 s
Fast	170.035945 s
SPIR	11.534777 s
G-SPIR	1.469353 s

(a)

(b)

Figura 4.50: (a) Imagem original; (b) Máscara.

(a)

(b)

(c)

Figura 4.51: Imagem retocada por: (a) Algoritmo BSCB; (b) Algoritmo *Fast*; (c) Algoritmo SPIR.

Figura 4.52: Retoque com o algoritmo G-SPIR.

Tabela 4.45: Desempenho dos algoritmos .

Algoritmo	**Tempo decorrido**
BSCB	16045.053277 s
Fast	4012.081417 s
SPIR	72.312143 s
G-SPIR	34.955103 s

4.5 Considerações finais

Alguns parâmetros, como o limiar ϵ e taxa de aprimoramento Δt presentes nos algoritmos BSCB e NS, ou o parâmetro κ na difusão anisotrópica, influenciam indiretamente na qualidade do resultado final. O ajuste destes parâmetros e a combinação ideal dos mesmos é uma questão subjetiva, podendo consumir bastante tempo de trabalho.

O retoque com o algoritmo NS se mostrou uma tarefa trabalhosa. Como o algoritmo só opera sobre domínios retangulares, ao efetuarmos o retoque em um domínio de geometria irregular, nos confrontamos com dois problemas. Optar por um retângulo $\bar{\Omega}$ que contenha totalmente o domínio de retoque Ω pode conduzir a sistemas de equações demasiadamente grandes durante a execução do algoritmo, provocando o estouro da memória. Ou ainda, as informações da fronteira de $\bar{\Omega}$ podem diferir muito das presentes na fronteira de Ω produzindo resultados indesejáveis. Por outro lado, executar o retoque em etapas intermediárias, usando retângulos $\bar{\Omega}$ menores, mas que se aproximem mais da fronteira de Ω, torna-se um trabalho quase artesanal, indo de encontro ao princípio do retoque digital que prega a intervenção mínima do usuário.

Apesar de o algoritmo SPIR ter sido idealizado para o problema de remoção de ruídos (Figura 4.53), os resultados obtidos com sua aplicação nos testes servem para enfatizar a importância da abordagem por agrupamento (usada na formulação do algoritmo G-SPIR)

na etapa de discretização da representação integral do SPH, quando este é aplicado ao problema do retoque digital.

(a) (b)

Figura 4.53: Remoção de ruído com o algoritmo SPIR.

Obtivemos resultados satisfatórios com o algoritmo G-SPIR, que comprovam sua eficiência computacional, e cuja qualidade visual se aproxima das obtidas com os algoritmos BSCB e *Fast*. O uso de barreiras de difusão também melhorou a qualidade do retoque. Assim como nos outros algoritmos testados, o tamanho da região retocada influenciou diretamente o resultado final. Também constatamos que pequenas alterações na máscara de retoque podem conduzir a resultados discrepantes (Figura 4.54).

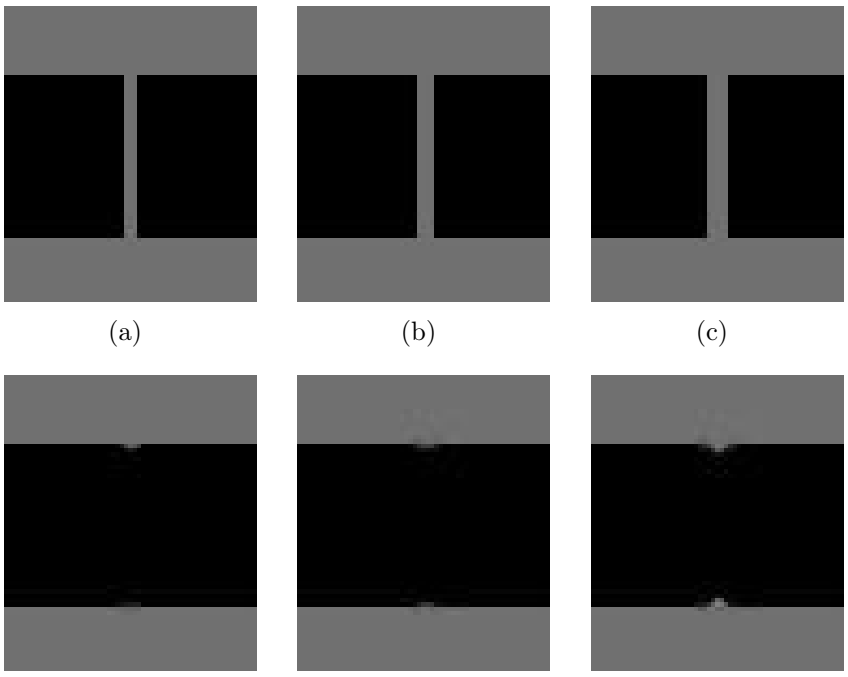
(a) (b)

(c) (d)

Figura 4.54: Influência das variações no traçado da máscara sobre o resultado final.

Em alguns dos testes feitos com o algoritmo G-SPIR, detectamos a ocorrência de um

defeito (Figura 4.18) na conexão de bordas que se agrava a medida que a distância entre estas aumenta (Figura 4.55).



(a) (b) (c)

(d) (e) (f)

Figura 4.55: Conexão de bordas malsucedida. O defeito se torna mais evidente à medida que aumenta a distância entre as partes a serem conectadas. Em (a), (b) e (c), as distâncias correspondem a 3, 4 e 5 *pixels*, respectivamente. Em (d), (e) e (f), os respectivos retoques com o algoritmo G-SPIR.

A incapacidade de transportar informações de textura para o domínio de retoque é um problema inerente aos métodos de retoque estrutural. Neste sentido, os resultados obtidos com o algoritmo G-SPIR também foram insatisfatórios (Figura 4.56).

(a) (b)

Figura 4.56: O G-SPIR não reproduz a textura da vizinhança.

Além das considerações feitas acima, é importante observar que em nenhum dos trabalhos citados no capítulo 2 foi disponibilizado o código original do algoritmo. Todas as implementações usadas aqui são interpretações pessoais baseadas nas descrições presentes em cada texto e podem, portanto, ter interferido no desempenho dos algoritmos.

Capítulo 5

Conclusões

A qualidade visual do retoque é, sem dúvida, o fator preponderante na escolha do método. Porém, em algumas aplicações, o tempo de execução de um método pode ser tão grande que torna a sua aplicação inviável do ponto de vista prático.

Embora o algoritmo BSCB tenha apresentado resultados satisfatórios em várias aplicações, o tempo decorrido foi grande quando comparado ao desempenho do algoritmo *Fast*. Por outro lado, embora o algoritmo *Fast* realize a restauração da imagem em um tempo consideravelmente menor, a própria natureza do método tende a comprometer a qualidade do retoque levando informações de cor de um lado para outro das linhas de contorno (isófotos), "poluindo" o resultado final. Apesar deste problema ser minimizado, ou até mesmo eliminado em alguns casos, com a aplicação das barreiras de difusão, pode ser trabalhoso determinar onde utilizar tais barreiras, dependendo da complexidade do domínio de retoque.

Neste trabalho propusemos o algoritmo de retoque digital G-SPIR, com uma simples formulação, que alia a eficiência na coleta da informação da área circundante com a versatilidade do método SPH no transporte dessas informações para o domínio de retoque. Neste ponto, nosso algoritmo difere do algoritmo de *denoising* SPIR, uma vez que adotamos a abordagem por agrupamento na etapa de discretização da representação integral do método SPH. Esta escolha permitiu a recuperação de áreas com um grande número de *pixels* adjacentes (sempre presentes no problema do retoque digital), tarefa que se mostrou inviável com a utilização do algoritmo SPIR, como pudemos constatar no capítulo 4. Além disso, a combinação do SPH com a abordagem por agrupamento resultou em satisfatória eficiência computacional e resultados visuais próximos dos obtidos com os algoritmos já consolidados. Por outro lado, como os outros algoritmos apresentados, o G-SPIR não foi capaz de transportar a textura da vizinhança para o domínio de retoque.

Para estudos futuros, pretendemos analisar a viabilidade do uso de outros operadores SPH para o aprimoramento dos resultados, principalmente no aspecto da conexão de

linhas de contorno em domínios de retoque que cubram grandes áreas da imagem.

Referências Bibliográficas

[AT01] Wilson Au and Ryo Takei. Image inpainting with the navier-stokes equations. *Final Report, APMA*, 930, 2001.

[Bai11] Ana Bailão. As técnicas de reintegração cromática na pintura: revisão historiográfica. *Ge-conservación*, pages 45–65, 2011.

[BBS01] Marcelo Bertalmio, Andrea L Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–355. IEEE, 2001.

[BSCB00] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[BVSO03] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE transactions on image processing*, 12(8):882–889, 2003.

[CB10] Wallace Correa de Oliveira Casaca and Maurílio Boaventura. A decomposition and noise removal method combining diffusion equation and wave atoms for textured images. *Mathematical Problems in Engineering*, 2010, 2010.

[CPT03] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–721. IEEE, 2003.

[CPT04] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[CS01a] Tony F Chan and Jianhong Shen. Morphologically invariant pde inpaintings. *Citeseer*, 2001.

[CS01b] Tony F Chan and Jianhong Shen. Non-texture inpainting by curvature-driven diffusions (cdd). *Journal of visual communication and image representation*, 12(4):436, 2001.

[CS01c] Tony F Chan and Jianhong Shen. Nontexture inpainting by curvature-driven diffusions. *Journal of Visual Communication and Image Representation*, 12(4):436–449, 2001.

[DBFTT11] G Di Blasi, E Francomano, A Tortorici, and E Toscano. A smoothed particle image reconstruction method. *Calcolo: a quarterly on numerical analysis and theory of computation*, 48(1):61–74, 2011.

[DCOY03] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. Fragment-based image completion. In *ACM Transactions on graphics (TOG)*, pages 303–312. ACM, 2003.

[EL99] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[Gon09] Rafael C Gonzalez. *Digital image processing*. Pearson Education India, 2009.

[GV97] Jonas Gomes and Luiz Velho. *Image processing for computer graphics*. Springer Science & Business Media, 1997.

[HT96] Anil N Hirani and Takashi Totsuka. Combining frequency and spatial domain information for fast interactive image noise removal. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 269–276. ACM, 1996.

[HVV01] SIMON S HAYKIN and Barry Van Veen. *Sinais e sistemas*. Bookman, 2001.

[Kan79] Gaetano Kanizsa. *Organization in vision: Essays on Gestalt perception*. Praeger Publishers, 1979.

[Koe84] Jan J Koenderink. The structure of images. *Biological cybernetics*, 50(5):363–370, 1984.

[LL03] Gui-Rong Liu and Moubin B Liu. *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific, 2003.

[MM98] Simon Masnou and Jean-Michel Morel. Level lines based disocclusion. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pages 259–263. IEEE, 1998.

[MO00] Antonio Marquina and Stanley Osher. Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. *SIAM Journal on Scientific Computing*, 22(2):387, 2000.

[NMS93] Mark Nitzberg, David Mumford, and Takahiro Shiota. *Filtering, segmentation, and depth*. Springer-Verlag New York, Inc., 1993.

[OBMC01] Manuel M Oliveira, Brian Bowen, Richard McKenna, and Yu-Sung Chang. Fast digital image inpainting. In *Appeared in the Proceedings of the International Conference on Visualization, Imaging and Image Processing (VIIP 2001), Marbella, Spain*, pages 106–107, 2001.

[OS88] Stanley Osher and James A Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[PM90] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[PPLT09] Afonso Paiva, Fabiano Petronetto, Thomas Lewiner, and Geovan Tavares. *Simulação de Fluidos sem Malha: Uma Introdução ao Método SPH*. IMPA, Rio de Janeiro, 2009. 27° Colóquio Brasileiro de Matemática.

[RO94] Leonid I Rudin and Stanley Osher. Total variation based image restoration with free local constraints. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 1, pages 31–35. IEEE, 1994.

[ROF92] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[SC01] Jianhong Shen and Tony Chan. Variational restoration of nonflat image features: Models and algorithms. *SIAM Journal on Applied Mathematics*, 61(4):1338–1361, 2001.

[SC02] Jianhong Shen and Tony F Chan. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002.

[ZZSZ13] Wangmeng Zuo, Lei Zhang, Chunwei Song, and David Zhang. Texture enhanced image denoising via gradient histogram preservation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1203–1210, 2013.