O método do gradiente espectral
projetado aplicado ao problema de
Image Inpainting usando
regularização ℓ_1 .

Anderson Conceição de Almeida

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Programa: Matemática Aplicada
Orientador: Prof. Dr. Ernesto G. Birgin

Durante o desenvolvimento deste trabalho, o autor recebeu auxílio financeiro da CAPES.

São Paulo, 14 de novembro de 2015

O método do gradiente espectral projetado aplicado ao problema de *Image Inpainting* usando regularização ℓ_1 .

Esta é a versão original da dissertação/tese elaborada pelo candidato Anderson Conceição de Almeida, tal como submetida à Comissão Julgadora.

Agradecimentos

Agradeço a minha mãe Edite, pelo seu infindável amor, compreensão e companheirismo com que sempre me acompanhou desde meu nascimento.

Agradeço ao meu pai Adolfo (*in memoriam*), que mesmo não estando presente desde minha infância, deixou plantadas condições para que eu pudesse me desenvolver como homem e ser humano.

Agradeço ao meu orientador, Ernesto G. Birgin pelo exemplo de pesquisador, professor, caráter, e por que não, amizade.

Agradeço a minha família: minha avó Arlinda, meus tios Roberto, Zé Carlos, Cosme, Gerson, minhas tias Elizete, Evanete, Ezenite, Cida, Ednesia pelo aconchego e pela alegria com que sempre me receberam. Sem vocês tudo teria sido muito mais difícil.

Aos colegas do grupo de otimização matemática computacional entre eles Oberlan Romão, Rafael Lobato, John Gardenghi pelos almoços e conversas no laboratório regados a risadas e lições sobre programação. Assim como todos os colegas do laboratório 118.

Ao meu amigo de longa data Rafael Nantes, pelas cervejas acompanhadas de bons diálogos sobre a vida, ciência e por ter aberto a sua casa sempre que precisei.

Ao amigo Cristian Vasconcellos, por sempre conseguir aliar filosofia e amizade, independentemente das nossas visões de mundo.

Aos amigos Leandro Junqueira e Alana Reis por todas as agradabilíssimas conversas, visitas e cafés.

Aos colegas da graduação Gabriel Fernandes, Glaucio Fabiano, Diolan Godinho, João Paulo Mello por todo o crescimento intelectual e ajuda mútua. Agradeço também a todos os amigos da pós-graduação do IME- USP, entre eles, Valdir Filho, Belmiro Galo, André Zaidan, Inocência Zimba, Flávia Teles, Bruna Cassol.

A todos os meus familiares e amigos que de uma forma ou de outra me ajudaram nesse trajeto, muitas vezes não diretamente, mas sempre de maneira significativa.

Resumo

ALMEIDA, A. C. de **O método do gradiente espectral projetado aplicado ao problema de *Image Inpainting* usando regularização ℓ_1 .** 2015. 55 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2010.

O problema de reconstrução digital de imagens (*Image Inpainting*) possui diversas abordagens para sua resolução. Uma possibilidade consiste na sua modelagem como um problema de otimização contínua. Na presente dissertação aplica-se o método do gradiente espectral projetado a esse problema. Desenvolve-se inteiramente a modelagem do problema assim como a implementação computacional do método de otimização que o resolve. Resultados computacionais demonstram a qualidade do método para um conjunto de imagens digitais.

Palavras-chave: Reconstrução digital de Imagens, Otimização Matemática, Gradiente Espectral Projetado, Regularização ℓ_1 .

Abstract

ALMEIDA, A. C. de **The spectral gradient method applied to the Image Inpainting problem using ℓ_1 -Regularization.** 2015. 55 f. Dissertation (Master Degree) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2010.

The image inpainting problem has several resolution approaches. One possibility consists in its modeling as a continuous optimization problem. In the present dissertation we apply the spectral projected gradient method to this problem. We develop the whole modeling of the problem as well as the computational implementation of the optimization method to solve it. Computational results show the quality of the method for a set of digital images.

Keywords: Image Inpainting, Mathematical Optimization, Spectral Projected Gradient, ℓ_1 - Regularization.

Sumário

1	Introdução	1
2	Otimização e Image Inpainting	5
2.1	O Método proposto	5
2.1.1	Caso com um número qualquer de coberturas	11
2.2	O método do gradiente espectral projetado	12
2.3	Implementação da função objetivo e do gradiente da função objetivo	14
3	Experimentos computacionais	23
3.1	A escolha do dicionário	24
3.2	Relação de n_g e $psnr$	26
3.3	Relação de t e $psnr$	31
3.4	Relação de λ e $psnr$	39
3.5	Recuperando a imagem “Barbara” com parâmetros fixados <i>a priori</i>	43
4	Conclusões e trabalhos futuros	49
	Referências Bibliográficas	51

Capítulo 1

Introdução

A imagem está associada ao próprio desenvolvimento da linguagem humana, e consequentemente ao próprio desenvolvimento da civilização. Com o desenvolvimento da tecnologia e da arte, a imagem se transforma cada vez mais em técnica. Desde a sua representação computacional ao tratamento dos mais diversos problemas em processamento de imagens (ou em geral, processamento de sinais), encontramos desafios nas mais variadas áreas, entre elas: engenharia, estatística, matemática e computação; ver por exemplo [AK06, EDE96, Yar13, YY85]. A presente dissertação tem como foco o tratamento matemático e computacional de um desses problemas, o problema de retoque digital (*Image Inpainting*). O problema consiste em reconstruir uma região pré-definida de uma imagem de maneira que a imagem final seja a mais “real” possível, ou em outras palavras, que um observador casual não perceba a manipulação da imagem.

Técnicas manuais de recuperação de imagens são antigas e tem suas origens na recuperação de capelas, quadros, entre outros [Wal85, EM76]. Já do ponto de vista computacional, os primeiros trabalhos para tratar esse tipo de problema surgiram por volta do final do século XX [KMFR95, HT96, EL⁺99, HB95, SP98, NMS93, MM98]. Desde então, diversas formas de abordagem à reconstrução de imagens digitais foram desenvolvidas. Um panorama detalhado dos diferentes enfoques em *image inpainting*, avanços recentes e links para códigos computacionais podem ser encontrados em [GLM14].

Baseando-se nas características principais dos métodos computacionais usados para tratar o problema de *Image Inpainting*, é comum, na literatura de processamento de imagens, classificá-los em alguns paradigmas principais. Entre eles destacam-se: reconstrução baseada em sistemas dinâmicos e equações diferenciais parciais [BSCB00, BBS01], reconstrução baseada em ampliação de textura (*Texture Synthesis*) [EL⁺99, WL00] e reconstrução baseada em exemplares (Exemplar-based inpainting methods); ver por exemplo [CPT04, LLX⁺01]. O primeiro deles usa a teoria de difusão do calor para propagar estruturas externas à região a ser recuperada para dentro da mesma. Esse tipo de abordagem foi primeiramente apresentado em [BSCB00] e se inspira fortemente nos trabalhos desenvolvidos em [MM98]. É interessante notar que é também em [BSCB00] que pela primeira vez aparece a nomenclatura *Image Inpainting* para o problema de reconstrução de imagens digitais. Nesse trabalho, os autores interpretam uma imagem como uma chapa com temperaturas distintas em cada ponto, onde a intensidade de cada pixel representa a temperatura da chapa naquele ponto. O método de reconstrução de imagens então se reduz em grande parte

ao problema de equilíbrio térmico proveniente da Física. Linhas com mesma intensidade de pixel que chegam na borda da região a ser recuperada são estendidas iterativamente, usando o gradiente da imagem no ponto da borda para a correta direção de propagação dessas linhas. Após esse processo, é usada a técnica de difusão anisotrópica com o objetivo de garantir a convergência do método e harmonizar as linhas propagadas, assim como estender grandes regiões da imagem para o interior da região danificada. Tal método, apesar de inovador, apresenta deficiências significativas quando trata de replicar texturas na região danificada e também quando essa mesma região tem uma extensão grande em relação à imagem. Uma das grandes vantagens, porém, é o fato de não precisar qualquer intervenção de um usuário, exceto na determinação da região a ser restaurada.

A segunda classe de métodos é na verdade um campo de estudos à parte em processamento digital de imagens, mas que se aplica perfeitamente ao problema de *Image Inpainting*. *Texture Synthesis* refere-se ao problema de reproduzir uma textura em grandes regiões, baseando-se apenas numa pequena amostra daquela textura. Tal método, quando aplicado à recuperação digital de imagens, funciona muito bem quando a região danificada não possui detalhes muito específicos a serem recuperados, como cantos e linhas que caracterizam a imagem. Esse método foi apresentado nos trabalhos [EL⁺99, WL00] e vários métodos híbridos o usam como um complemento na recuperação de grandes regiões.

Por último, a reconstrução baseada em exemplares tem como ideia central a cópia de “retalhos” (*patch*) da área externa à região desconhecida para dentro da mesma. A escolha certa do conjunto no qual é feita a busca por tais retalhos é de fundamental importância para o sucesso de métodos baseados em exemplares. Esse conjunto, também chamado de “dicionário”, pode ser formado de diversas maneiras, tanto com partes da região intacta, como com partes de imagens pertencentes a um banco de imagens pré definido. Uma alternativa que tem sido bastante explorada na construção dos retalhos mais adequados ao preenchimento da região danificada é a consideração de combinações lineares dos elementos do dicionário [Li11, Woh11, XS10]. Nesse caso o dicionário também pode ser complementado com conjuntos geradores clássicos como *wavelets*, *curvelets* [FSM09], entre outros. Tal abordagem visa conseguir uma riqueza de detalhes maior do que a opção de exemplares únicos.

O enfoque por combinações lineares leva naturalmente à busca de soluções esparsas, ou seja, soluções que carreguem apenas as informações imprescindíveis da parte intacta da imagem. O problema de encontrar soluções esparsas para problemas desse tipo tem sido chamado de *basis pursuit denoising* na literatura e tem sido alvo de diversos estudos nos últimos anos devido, principalmente, ao surgimento de uma vasta gama de aplicações práticas. Entre elas, podemos citar *compressive sensing* [Don06, CRT06, Lla12], máquinas de suporte vetorial [Gir98, BDE09, SS04], problema de *PageRank* do Google [LM11], *machine learning* [FHT01]. Entre os métodos numéricos de otimização mais usuais para sua resolução destacam-se métodos baseados em continuação homotópica [MCW05], gradientes projetados [FNW07] e o algoritmo *in-crowd* [GWM11]. Uma análise detalhada sobre problemas de *basis pursuit denoising* pode ser encontrada em [VDBF08].

Nessa dissertação, desenvolvemos um método para *image inpainting* baseado em combinação linear esparsa de exemplares exposto em [Woh11]. Esse método usa regularização- ℓ_1 para o problema de *basis pursuit denoising*, focando na otimização de uma única função objetivo que visa capturar as propriedades globais da região conhecida da imagem. Resolveremos o problema de otimização

pelo método de gradiente projetado; ver [BMR09].

O trabalho se organiza da seguinte maneira. No Capítulo 2, apresentamos primeiramente a modelagem do problema para um caso particular e generalizamos sua abordagem para o caso geral (Seção 2.1 e Subseção 2.1.1). Em seguida, descrevemos o método de otimização numérica usado no problema de otimização proveniente do modelo de *Image Inpainting* (Seção 2.2). Na Seção 2.3, apresentamos pseudo-códigos dos algoritmos usados na implementação do programa computacional. No Capítulo 3, são expostos experimentos computacionais, com imagens, tabelas e gráficos mostrando a influência dos diversos parâmetros na qualidade das imagens recuperadas pelo método proposto (Seções 3.1 a 3.4). Na Seção 3.5, recuperamos uma imagem nova, com os parâmetros fixados *a priori* baseando-se nos experimentos realizados. As conclusões e possíveis vias de continuação do trabalho são discutidas no último capítulo.

Capítulo 2

Otimização e Image Inpainting

Nesse capítulo, descreveremos o método que motivou a elaboração da presente dissertação. Seguiremos de perto o artigo [Woh11]. Este método consiste em modelar o problema de *Image Inpainting* como um problema de otimização não linear, com restrições de caixa. Tal problema de otimização tem demonstrado bons resultados computacionais ao ser resolvido por métodos do tipo gradiente espectral (ver [FNW07]). Descreveremos também nessa seção um desses métodos: o método do gradiente espectral projetado, desenvolvido no artigo [BMR00] em 2000. Na Seção 2.1, começamos com uma descrição formal para o nosso modelo. Algumas definições importantes são apresentadas a seguir.

Definição 1 *Sejam $nl, nc \in \mathbb{N}$, chamamos “imagem” de dimensão $nl \times nc$ uma função I definida em um subconjunto $S \subset Z = \{1, \dots, nl\} \times \{1, \dots, nc\}$ tomando valores inteiros em $[0, 255]$. Os elementos de Z são chamados *pixels* e $I(i, j)$ é o valor do pixel (i, j) , para $(i, j) \in S$.*

Definição 2 *Seja $A \subset Z$, uma “cobertura” de A é um conjunto $P \subset \mathcal{P}(Z)$, com $P = \{B_1, B_2, \dots, B_k\}$, tal que os B_i são conjuntos de pixels $B_i \cap B_j = \emptyset$ se $i \neq j$, $B_i \cap A \neq \emptyset \quad \forall i \in \{1, \dots, k\}$ e $A \subset \cup P$.*

Definição 3 *Se ϕ e ξ são funções reais definidas em um conjunto A e $C = \{x_1, x_2, x_3, \dots, x_n\} \subset A$, a distância entre ϕ e ξ , relativa a C , é definida por $d|_C(\phi, \xi) = \sum_{i=1}^n (\phi(x_i) - \xi(x_i))^2$.*

2.1 O Método proposto

Dado $\Omega \subset Z$ e uma imagem I definida em $Z \setminus \Omega$, chamamos Ω de “região a ser restaurada” e $Z \setminus \Omega$ de “região intacta”. O problema de *image inpainting* consiste em definir valores para função I nos pixels de $\Omega \subset Z$ de maneira que não haja uma discrepância muito grande em relação ao complementar de Ω em Z . O significado de “discrepância muito grande” será determinado em seguida.

Para realizarmos tal tarefa, tomaremos diversas coberturas para a região desconhecida Ω e em seguida, atribuiremos valores aos elementos dessas coberturas pedindo que esses valores se distanciem o mínimo possível, tanto entre si (quando houver intersecção entre os conjuntos que formam as coberturas) quanto em relação aos elementos intactos da imagem.

Em outras palavras, seja $I : Z \setminus \Omega \rightarrow [0, 255] \cap \mathbb{Z}$ uma imagem e $\Omega \subset Z$ um conjunto de pixels danificados na imagem. Sejam ainda $P = \{B_i\}_{i \in \{1, \dots, \ell\}}$ e $P' = \{B'_i\}_{i \in \{1, \dots, \ell'\}}$ coberturas de Ω , nosso objetivo é definir funções $f_i : B_i \rightarrow [0, 255] \cap \mathbb{Z}$, $i = 1, \dots, \ell$ e $f'_i : B'_i \rightarrow [0, 255] \cap \mathbb{Z}$, $i = 1, \dots, \ell'$ de maneira que

$$\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} d|_{B_i \cap B'_j \cap \Omega}(f_i, f'_j) + \sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B_i \cap (Z \setminus \Omega)}(f_i, I) + \sum_{\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B'_i \cap (Z \setminus \Omega)}(f'_i, I) \quad (2.1)$$

seja mínima.

Aqui, é importante salientar que essa função pode ser mal escalada. Isso se deve ao fato de que a quantidade de termos na somatória pode ser tão grande quanto se queira. Com isso em mente, é natural que insiramos constantes divisoras. Temos assim, nossa nova função escalada, dada por:

$$\frac{1}{N_1} \left(\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} d|_{B_i \cap B'_j \cap \Omega}(f_i, f'_j) \right) + \frac{1}{N_2} \left(\sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B_i \cap (Z \setminus \Omega)}(f_i, I) \right) + \frac{1}{N_2} \left(\sum_{\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B'_i \cap (Z \setminus \Omega)}(f'_i, I) \right), \quad (2.2)$$

onde N_1 representa a quantidade de termos em

$$\left(\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} d|_{B_i \cap B'_j \cap \Omega}(f_i, f'_j) \right)$$

e N_2 é a quantidade de termos na somatória

$$\left(\sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B_i \cap (Z \setminus \Omega)}(f_i, I) + \sum_{\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B'_i \cap (Z \setminus \Omega)}(f'_i, I) \right).$$

Seja agora $z \in \Omega$, definido o valor de $f_i(z)$ e $f'_j(z)$, queremos definir o valor da função I em z . Uma possível alternativa para definir $I(z)$, será a média aritmética dos valores $f_i(z)$ e $f'_j(z)$ determinados pelo problema de minimização apresentado anteriormente. Veremos que, do ponto de vista computacional, essa ainda não é uma boa opção. Note que poderíamos fazer o mesmo se tivéssemos um número qualquer de coberturas de Ω .

É preciso agora fixar espaços de funções nos quais gostaríamos de fazer a busca por funções f que minimizem (2.1). É razoável que o conjunto escolhido carregue alguma informação de $Z \setminus \Omega$. Sendo assim, nada mais natural do que pegar “pedaços” da imagem intacta para definir bases para nossos espaços de busca. Ou seja, para cada B_i pertencente à cobertura P de Ω e $f_i : B_i \rightarrow [0, 255] \cap \mathbb{Z}$ procuradas, gostaríamos de escolher conjuntos

$$D_i^k \subset Z \setminus \Omega, \quad k = 1, \dots, m_i,$$

onde $m_i \in \mathbb{N}$ e considerar os espaços vetoriais

$$\text{span}(\{I|_{D_i^k}\}_{k=1 \dots m_i}),$$

determinando, assim, espaços nos quais serão buscadas as funções f_i que minimizem nosso problema¹. A partir de agora usaremos a notação $I|_{D_k^i} = \phi_i^k$, e portanto $f_i = \sum_{k=1}^{m_i} \alpha_i^k \phi_i^k = \langle \alpha_i, \bar{\phi}_i \rangle$, onde $\alpha_i = (\alpha_i^1, \dots, \alpha_i^{m_i})^T$, e $\bar{\phi}_i = (\phi_i^1, \dots, \phi_i^{m_i})^T$.

Dessa forma, a procura por cada f_i , ($i = 1, \dots, \ell$) transforma-se na procura pelos coeficientes $\alpha_i^1, \dots, \alpha_i^{m_i}$, e o mesmo vale para f'_i , $i = 1, \dots, \ell'$.

Definição 4 *O conjunto de todos os elementos ϕ_i^k , $k = 1, \dots, m_i$, $i = 1, \dots, \ell$, é chamado comumente de “Dicionário”.*

Teremos assim para as coberturas P, P' , o seguinte problema de minimização:

$$\min_{\alpha} \frac{1}{N_1} \left(\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} d|_{B_i \cap B'_j \cap \Omega} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k, \sum_{k=1}^{m'_j} \alpha'_j{}^k \phi'_j{}^k \right) \right) + \frac{1}{N_2} \left(\sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B_i \cap (Z \setminus \Omega)} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k, I \right) + \sum_{\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B'_i \cap (Z \setminus \Omega)} \left(\sum_{k=1}^{m'_i} \alpha'_i{}^k \phi'_i{}^k, I \right) \right) \quad (2.3)$$

onde $\alpha = (\vec{\alpha}_1, \vec{\alpha}_2, \dots, \vec{\alpha}_\ell, \vec{\alpha}'_1, \vec{\alpha}'_2, \dots, \vec{\alpha}'_{\ell'})$.

Apesar da expressão carregada, podemos reescrever o problema acima em uma forma mais compacta e informativa. Para tanto, utilizaremos algumas convenções: para os conjuntos que intersectam a parte danificada da imagem, ou seja, os conjuntos $\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}$, $\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}$, denotaremos por N_P e $N_{P'}$ suas cardinalidades. Indexaremos também seus elementos por $\{1, 2, \dots, N_P\}$ e $\{1, 2, \dots, N_{P'}\}$.

Formalmente, escrevemos:

$$N_P := \#\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}, \quad \{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\} =: \{i^1, \dots, i^{N_P}\},$$

$$N_{P'} := \#\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}, \quad \{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\} =: \{j^1, \dots, j^{N_{P'}}\},$$

Usaremos também notações semelhantes para o conjunto de índices que correspondem a elementos das duas partições que se intersectam. Isto é:

$$N_{P,P'} := \#\{(i,j)|B_i \cap B'_j \cap \Omega \neq \emptyset\}, \quad \{(i,j)|B_i \cap B'_j \cap \Omega \neq \emptyset\} =: \{(a^1, b^1), \dots, (a^{N_{P,P'}}, b^{N_{P,P'}})\},$$

Nossas próximas convenções referem-se à cardinalidade da intersecção dos B_i e B'_i com a parte intacta $(Z \setminus \Omega)$ e à indexação dessa intersecção por um subconjunto dos números naturais. A seguir, temos a escrita formal dessa descrição:

$$N_i := \#(B_i \cap (Z \setminus \Omega)), \quad B_i \cap (Z \setminus \Omega) =: \{x_i^1, \dots, x_i^{N_i}\}$$

$$N'_i := \#(B'_i \cap (Z \setminus \Omega)), \quad (B'_i \cap (Z \setminus \Omega)) =: \{x'^1_i, \dots, x'^{N'_i}_i\},$$

¹Note que isso cria um problema: As funções da base estarão todas definidas em domínios diferentes da f . Para resolver isso, basta escolher para os domínios das funções que formam a base, conjuntos que tenham uma correspondência biunívoca com o domínio da solução. Com um abuso de linguagem, assumiremos que essa correspondência já está estabelecida.

$$N_{i,j} := \#(B_i \cap B'_j \cap \Omega), \quad (B_i \cap B'_j \cap \Omega) =: \{x_{i,j}^1, \dots, x_{i,j}^{N_{i,j}}\}.$$

Tais convenções nos permitirão chegar na expressão matricial de (2.3). Com vista a esse objetivo, examinemos agora cada um dos termos de (2.3):

Por ora, esqueçamos a constante de escalamento N_1 , e vejamos o termo

$$\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} d|_{B_i \cap B'_j \cap \Omega} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k, \sum_{k=1}^{m'_j} \alpha_j'^k \phi_j'^k \right) \quad (2.4)$$

em (2.3), que pode ser reescrito como

$$\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} \left(\sum_{x \in B_i \cap B'_j \cap \Omega} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x) - \sum_{k=1}^{m'_j} \alpha_j'^k \phi_j'^k(x) \right)^2 \right).$$

Ou em formato matricial,

$$\begin{aligned} & \sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} \left\| \begin{pmatrix} \sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x_{i,j}^1) - \sum_{k=1}^{m'_j} \alpha_j'^k \phi_j'^k(x_{i,j}^1) \\ \sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x_{i,j}^2) - \sum_{k=1}^{m'_j} \alpha_j'^k \phi_j'^k(x_{i,j}^2) \\ \vdots \\ \sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x_{i,j}^{N_{i,j}}) - \sum_{k=1}^{m'_j} \alpha_j'^k \phi_j'^k(x_{i,j}^{N_{i,j}}) \end{pmatrix} \right\|_2^2 = \\ & \sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} \left\| \begin{pmatrix} \phi_i^1(x_{i,j}^1) & \phi_i^2(x_{i,j}^1) & \dots & \phi_i^{m_i}(x_{i,j}^1) \\ \phi_i^1(x_{i,j}^2) & \phi_i^2(x_{i,j}^2) & \dots & \phi_i^{m_i}(x_{i,j}^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_i^1(x_{i,j}^{N_{i,j}}) & \phi_i^2(x_{i,j}^{N_{i,j}}) & \dots & \phi_i^{m_i}(x_{i,j}^{N_{i,j}}) \end{pmatrix} \begin{pmatrix} \alpha_i^1 \\ \alpha_i^2 \\ \vdots \\ \alpha_i^{m_i} \end{pmatrix} - \right. \\ & \left. \begin{pmatrix} \phi_j'^1(x_{i,j}^1) & \phi_j'^2(x_{i,j}^1) & \dots & \phi_j'^{m'_j}(x_{i,j}^1) \\ \phi_j'^1(x_{i,j}^2) & \phi_j'^2(x_{i,j}^2) & \dots & \phi_j'^{m'_j}(x_{i,j}^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_j'^1(x_{i,j}^{N_{i,j}}) & \phi_j'^2(x_{i,j}^{N_{i,j}}) & \dots & \phi_j'^{m'_j}(x_{i,j}^{N_{i,j}}) \end{pmatrix} \begin{pmatrix} \alpha_j'^1 \\ \alpha_j'^2 \\ \vdots \\ \alpha_j'^{m'_j} \end{pmatrix} \right\|_2^2. \end{aligned}$$

De forma mais compacta, temos que a expressão acima pode ser escrita como

$$\sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} \left\| \begin{bmatrix} \phi_i(x_{i,j}) & -\phi_j'(x_{i,j}) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j' \end{bmatrix} \right\|_2^2, \quad (2.5)$$

onde

$$\phi_i(x_{i,j}) = \begin{pmatrix} \phi_i^1(x_{i,j}^1) & \phi_i^2(x_{i,j}^1) & \dots & \phi_i^{m_i}(x_{i,j}^1) \\ \phi_i^1(x_{i,j}^2) & \phi_i^2(x_{i,j}^2) & \dots & \phi_i^{m_i}(x_{i,j}^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_i^1(x_{i,j}^{N_{i,j}}) & \phi_i^2(x_{i,j}^{N_{i,j}}) & \dots & \phi_i^{m_i}(x_{i,j}^{N_{i,j}}) \end{pmatrix}, \quad \phi'_j(x_{i,j}) = \begin{pmatrix} \phi'_j{}^1(x_{i,j}^1) & \phi'_j{}^2(x_{i,j}^1) & \dots & \phi'_j{}^{m'_j}(x_{i,j}^1) \\ \phi'_j{}^1(x_{i,j}^2) & \phi'_j{}^2(x_{i,j}^2) & \dots & \phi'_j{}^{m'_j}(x_{i,j}^2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi'_j{}^1(x_{i,j}^{N_{i,j}}) & \phi'_j{}^2(x_{i,j}^{N_{i,j}}) & \dots & \phi'_j{}^{m'_j}(x_{i,j}^{N_{i,j}}) \end{pmatrix}$$

e, claramente,

$$\alpha_i = (\alpha_i^1, \alpha_i^2, \dots, \alpha_i^{m_i})^T, \quad \alpha'_j = (\alpha'_j{}^1, \alpha'_j{}^2, \dots, \alpha'_j{}^{m'_j})^T.$$

Recuperando a constante N_1 , poderemos escrever (2.5) da seguinte maneira,

$$\frac{1}{N_1} \sum_{\{i,j|B_i \cap B'_j \cap \Omega \neq \emptyset\}} \left\| [\phi_i(x_{i,j}) \quad -\phi'_j(x_{i,j})] \begin{bmatrix} \alpha_i \\ \alpha'_j \end{bmatrix} \right\|_2^2 = \frac{1}{N_1} \left\| \begin{bmatrix} H & -U \end{bmatrix} \alpha \right\|_2^2,$$

onde

$$H_{n,k} = \begin{cases} \phi_k(x_{i,j}), & \text{se } k = a^n, \text{ para } n = 1, \dots, N_{P,P'} \text{ e } k = 1, \dots, \ell. \\ 0, & \text{c.c.} \end{cases},$$

$$U_{n,w} = \begin{cases} \phi'_w(x_{i,j}), & \text{se } w = b^n, \text{ para } n = 1, \dots, N_{P,P'} \text{ e } w = 1, \dots, \ell'. \\ 0, & \text{c.c.} \end{cases}$$

$$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_\ell, \alpha'_1, \alpha'_2, \dots, \alpha_{\ell'})^T. \quad \text{2}$$

Por enquanto, essas observações são suficientes para (2.5).

Outra vez, esqueçamos nossa constante de escalamento e passemos para o termo

$$\sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B_i \cap (Z \setminus \Omega)} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k, I \right). \quad (2.6)$$

De novo, substituindo a expressão da função distância, temos

$$\sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} \left(\sum_{x \in B_i \cap (Z \setminus \Omega)} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x) - I(x) \right)^2 \right),$$

que se expressa também como

$$\sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} \left\| \sum_{k=1}^{m_i} \alpha_i^k \phi_i(x_i) - \begin{pmatrix} I(x_i^1) \\ I(x_i^2) \\ \vdots \\ I(x_i^{N_i}) \end{pmatrix} \right\|_2^2.$$

²Note que as entradas de H e U são matrizes, e não números.

Ou seja, recuperando N_2 o termo (2.6) pode escrito como

$$||M\alpha - w||_2^2/N_2,$$

onde

$$M_{n,k} = \begin{cases} \phi_k(x_i), & \text{se } k = i^n \text{ para } n = 1, \dots, N_{P,P'} \text{ e } k = 1, \dots, \ell. \\ 0, & \text{c.c.} \end{cases},$$

$$w = \left(I(x_1^1), \dots, I(x_1^{N_1}), \dots, I(x_{N_P}^1), \dots, I(x_{N_P}^{N_{N_P}}) \right)^T.$$

Uma análise semelhante pode ser feita para o termo

$$\sum_{\{i|B'_i \cap (Z \setminus \Omega) \neq \emptyset\}} d|_{B'_i \cap (Z \setminus \Omega)} \left(\sum_{k=1}^{m'_i} \alpha'^k_i \phi'^k_i, I \right),$$

chegando assim numa expressão na forma

$$||M'\alpha' - w'||_2^2/N_2,$$

com

$$M'_{n,k} = \begin{cases} \phi'_k(x), & \text{se } k = i'^n \text{ para } n = 1, \dots, N_{P,P'} \text{ e } k = 1, \dots, \ell'. \\ 0, & \text{c.c.} \end{cases},$$

$$w' = \left(I(x'_1^1), \dots, I(x'_1^{N_1}), \dots, I(x'_{N_{P'}}^1), \dots, I(x'_{N_P}^{N'_{N_P}}) \right)^T.$$

Concluimos então que o problema (2.3) é na verdade um problema de mínimos quadrados na forma

$$\min_{\alpha} ||A\alpha - b||_2^2, \quad (2.7)$$

sendo

$$A = \begin{pmatrix} M/N_2 & | & 0 \\ 0 & | & M'/N_2 \\ H/N_1 & | & -U/N_1 \end{pmatrix}, \quad b = \begin{pmatrix} w/N_1 \\ w'/N_1 \\ 0 \end{pmatrix}.$$

Até aqui, não teríamos nenhuma dificuldade para resolver (2.7). Mínimos quadrados é um problema bastante conhecido e é possível resolvê-lo de diversas maneiras. Uma delas inclusive, reduzindo-o à solução de um sistema linear. Isto é, gostaríamos de resolver o problema de minimizar a seguinte função em α :

$$f(\alpha) = \langle A\alpha - b, A\alpha - b \rangle = (A\alpha - b)^T (A\alpha - b) = \alpha^T A^T A \alpha - 2\alpha^T A^T b + b^T b$$

Escrevendo a expressão do gradiente de f temos

$$\nabla f(\alpha) = A^T A \alpha - 2A^T b.$$

E assim resolveríamos o sistema

$$A^T A \alpha = 2A^T b$$

para achar o ponto crítico da função f . O obstáculo aqui é que, na prática, as soluções densas desse problema se mostram muito pouco adequadas às aplicações (aprofundaremos esse ponto no

Capítulo 3). Assim, quando gostaríamos de obter, entre as soluções α , aquela que fosse a mais esparsa possível. No nosso problema em específico, a esparsidade significa que queremos encontrar soluções f_i que sejam mais “puras” possíveis, ou seja, se pareçam ao máximo com o nosso dicionário (Definição 4). Ou seja, gostaríamos de minimizar a chamada *norma 0* de α :

$$\|\alpha\|_0 = \min \{i | \alpha_i \neq 0\}.$$

Todavia, minimizar a *norma 0* de um vetor, é provado ser *NP*-difícil (ver [GJ02]). Entretanto, podemos aproximar a solução desse problema resolvendo um outro, a saber (ver [FNW07]):

$$\min \|\alpha\|_1.$$

Em resumo gostaríamos de obter a solução para o problema bi-objetivo:

$$\min(\|A\alpha - b\|_2^2, \|\alpha\|_1). \quad (2.8)$$

Para uma descrição detalhada sobre otimização multiobjetivo e as diferentes técnicas de resolução, ver por exemplo [Sam11].

O problema (2.8) permite várias formas de abordagem. A forma que elegemos para tratá-lo é chamada mínimos quadrados penalizados (ver por exemplo [FNW07].) :

$$\min_{\alpha \in \mathbb{R}^n} \|A\alpha - b\|^2 + \lambda \|\alpha\|_1, \quad \lambda \in (0, 1). \quad (2.9)$$

Note que (2.9) possui uma função objetivo contínua, porém não diferenciável. Como mostrado em [FNW07], fazendo a mudança de variáveis $\alpha = u - v$, nosso problema se mostra equivalente a resolver um problema de programação quadrática com restrições de não-negatividade, ou seja

$$\begin{aligned} \min_{u, v \in \mathbb{R}^n} \quad & \|A(u - v) - b\|^2 + \lambda \sum_{i=1}^n (u_i + v_i) \\ \text{s.a.} \quad & u_i \geq 0, v_i \geq 0, \quad i = 1, \dots, n. \end{aligned} \quad (2.10)$$

Em [FNW07], o método do gradiente espectral projetado mostrou-se muito adequado para resolver (2.10). Essa abordagem está entre os objetivos das nossas próximas seções.

2.1.1 Caso com um número qualquer de coberturas

Desenvolveremos aqui alguns detalhes importantes para a definição da função objetivo quando nosso número de coberturas é $n_g \geq 0$. Supondo que cada uma de nossas coberturas P_i sejam formadas por ℓ_i conjuntos, isto é, $P_i = \{B_{i,1}, B_{i,2}, \dots, B_{i,\ell_i}\}$ para $i = 1, 2, 3, \dots, n_g$, como podemos generalizar o cálculo da nossa função objetivo em (2.1)?

Usando a linguagem da seção anterior, definamos para duas coberturas P e P' quaisquer entre $P_1, \dots, P_{n_g}$ a seguinte função:

$$\Psi_{P,P'}(\alpha, \alpha') = \sum_{\{i,j | B_i \cap B'_j \cap \Omega \neq \emptyset\}} \left(\sum_{x \in B_i \cap B'_j \cap \Omega} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x) - \sum_{k=1}^{m'_j} \alpha'_j{}^k \phi'_j{}^k(x) \right)^2 \right)$$

Note que $\Psi_{P,P'}$ já apareceu em (2.3). Em palavras, seu efeito é calcular a distância, sujeita às coordenadas α, α' , de todas as funções f_i, f'_j que se intersectam na região Ω . Definimos também a função:

$$\Psi_P(\alpha) = \sum_{\{i|B_i \cap (Z \setminus \Omega) \neq \emptyset\}} \left(\sum_{x \in B_i \cap (Z \setminus \Omega)} \left(\sum_{k=1}^{m_i} \alpha_i^k \phi_i^k(x) - I(x) \right)^2 \right)$$

Já Ψ_P calcula a distância entre as funções f_i definidas nos conjuntos da partição P e a região $Z \setminus \Omega$. Note que, com essa notação, podemos reescrever nosso problema de mínimos quadrados (2.3) da última seção como:

$$\min \Psi_{P_1} + \Psi_{P_1, P_2} + \Psi_{P_2} \quad (2.11)$$

Agora, para generalizarmos 2.11, apenas estenderemos nossa expressão para a expressão do problema geral, como:

$$\min_{\alpha} \Psi_{P_1} + \Psi_{P_1, P_2} + \Psi_{P_2} + \Psi_{P_2, P_3} + \Psi_{P_3} + \Psi_{P_3, P_4} + \dots$$

Seguindo os mesmos passos da seção anterior, não é difícil mostrar que continuaremos chegando num problema na forma:

$$\begin{aligned} \min_{\bar{u}, \bar{v} \in \mathbb{R}^n} \quad & \|\bar{A}(u - v) - \bar{b}\|^2 + \lambda \sum_{i=1}^n (\bar{u}_i + \bar{v}_i) \\ \text{s.a.} \quad & \bar{u}_i \geq 0, \bar{v}_i \geq 0, \quad i = 1, \dots, n, \end{aligned} \quad (2.12)$$

com $\bar{A}, \bar{b}, \bar{u}, \bar{v}$, generalizando os correspondentes vetores e matrizes da seção anterior.

2.2 O método do gradiente espectral projetado

O método do Gradiente Espectral Projetado foi introduzido em [BMR09] e combina o método clássico de gradiente projetado com dois ingredientes: o passo de Barzilai-Borwein [BB88] e a busca linear não-monótona de Grippo, Lampariello, e Lucidi [GLL86]. Na última década, o SPG foi utilizado com sucesso em diversas aplicações práticas. Dentre elas destacamos a resolução dos problemas de otimização associados a máquinas de suporte vetorial (ver, por exemplo, [CEGLJ09], [SZZ05]), ao cálculo de perturbações condicionais não-lineares ótimas (ver, por exemplo, [Lim]), e a *compressive sensing* (ver por exemplo [VDBF08], [FNW07]) dentre outros.

O método do Gradiente Espectral Projetado (SPG) (do inglês *Spectral Projected Gradient*) aplica-se ao problema

$$\text{Min } f(x) \quad \text{sujeita a } x \in \Omega$$

onde $f : \mathbb{R}^n \rightarrow \mathbb{R}$ é contínua e diferenciável e Ω é um conjunto convexo. Por utilizar apenas informação de primeira ordem e precisar de uns poucos vetores de tamanho n na sua implementação, o SPG é adequado para problemas de grande porte. A eficiência do método está diretamente relacionada à existência de uma forma eficiente de projetar um ponto arbitrário $x \in \mathbb{R}^n$ no conjunto convexo Ω .

As iterações do método SPG são da forma $x_{k+1} = x_k + \alpha_k d_k$ onde $d_k = P_{\Omega}(x_k - \lambda_k \nabla f(x_k)) - x_k$ e α_k é tal que x_{k+1} satisfaz a condição de Armijo não monótona dada por

$$f(x_{k+1}) \leq f_{\max} + \alpha_k \gamma_{\text{arm}} \nabla f(x_k)^T d_k. \quad (2.13)$$

Na determinação de d_k , $P_\Omega(\cdot)$ representa a projeção Euclideana no conjunto convexo Ω , i.e.,

$$P_\Omega(\bar{x}) = \operatorname{argmin} \frac{1}{2} \|x - \bar{x}\|_2^2 \quad \text{sujeita a } x \in \Omega$$

e λ_k é o passo espectral dado por

$$\lambda_k = \max \left\{ \lambda_{\min}, \min \left\{ \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}, \lambda_{\max} \right\} \right\}, \quad (2.14)$$

onde, $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$ e $0 \leq \lambda_{\min} \leq \lambda_{\max} \leq +\infty$ são salvaguardadas dadas. Na prática, utiliza-se $\lambda_{\min} = 10^{-30}$ e $\lambda_{\max} = 10^{30}$. Em (2.13) $\gamma_{arm} > 0$ é a constante de Armijo (tipicamente $\gamma_{arm} = 10^{-4}$) e

$$f_{\max} = \max\{f(x_{k-j}) | 0 \leq j \leq \min\{k, M-1\}\}. \quad (2.15)$$

Acima, $M \geq 1$ é uma constante que regula a não monotonicidade da busca linear. A condição (2.15) garante decréscimo suficiente a cada M iterações. Se $M = 1$ então (2.15) coincide com o critério clássico de Armijo. Em [BMR00] sugere-se $M = 10$ como valor *default*. O passo espectral λ_k definido em (2.14) relaciona o método SPG com os métodos do tipo secante. Os métodos quasi-Newton [DJS96] do tipo secante obedecem a fórmula $x_{k-1} = x_k + \alpha_k d_k$ com

$$d_k = B_k^{-1} \nabla f(x_k), \quad (2.16)$$

onde as matrizes B_k satisfazem as equações secante

$$B_k s_{k-1} = y_{k-1}. \quad (2.17)$$

Naturalmente, o custo de resolver o sistema linear (2.16) é o fator decisivo do custo da iteração dos métodos quasi-Newton e a escolha da matriz B_k pode fazer o método estar tão próximo de um método do tipo gradiente ou do método de Newton quanto desejado. Se impusermos $B_k = \tau_k I$, com $\tau_k \in \mathbb{R}$, então a equação secante (2.17) fica $\tau_k s_{k-1} = y_{k-1}$. Esta equação muito provavelmente não terá solução, mas se considerarmos a solução de quadrados mínimos teremos

$$\tau_k = s_{k-1}^T y_{k-1} / s_{k-1}^T s_{k-1}.$$

Ou seja, $B_k = \tau_k I$ e $d_k = \bar{\lambda}_k \nabla f(x_k)$ com $\bar{\lambda}_k = 1/\tau_k = s_{k-1}^T s_{k-1} / s_{k-1}^T y_{k-1}$. O passo espectral λ_k em (2.14) é a projeção de $\bar{\lambda}_k$ no intervalo $[\lambda_{\min}, \lambda_{\max}]$. Assim, o passo espectral proporciona ao SPG algum tipo de informação de segunda ordem com custo mínimo. O uso da busca linear não monótona do tipo Armijo (no lugar da versão monótona) faz com que a primeira tentativa de passo na direção d_k que, como acabou de ser explicada, carrega alguma informação de segunda ordem, tenha mais chances de ser aceita.

Sejam $M \geq 1, 0 < \lambda_{\min} \leq \lambda_{\max} < +\infty, 0 < \sigma_1 \leq \sigma_2 < 1, \gamma_{arm} > 0$ e $\epsilon > 0$ parâmetros dados. O método começa com $x_0 \in \Omega$ e $\lambda_0 \in [\lambda_{\min}, \lambda_{\max}]$ arbitrários. Dados $x_k \in \Omega$ e $\lambda_k \in [\lambda_{\min}, \lambda_{\max}]$, o algoritmo que segue descreve como obter x_{k+1} e λ_{k+1} e como terminar o processo.

Algoritmo

Passo 1. *Critério de parada*

Se $\|P_\Omega(x_k - \nabla f(x_k)) - x_k\| \leq \epsilon$, pare declarando que x_k é ponto estacionário.

Passo 2. *Backtracking*

Passo 2.1. Calcule $d_k = P_\Omega(x_k - \lambda_k \nabla f(x_k)) - x_k$ e faça $\alpha \leftarrow 1$.

Passo 2.2. Se

$$f(x_k + \alpha d_k) \leq \max_{\{0 \leq j \leq \min\{k, M-1\}\}} f(x_{k-j}) + \gamma_{arm} \alpha \nabla f(x_k)^T d_k$$

defina $\alpha_k = \alpha$, $x_{k+1} = x_k + \alpha d_k$, $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ e vá para o Passo 3. Caso contrário, defina $\alpha_{new} \in [\sigma_1 \alpha, \sigma_2 \alpha]$, faça $\alpha \leftarrow \alpha_{new}$ e volte ao Passo 2.2.

Passo 3. *Cálculo do passo spectral.*

Se $s_k^T y_k \leq 0$, defina $\lambda_{k+1} = \lambda_{max}$. Senão, defina, $\lambda_{k+1} = \max\{\lambda_{min}, \min\{s_k^T s_k / s_k^T y_k, \lambda_{max}\}\}$ e volte ao Passo 1.

No Passo 2.2, o cálculo de α_{new} utiliza uma interpolação quadrática unidimensional com salvaguarda tomando o passo $\alpha_{new} = \alpha/2$ quando o minimizador da quadrática unidimensional fica fora do intervalo $[\sigma_1, \sigma_2 \alpha]$. Note que o processo de salvaguarda baseia-se em $[\sigma_1, \sigma_2 \alpha]$ no lugar da escolha usual $[\sigma_1 \alpha, \sigma_2 \alpha]$. Isto significa que, quando a interpolação tende a rejeitar 90% (para $\sigma_1 = 0.1$) do intervalo original $[0, 1]$, julga-se que a previsão não é confiável e o processo mais conservativo de bissecção é utilizado. Experimentos numéricos em [BMR00] mostraram que, para algumas classes de problemas, este procedimento obtém melhores resultados do que o procedimento clássico.

O algoritmo seguinte descreve a busca linear em detalhe.

Algoritmo

Passo 1. Calcule $f_{max} = \max\{f(x_{k-j}) | 0 \leq j \leq \min\{k, M-1\}\}$ e $\delta = \nabla f(x_k)^T d_k$ e faça $\alpha \leftarrow 1$.

Passo 2. Faça $x_+ \leftarrow x_k + \alpha d_k$. Se

$$f(x_+) \leq f_{max} + \gamma_{arm} \alpha \delta$$

faça $\alpha_k = \alpha$ e termine.

Passo 3. Faça $\alpha_{tmp} \leftarrow \frac{1}{2} \alpha^2 \delta / (f(x_+) - f(x_k) - \alpha \delta)$.

Se $\alpha_{tmp} \geq \sigma_1$ e $\alpha_{tmp} \leq \sigma_2 \alpha$, faça $\alpha \leftarrow \alpha_{tmp}$. Caso contrário, faça $\alpha \leftarrow \alpha/2$. Vá para o Passo 2.

A boa definição e a teoria de convergência do SPG podem ser encontradas em [BMR00], onde também são descritos experimentos numéricos relacionados à aplicação do SPG a problemas de minimização em caixas. Em [BMR01] são mostrados experimentos numéricos com uma família de problemas de grande porte (com milhões de variáveis e restrições) associados a alocação de pontos no plano. A extensão do SPG para o caso de projeções inexatas é apresentada em [BMR03] e a extensão para problemas com restrições lineares foi introduzida em [ABMY05]. Para uma contextualização histórica do SPG e uma análise detalhada de suas variantes, veja [BMR09].

2.3 Implementação da função objetivo e do gradiente da função objetivo

Nesta seção, apresentaremos um algoritmo que, dada uma imagem, uma região danificada Ω e blocos reconstrutores, calcule o valor da função objetivo do problema (2.1). Apesar da precisão

e generalidade do nosso modo de apresentação na Seção 2.1, esse não é um modo muito prático para expor uma implementação do nosso método. Sendo assim, se torna necessário especificar alguns parâmetros de modo a uniformizar os testes computacionais que desenvolveremos. Mas antes, definiremos um importante conceito usado ao longo dessa seção:

Definição 5 Chamamos de “bloco de tamanho t ” uma função ou matriz B tomando valores no conjunto $[0, 255] \cap \mathbb{Z}$ definida num conjunto de pixels da forma $\{x, x+1, \dots, x+(t-1)\} \times \{y, y+1, \dots, y+(t-1)\}$. O par ordenado $\text{pos}(B) = (\text{posx}(B), \text{posy}(B)) = (x, y)$ é dito a “posição do bloco B ” na imagem I .

Alguns parâmetros agora serão fixados. O primeiro deles é o número de elementos do dicionário:

Definição 6 Assumiremos que todas as nossas funções procuradas f_i, f'_j , $i = 1, 2, \dots, \ell$, $j = 1, 2, \dots, \ell'$, definidas na seção anterior serão blocos de um tamanho pré-fixado t . Assumiremos também que será dada uma lista de pares ordenados, com as posições de cada um desses blocos, isto é: $(\text{posx}(B_1), \text{posy}(B_1)), (\text{posx}(B_2), \text{posy}(B_2)), \dots, (\text{posx}(B_{\ell_1+\ell_2+\ell_3+\dots+\ell_g}), \text{posy}(B_{\ell_1+\ell_2+\ell_3+\dots+\ell_g}))$. Observação: Por uniformidade na apresentação, nos referiremos aos blocos, apenas citando seus índices. Ou seja, a lista supracitada será dada como: $(\text{posx}(1), \text{posy}(1)), (\text{posx}(2), \text{posy}(2)), \dots, (\text{posx}(\ell_1 + \ell_2 + \ell_3 + \dots + \ell_g), \text{posy}(\ell_1 + \ell_2 + \ell_3 + \dots + \ell_g)))$.

Definição 7 Por convenção assumiremos também que $m_1 = m_2 = \dots = m_\ell = m'_1 = m'_2 = \dots = m'_{\ell'} = M$. Além disso, $\phi_i^1 = \phi_i^2 = \dots = \phi_i^M = \Phi_i$, $\phi_j^1 = \phi_j^2 = \dots = \phi_j^M = \Phi_j \quad \forall i = 1, 2, \dots, \ell, \quad j = 1, 2, \dots, \ell'$. Em palavras, pedimos que o dicionário seja o mesmo para todos os blocos $B_1, B_2, \dots, B_{\ell+\ell'}$. Sendo assim, nosso dicionário será o conjunto: $\{\Phi_1, \Phi_2, \dots, \Phi_M\}$.

Definição 8 Chamaremos de imagem danificada a função $R: Z \rightarrow [-1, 255]$ definida por:

$$R(i, j) = \begin{cases} I(i, j), & \text{se } (i, j) \in Z \setminus \Omega \\ -1, & \text{se } (i, j) \in \Omega. \end{cases}$$

Para implementar nossa função objetivo e o gradiente da mesma, precisaremos de algumas rotinas auxiliares. O objetivo de cada uma dessas rotinas ficará claro em breve. Por ora, apenas daremos suas implementações e chamaremos atenção para o fato de que os parâmetros t, λ, nl, nc serão sempre dados como entrada.

Para facilitar a notação em que escreveremos os códigos, faremos a seguinte definição:

Definição 9 Seja A uma matriz qualquer e b um vetor, escreveremos $A(i_1 : i_2, j_1 : j_2)$ para representar a submatriz de A formada pelas entradas $A(i_1, j_1), A(i_1 + 1, j_1), A(i_1 + 2, j_1), \dots, A(i_2, j_1), A(i_1, j_2), A(i_1 + 1, j_2), \dots, A(i_2, j_2), \dots, A(i_2, j_2)$ de A . Analogamente, $b(i_1 : i_2)$, representa o “subvetor” de b formado por $b(i_1), b(i_1 + 1), b(i_1 + 2), b(i_1 + 3), \dots, b(i_2)$.

A seguir, temos as descrições e a implementação de cada uma das rotinas:

Rotina 1. Cria uma lista com a posição de cada um dos blocos reconstrutores, calculando também a quantidade desses blocos e atribuindo um índice a cada um deles. Essa rotina percorre a imagem construindo n_g coberturas formadas por L blocos de tamanho t com um deslocamento de 2 pixels na

Rotina 1 Calculo L , $posx$ e $posy$

```
1: input :  $n_g, nl, nc, t, R$ ;  
2: output :  $L, posx, posy$ ;  
3:  $L \leftarrow 0$ ;  
4:  $desloc \leftarrow 0$ ;  
5: for  $k = 1, \dots, n_g$  do  
6:    $i \leftarrow 1$   
7:   while  $i \leq nl$  do  
8:      $j \leftarrow 1$   
9:     while  $j \leq nc$  do  
10:      if  $\min(R(i : \min(i + t - 1, nl), j : \min(j + t - 1, nc))) == -1$  then  
11:         $L \leftarrow L + 1$ ;  
12:         $posx(L) \leftarrow i$ ;  
13:         $posy(L) \leftarrow j$ ;  
14:      end if  
15:       $j \leftarrow j + desloc$ ;  
16:    end while  
17:     $i \leftarrow i + desloc$ ;  
18:  end while  
19:   $desloc \leftarrow desloc + 2$ ;  
20: end for
```

horizontal e na vertical de uma para outra. Para analisar a complexidade da Rotina 1, observe que temos três laços, com tamanhos de n_g, nl, nc , e um cálculo do mínimo aninhado com complexidade t^2 no pior caso. Tem-se no total $O(n_g \times nl \times nc \times t^2)$ operações no pior caso.

Rotina 2. Nessa Rotina, calculamos duas matrizes: h e v . Cada elemento da matriz h associa a cada pixels da imagem o número de blocos que o sobrepõem. Como escolheremos entre 2 e 5 coberturas, $h(i, j)$ será sempre limintado superiormente por 5. Cada elemento da matriz multi-dimensional v , por sua vez, associa a cada pixel os índices de todos os blocos que o sobrepõem. Novamente aqui, temos três laços aninhados, com tamanhos respectivamente de L, t, t , totalizando assim um número total de operações $O(L \times t^2)$.

Rotina 3. Cria uma lista com a posição dos pixels que são sobrepostos por algum dos blocos re-construtores, ou seja, cria uma lista com os pixels (i, j) tais que $h(i, j) \neq 0$. Essa lista é armazenada nos vetores $pixhx$ e $pixhy$, e a quantidade é armazenada na variável $Npixh$. Novamente aqui não é difícil concluir que na rotina 3 temos uma ordem complexidade de $O(nl \times nc)$.

Rotina 4. Calcula os blocos que farão parte do dicionário, e a quantidade de blocos no mesmo. Chamaremos de tam_dic a quantidade de blocos no dicionário. Essa rotina será implementada usando a região conhecida da imagem. Observamos aqui que essa rotina usa todos os blocos do dicionário. Na prática porém, usamos apenas alguns blocos da região conhecida da imagem. (ver capítulo 3) Seguindo o mesmo raciocínio e contando a quantidade de laços no algoritmo, chegamos a complexidade $O(nl \times nc \times t^2)$. Note que nesse caso, a operação da linha 5, conta como um laço

Rotina 2 Calculando h e v

```
1: input :  $L, posx, posy$ ;  
2: output :  $h, v$ ;  
3:  $h \leftarrow 0$   
4: for  $k = 1, \dots, L$  do  
5:   for  $i = posx(k), \dots, posx(k) + t - 1$  do  
6:     for  $j = posy(k), \dots, posy(k) + t - 1$  do  
7:        $h(i, j) \leftarrow h(i, j) + 1$ ;  
8:        $v(i, j, h(i, j)) \leftarrow k$ ;  
9:     end for  
10:  end for  
11: end for
```

Rotina 3 Lista $pixhx$, $pixhy$ e a quantidade $Npixh$.

```
1: input :  $nl, nc, h$   
2: output :  $Npixh, pixhx, pixy$   
3:  $Npixh \leftarrow 0$   
4: for  $i = 1, \dots, nl$  do  
5:   for  $j = 1, \dots, nc$  do  
6:     if  $h(i, j) \neq 0$  then  
7:        $Npixh \leftarrow Npixh + 1$ ;  
8:        $pixhx(Npixh) \leftarrow i$ ;  
9:        $pixhy(Npixh) \leftarrow j$ ;  
10:    end if  
11:  end for  
12: end for
```

Rotina 4 Calculando a matriz tridimensional ϕ e sua quantidade de blocos tam_dic .

```
1: input :  $nl, nc, t, tam\_dic$ ;  
2: output :  $\phi$ ;  
3: for  $i = 1, \dots, nl$  do  
4:   for  $j = 1, \dots, nc$  do  
5:      $bloco \leftarrow R(i : \min(i + t - 1, nl), j : \min(j + t - 1, nc))$ ;  
6:     if  $\min(bloco) \neq -1$  then  
7:        $tam\_dic \leftarrow tam\_dic + 1$ ;  
8:        $\phi(:, :, tam\_dic) \leftarrow bloco$ ;  
9:     end if  
10:  end for  
11: end for
```

de tamanho t^2 .

Rotina 5. Calcula a função *block*. Essa função faz a combinação linear dos blocos que formam a base e armazena numa matriz tridimensional. A rotina 5 tem complexidade $O(L \times t^2)$.

Rotina 5 Calculando a função *block*.

```

1: input :  $n, x, L, tam\_dic, t, phi$ 
2: output : bloco
3:  $a \leftarrow x(1 : n/2)$ ;
4:  $b \leftarrow x(n/2 + 1 : n)$ ;
5:  $\alpha \leftarrow a - b$ ;
6: for  $i = 1, \dots, t$  do
7:   for  $j = 1, \dots, t$  do
8:     for  $l = 1, \dots, L$  do
9:        $bloco(i, j, l) \leftarrow 0$ ;
10:    end for
11:  end for
12: end for
13: for  $k = 1 \dots, L$  do
14:    $blocos(:, :, k) \leftarrow phi(:, :, 1 : tam\_dic) * \alpha((k - 1) * tam\_dic + 1 : k * tam\_dic)$ ;
15: end for

```

Agora, podemos implementar nossa função objetivo como apresentado no Algoritmo 6.

Para analisar a ordem de complexidade da nossa função objetivo, observe que o laço da linha 13 tem, para cada l , $h(pixh(l), pixy(l))$ operações. No entanto, a função h é limitada superiormente pelo número n_g , o que significa que, no pior caso, temos n_g operações no laço da linha 13. Assim, contando os laços das linhas 11, 13 e 22, temos uma quantidade de operações da ordem de $Npixh \times n_g$.

Na linha 5, temos a quantidade de operações dada pela função *block*, ou seja, $O(L \times t \times t)$. Já na linha 35, temos $O(n)$ operações. Totalizamos assim, para cada execução da função objetivo, uma complexidade de execução de $O(Npixh \times n_g + L \times t \times t + n)$. Ou seja, a complexidade na execução da nossa função objetivo depende da quantidade de pixels na nossa região Ω , expressa pela grandeza $Npixh$, da quantidade de coberturas n_g que escolhemos para “rodar” nosso método e do tamanho de blocos t que escolhemos.

Adiantamos que, no método de otimização usado nessa dissertação, apenas o gradiente da função objetivo será necessário. Aproveitamos para apresentar no Algoritmo 7 / 8 o pseudocódigo do gradiente.

A complexidade na execução do gradiente se diferencia da complexidade da própria função objetivo f principalmente pelas operações realizadas nas linhas 20, 21, 37, 38, 39 e 40. Sendo que em cada linha temos uma complexidade de operações da ordem de $O(tam_dic)$. Sendo assim, nos laços das linhas 14 e 25 temos um grau de complexidade da ordem de $n_g \times tam_dic$, onde n_g vem da

Algoritmo 6 Função Objetivo

```
1: Input :  $n, x, tam\_dic, t, nl, nc, v, R, L, phi, h, Npixh, pixhx, pixhy, posx, posy, \lambda$ ;  
2: Output :  $f$ ;  
3:  $a \leftarrow (x_1, x_2, \dots, x_{n/2})$ ;  
4:  $b \leftarrow (x_{n/2+1}, x_{n/2+2}, \dots, x_n)$ ;  
5:  $blocos \leftarrow block(n, x, L, tam\_dic, t, phi)$ ;  
6:  $fparc1 \leftarrow 0$ ;  
7:  $fparc2 \leftarrow 0$ ;  
8:  $f \leftarrow 0$ ;  
9:  $N_1 \leftarrow 0$ ;  
10:  $N_2 \leftarrow 0$ ;  
11: for  $l = 1, \dots, Npixh$  do  
12:    $i \leftarrow pixhx(l), j \leftarrow pixhy(l)$ ;  
13:   if  $R(pixh(l), pixy(l)) \neq -1$  then  
14:     for  $k = 1, \dots, h(pixh(l), pixy(l))$  do  
15:        $bind \leftarrow v(i, j, k)$ ;  
16:        $ipos \leftarrow i - posx(bind) + 1$ ;  
17:        $jpos \leftarrow j - posy(bind) + 1$ ;  
18:        $fparc1 \leftarrow fparc1 + (blocos(ipos, jpos, bind) - R(pixh(l), pixy(l)))^2$ ;  
19:        $N_1 \leftarrow N_1 + 1$ ;  
20:     end for  
21:   else  
22:     for  $k = 1, \dots, h(pixh(l), pixy(l)) - 1$  do  
23:        $bind1 \leftarrow v(i, j, k1)$ ;  
24:        $ipos1 \leftarrow i - posx(bind1) + 1$ ;  
25:        $jpos1 \leftarrow j - posy(bind1) + 1$ ;  
26:        $bind2 \leftarrow v(i, j, k + 1)$ ;  
27:        $ipos2 \leftarrow i - posx(bind2) + 1$ ;  
28:        $jpos2 \leftarrow j - posy(bind2) + 1$ ;  
29:        $fparc2 \leftarrow fparc2 + (blocos(ipos1, jpos1, bind1) - blocos(ipos2, jpos2, bind2))^2$ ;  
30:        $N_2 \leftarrow N_2 + 1$ ;  
31:     end for  
32:   end if  
33: end for  
34:  $f \leftarrow fparc1/N_1 + fparc2/N_2 + \lambda * (\sum_{i=1}^n (a(i) + b(i)))$ ;
```

Algoritmo 7 Gradiente da Função - Parte 1

```
1: Input :  $n, x, tam\_dic, t, nl, nc, v, R, L, phi, h, Npixh, pixhx, pixhy, posx, posy, \lambda$ ;  
2: Output :  $\nabla f(x)$ ;  
3:  $a \leftarrow (x_1, x_2, \dots, x_{n/2})$ ;  
4:  $b \leftarrow (x_{n/2+1}, x_{n/2+2}, \dots, x_n)$ ;  
5:  $blocos \leftarrow block(n, x, L, tam\_dic, t, phi)$ ;  
6:  $gparc1(1 : n/2) \leftarrow 0$ ;  
7:  $gparc2(1 : n/2) \leftarrow 0$ ;  
8:  $g(1 : n) \leftarrow 0$ ;  
9:  $N_1 \leftarrow 0$ ;  
10:  $N_2 \leftarrow 0$ ;  
11: CONTINUA...
```

limitação superior de $h(i, j)$. Portanto, a complexidade de execução na avaliação do gradiente da nossa função objetivo é da ordem de $O(Npixh \times n_g \times tam_dic + L \times t^2 + n)$, onde $L \times t^2$ corresponde à complexidade de execução da função *block* na linha 5 do pseudo-código do gradiente.

Algoritmo 8 Gradiente da Função - Parte 2

```
12: for  $l = 1, \dots, N_{pixh}$  do
13:    $i \leftarrow pixhx(l), j \leftarrow pixhy(l);$ 
14:   if  $R(i, j) \neq -1$  then
15:     for  $k = 1, \dots, h(i, j)$  do
16:        $bind \leftarrow v(i, j, k);$ 
17:        $ipos \leftarrow i - posx(bind) + 1;$ 
18:        $jpos \leftarrow j - posy(bind) + 1;$ 
19:        $indvar \leftarrow (bind - 1) * tam\_dic;$ 
20:        $temp \leftarrow 2 * (blocos(ipos, jpos, bind) - Rij);$ 
21:        $phi\_temp \leftarrow (phi(ipos, jpos, 1), phi(ipos, jpos, 2), \dots, phi(ipos, jpos, tam\_dic));$ 
22:        $gparc1(indvar + 1 : indvar + tam\_dic) \leftarrow gparc1(indvar + 1 : indvar + tam\_dic) +$ 
         $temp * phi;$ 
23:        $N_1 \leftarrow 0;$ 
24:     end for
25:   else
26:     for  $k = 1 : (h(i, j) - 1)$  do
27:        $bind1 \leftarrow v(i, j, k_1);$ 
28:        $ipos1 \leftarrow i - posx(bind1) + 1;$ 
29:        $jpos1 \leftarrow j - posy(bind1) + 1;$ 
30:        $indvar1 \leftarrow (bind1 - 1) * tam\_base;$ 
31:        $temp1 \leftarrow blocos(ipos1, jpos1, bind1);$ 
32:        $bind2 \leftarrow v(i, j, k_1 + 1);$ 
33:        $ipos2 \leftarrow i - posx(bind2) + 1;$ 
34:        $jpos2 \leftarrow j - posy(bind2) + 1;$ 
35:        $indvar2 \leftarrow (bind2 - 1) * tam\_dic;$ 
36:        $temp2 \leftarrow blocos(ipos2, jpos2, bind2);$ 
37:        $diff \leftarrow 2 * (temp1 - temp2);$ 
38:        $phi\_temp1 \leftarrow phi(ipos1, jpos1, 1), phi(ipos1, jpos1, 2), \dots, phi(ipos1, jpos1, tam\_dic);$ 
39:        $phi\_temp2 \leftarrow phi(ipos2, jpos2, 1), phi(ipos2, jpos2, 2), \dots, phi(ipos2, jpos2, tam\_dic);$ 
40:        $gparc1(indvar1+1 : indvar1+tam\_dic) \leftarrow gparc1(indvar1+1 : indvar1+tam\_dic) +$ 
         $diff * phi1;$ 
41:        $gparc2(indvar2+1 : indvar2+tam\_dic) \leftarrow gparc2(indvar2+1 : indvar2+tam\_dic) +$ 
         $diff * phi2;$ 
42:        $N_2 \leftarrow N_2 + 1;$ 
43:     end for
44:   end if
45:    $g(1 : n/2) = gparc1(1 : n/2)/N_1 + gparc2(1 : n/2)/N_2;$ 
46:    $g(n/2 + 1 : n) = -g(1 : n/2);$ 
47:    $g = g + \lambda / (max(N_1, N_2));$ 
48: end for
```

Capítulo 3

Experimentos computacionais

Esse capítulo terá como foco a análise da influência dos parâmetros ϕ , t , n_g , e λ na imagem recuperada, assim como a apresentação dos resultados/imagens devolvidas pelo nosso método. Todos os experimentos deste capítulo foram conduzidos em um computador com processador Intel Xeon com 2.67GHz e 8GB de memória RAM, rodando o sistema operacional GNU/Linux. As implementações foram feitas em Matlab versão R2013a (8.1.0.604) 64-bit (glnxa64).

Lembramos que no capítulo 1, definimos a imagem recuperada calculando a média aritmética dos pixels das camadas que sobrepõem cada um dos pixels (i, j) da região desconhecida Ω . Propomos aqui, um novo modo de definir a imagem recuperada: escolheremos uma cobertura em específico, e definiremos os pixels dessa cobertura após a resolução do problema de otimização, como os pixels da região recuperada. No nosso caso, escolhemos como cobertura padrão, a primeira cobertura. Experimentos numéricos e visuais mostraram a qualidade da imagem recuperada com essa nova definição em alguns casos.

Nas análises apresentadas nesse capítulo, a notação utilizada é a seguinte:

- $f(x^*)$: Valor da função objetivo no ponto ótimo x^* ;
- MQ : Valor da expressão de mínimos quadrados aplicada no ponto ótimo: $\|Ax^* - b\|_2^2$;
- $\|x^*\|_1$: Norma 1 do ponto ótimo;
- $\|x^*\|_0$: Norma 0 do ponto ótimo;
- L : Quantidade total de blocos gerados pelo método;
- t : Tamanho de cada bloco;
- $iter$: Quantidade total de iterações executadas pelo método de otimização de gradiente espectral projetado;
- $\#fcnt$: Quantidade total de avaliações da função objetivo no método de otimização;
- $\|g_p\|_\infty$: Norma infinita do gradiente espectral projetado (tamanho de cada passo no método de otimização);

- $psnr$: Medida $psnr$, definida por $psnr(x, x^*) = 10 \times \log_{10} \left(\frac{1}{\frac{1}{n} \|x - x^*\|_2^2} \right)$, onde x é a imagem recuperada e x^* é uma imagem de referência. Em geral, quanto maior o valor da medida $psnr$, maior a qualidade da imagem recuperada.¹,
- $psnr_m$: Medida $psnr$, em que a imagem recuperada consiste na média dos pixels das camadas;
- $psnr_c$: Medida $psnr$, em que a imagem recuperada consiste nos pixels da camada 1.

Seguiremos agora na análise da relação entre os diferentes parâmetros ϕ , t , n_g , λ na qualidade da imagem reconstruída representada pela medida $psnr$. Usamos o critério de parada $\|P_\Omega(x_k - \nabla f(x_k)) - x_k\| \leq 10^{-6}$.

Serão apresentadas tabelas com os dados numéricos e gráficos para a medida $psnr$ em função de cada um dos parâmetros livres: ϕ , t , n_g , λ e ao final de cada seção é apresentada a imagem com medida $psnr$ mais alta. As imagens geradas com os outros valores de parâmetros com $psnr$ menor, assim como qualquer outro dado que o leitor possa se interessar relativo a presente dissertação, podem ser solicitadas pelo e-mail: almeida@ime.usp.br.

3.1 A escolha do dicionário

É importante salientar aqui que a escolha certa do dicionário é crucial para a qualidade da imagem recuperada e uma atenção especial é dedicada à análise da melhor escolha do dicionário em problemas de *basis pursuit*. Existem dois paradigmas centrais que são usualmente adotados nesse tipo de escolha: o primeiro explora propriedades matemáticas do modelo, fazendo a decomposição esparsa usando como dicionário funções matemáticas como *wavelets*, *curvelets*, *contourlets*. A segunda abordagem usa técnicas como *machine learning* para escolher os elementos do dicionário baseando-se em um conjunto de treinamento, que em geral carrega informações prévias do sinal ou imagem a ser reconstruída. Para um melhor aprofundamento de alguns procedimentos de escolha de dicionários em problemas de representação esparsa; ver, por exemplo, [RBE10, AEB06].

Nesse trabalho faremos a escolha do dicionário, primeiramente, inspirados na segunda abordagem citada. Escolheremos os blocos que formam o dicionário de maneira que os mesmos tragam alguma informação das regiões intactas da imagem. Tais regiões devem ser potencialmente boas para preencher a região Ω . Note que aqui há um fator altamente subjetivo na escolha dessas regiões. Por exemplo, para escolher o dicionário quando Ω está contido totalmente numa região da imagem com uma única textura², basta selecionarmos blocos daquela textura (bons resultados foram obtidos selecionando entre 4 e 8 blocos); a Figura 3.1 exemplifica esse tipo de caso.

Quando Ω atravessa mais de uma textura na imagem, é razoável que escolhamos blocos para formar um dicionário que carregue informações tanto da fronteira de encontro entre as duas texturas como também de cada uma das texturas separadamente. Em resumo, nosso dicionário deve conter informações (blocos) de todas as regiões que são atravessadas por Ω . A Figura 3.2 exemplifica este caso para a escolha do dicionário.

¹Para um detalhamento sobre a medida $psnr$ e algumas outras medidas de qualidade em processamento de imagens; ver por exemplo [EF95].

²Uma textura é definida basicamente como as diferentes regiões de grande extensão da imagem. Por exemplo, na Figura 3.5 temos as texturas T1, T2 e T3

Figura 3.1: Região Ω (surfista a esquerda) a ser retirada

Figura 3.2: Exemplo de escolha do dicionário quando a Ω atravessa mais de uma região

Além disso, observou-se na prática que nem sempre essa escolha é suficiente para um bom resultado final, sendo necessário acrescentarmos o que chamaremos de “blocos de ajuste”. Esses blocos adicionais tem o papel de ajustar a propagação de linhas que cheguem na fronteira de Ω para o interior de Ω . Em alguns casos é notável a diferença a imagem recuperada incluindo tais blocos no dicionário para com a imagem recuperada sem os mesmos; ver Figuras 3.3, 3.4.

Figura 3.3: Imagem poste a ser adulterada.

Os “blocos de ajuste” nos casos aqui apresentados serão matrizes nulas com única linha ou coluna não nula e com valor 1. Por exemplo, com $t = 4$, poderíamos ter o seguinte conjunto de blocos de ajuste acrescentado ao dicionário:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Como esses blocos pretendem propagar linhas chegando em Ω , se torna necessário que adicionemos esses blocos adicionais de maneira que as linhas não nulas representem os ângulos de encontros das linhas da figura que interceptam com a fronteira de Ω . Como exemplo considere a Figura 3.2 e note que existem linhas chegando em Ω que predominantemente são horizontais, isso nos sinaliza que acrescentando ao dicionário os blocos que são matrizes nulas com apenas uma linha não nula com valor 1, podemos melhorar a qualidade da nossa imagem, o que de fato acontece, como veremos nas próximas seções.

3.2 Relação de n_g e $psnr$

Nessa seção, vamos analisar como o número de coberturas altera a qualidade da imagem recuperada. Na Figura 3.1, que daqui em diante chamaremos de “Waves”, fixaremos primeiramente

(a) Imagem “poste” processada sem blocos adicionais.

(b) Imagem “poste” processada com blocos adicionais.

Figura 3.4: Manipulação de imagem, sem blocos adicionais e com blocos adicionais.

Figura 3.5: Texturas T1, T2 e T3 de uma imagem genérica.

$t = 15$ com ϕ dado pela Figura 3.6 e $\lambda = 0$ variando n_g e observando como $psnr$ se comporta para cada uma das diferentes imagens. Nos casos estudados aqui, a qualidade da imagem obtida com n_g maior do que 5 ficou semelhante àquelas obtidas com $n_g \in \{2, 3, 4, 5\}$. Devido a este fato, além de um considerável custo computacional acrescido a cada cobertura adicionada, foi suficiente variar n_g entre 2 e 5 para obtermos uma boa representação da imagem recuperada. A Tabela 3.1 apresenta os resultados obtidos para a imagem “waves” com $t = 15$. Para facilitar a visualização

λ	n_g	$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	$iter$	$\#fcnt$	$\ g_p\ _\infty$	$psnr_m$	$psnr_c$
0.00	2	9.30e-03	9.30e-03	4.33e+01	179/180	45	15	577	844	9.90e-07	19.26	18.98
0.00	3	8.94e-03	8.94e-03	7.17e+01	294/296	74	15	638	973	9.67e-07	19.20	18.91
0.00	4	8.80e-03	8.80e-03	8.97e+01	374/376	94	15	524	782	6.50e-07	19.52	19.04
0.00	5	8.74e-03	8.74e-03	1.19e+02	491/496	124	15	567	845	8.75e-07	19.53	19.06

Tabela 3.1: Variando n_g com $t = 15$ em “Waves”

Figura 3.6: “Waves” com dicionário. ($t = 21$).

da relação entre as variáveis, gráficos com as medidas $psnr_m$ e $psnr_c$ em função de n_g são mostrados na Figura 3.7.

Fixando agora $t = 21$ e ainda $\lambda = 0.0$ podemos ver os resultados na Tabela 3.2, novamente com gráficos dados na Figura 3.8.

λ	$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	$iter$	$\#fcnt$	$\ g_p\ _\infty$	$psnr_m$	$psnr_c$
0.00	8.96e-03	8.96e-3	3.14e+01	131/132	33	21	988	1469	9.90e-07	19.55	19.69
0.00	9.33e-03	9.33e-03	4.56e+01	196/196	49	21	1052	1614	9.14e-07	19.50	19.58
0.00	9.64e-03	9.64e-03	6.33e+01	264/268	67	21	815	1244	9.76e-07	19.56	19.83
0.00	9.37e-03	9.37e-03	7.87e+01	334/336	84	21	719	1089	9.89e-07	19.56	19.73

Tabela 3.2: Variando n_g com $t = 21$.

(a) n_g vs $psnr_c$. ($t=15$).

(b) n_g vs $psnr_m$. ($t=15$).

Figura 3.7: n_g vs $psnr$ em “Waves”. ($t=15$).

(a) n_g vs $psnr_c$. ($t=21$).

(b) n_g vs $psnr_m$. ($t=21$).

Figura 3.8: n_g vs $psnr$. ($t=21$).

Os gráficos nos mostram que tanto a medida $psnr_c$ por camadas, como a medida $psnr_m$ por média de camadas, tendem a crescer com o aumento de n_g , com um pico em $n_g = 4$ em “waves”.

Note que é bem razoável que tenhamos uma tendência ao crescimento da qualidade da imagem com o número de coberturas que tivermos na nossa região desconhecida, uma vez que o número de coberturas representa, de certa maneira, a quantidade de informação que podemos propagar para dentro da nossa região desconhecida. A melhor imagem recuperada para “waves”, com o parâmetro $t = 15$ e $n_g = 4$ estão na Figura 3.9.

Repetiremos a nossa análise com mais uma imagem, denominada “Lena”, mostrada na Figura 3.10.

Figura 3.9: “Waves” por camada única, com $n_g = 4$, $t = 15$, $\lambda = 0.00$.

Devido a espessura de Ω e para termos uma visão mais panorâmica da variação dos parâmetros, fixaremos agora t em 7, 9 e 15 para acompanharmos a variação de $psnr$. Os resultados obtidos se encontram na Tabela 3.3. O comportamento ainda não apresenta um padrão, no entanto, para $n_g = 4$, ainda obtivemos um valor bem próximo do máximo ($psnr = 30.45$) para as medidas de qualidade.

$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	n_g	$iter$	$\ g_p\ _\infty$	time(min)	$psnr_m$	$psnr_c$
3.40e-04	3.40e-04	1.11e+03	7549/7751	337	7	2	17059	9.52e-07	129.8	30.01	30.00
1.38e-03	1.38e-03	4.11e+02	2960/3105	135	15	2	40963	1.00e-06	393.7	28.41	28.27
1.96e-03	1.96e-03	2.08e+02	1850/1978	86	21	2	20542	9.66e-07	218.9	27.06	26.93
4.12e-04	4.12e-04	1.44e+03	11213/11546	502	7	3	13955	9.99e-07	228.6	30.32	30.22
1.45e-03	1.45e-03	5.26e+02	4149/4370	190	15	3	19849	9.94e-07	233.1	29.06	28.52
1.93e-03	1.93e-03	3.11e+02	2796/2990	130	21	3	23931	9.75e-07	332.4	27.47	27.03
4.47e-04	4.47e-04	1.62e+03	14503/14973	651	7	4	9888	9.85e-07	172.1	30.27	30.17
1.46e-03	1.46e-03	6.83e+02	5674/5934	258	15	4	17489	9.98e-07	252.6	28.88	28.29
1.89e-03	1.89e-03	4.18e+02	3796/4071	177	21	4	24400	9.50e-07	421.2	27.60	26.96
4.70e-04	4.70e-04	1.84e+03	18338/18952	824	7	5	7555	9.99e-07	155.2	30.45	30.21
1.44e-03	1.44e-03	8.19e+02	6938/7314	318	15	5	16475	9.96e-07	299.9	28.95	28.29

Tabela 3.3: “Lena” variando n_g com $t = 7, 15, 21$ fixado.

Figura 3.10: Lena com região Ω (riscos) em branco.

Frisamos que obtivemos resultados próximos do máximo $psnr$ com o $n_g = 3, 4, 5$ em todas as demais imagens testadas. Por isso, fixaremos nas seções subsequentes $n_g = 3$, $n_g = 4$ ou $n_g = 5$.

3.3 Relação de t e $psnr$

Agora, como mencionado e justificado na seção anterior, fixaremos $n_g = 3, 4, 5$ dependendo da imagem considerada. Em seguida variaremos o parâmetro t até o tamanho máximo para que nenhum bloco do dicionário intercepte a região desconhecida ou a borda da imagem, ainda deixando fixado $\lambda = 0.00$. Os resultados de “waves” são mostrados na Tabela 3.4.

Os gráficos da Figura 3.11 nos ajudam a enxergar o que está acontecendo na relação entre t e $psnr$. Fica claro pelos gráficos que tanto $psnr_c$ como $psnr_m$ atingem seu pico em $t = 21$. Note que não há um comportamento monótono para as duas medidas, ou seja, ambas não apresentaram um padrão bem definido, a não ser pelo fato de possuírem o mesmo t correspondendo ao seu maior valor. Mostramos nas Figuras 3.12 e 3.13 as imagens correspondentes a esse número de tamanho de bloco.

Nossos próximos dois exemplos para a variação de t serão a imagem “Lena” e uma imagem que chamaremos de “Wall”. As posições dos dicionários base para “Wall” e “Lena”, assim como a região Ω , estão representadas nas Figuras 3.14 e 3.15 respectivamente, com $t = 9$ apenas a título de ilustração. Uma observação relativa aos blocos adicionais do dicionário se torna necessária aqui. Note que em “Wall”, Ω atravessa dois tipos de texturas diferentes. Na região da imagem que contém tijolos, existe uma grande quantidade de linhas verticais e horizontais, o que dificulta obtermos uma

λ	$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	$iter$	$\#fcnt$	$\ g_p\ _\infty$	$psnr_m$	$psnr_c$
0.00	5.21e-03	5.21e-03	3.28e+02	1141/1168	292	7	1332	2114	9.95e-07	19.3	19.1
0.00	5.82e-03	5.82e-03	2.39e+02	913/932	233	8	989	1514	9.85e-07	19.4	19.2
0.00	7.05e-03	7.05e-03	2.02e+02	793/808	202	9	794	1203	9.99e-07	19.4	19.1
0.00	8.43e-03	8.43e-03	1.67e+02	668/672	168	10	586	889	9.10e-07	19.5	19.2
0.00	8.16e-03	8.16e-03	1.43e+02	581/584	146	11	632	984	9.97e-07	19.6	19.2
0.00	8.31e-03	8.31e-03	1.30e+02	525/532	133	12	604	902	9.40e-07	19.5	19.4
0.00	8.64e-03	8.64e-03	1.14e+02	466/468	117	13	498	757	9.17e-07	19.3	19.0
0.00	8.83e-03	8.83e-03	1.05e+02	431/432	108	14	454	672	7.84e-07	19.4	19.0
0.00	8.81e-03	8.81e-03	8.97e+01	374/376	94	15	524	782	6.50e-07	19.5	19.0
0.00	8.80e-03	8.80e-03	8.67e+01	356/364	91	16	745	1105	8.93e-07	19.3	19.2
0.00	8.65e-03	8.65e-03	8.15e+01	335/344	86	17	686	1033	9.00e-07	19.4	19.0
0.00	8.69e-03	8.69e-03	7.50e+01	317/320	80	18	720	1092	6.38e-07	19.4	19.2
0.00	8.93e-03	8.93e-03	6.99e+01	294/300	75	19	615	901	9.88e-07	19.3	19.0
0.00	9.90e-03	9.90e-03	6.34e+01	269/272	68	20	710	1097	7.50e-07	19.4	18.8
0.00	9.64e-03	9.64e-03	6.33e+01	264/268	67	21	815	1244	9.76e-07	19.6	19.8
0.00	9.28e-03	9.28e-03	6.01e+01	251/256	64	22	841	1283	9.83e-07	19.3	18.8
0.00	9.47e-03	9.47e-03	5.33e+01	226/228	57	23	695	1039	9.93e-07	19.5	19.6
0.00	9.19e-03	9.19e-03	4.93e+01	210/216	54	24	590	909	9.81e-07	19.3	19.1
0.00	9.11e-03	9.11e-03	4.69e+01	201/204	51	25	935	1409	9.61e-07	19.2	19.2
0.00	8.53e-03	8.53e-03	5.05e+01	215/216	54	26	788	1231	9.95e-07	19.2	18.9
0.00	8.65e-03	8.65e-03	4.77e+01	204/208	52	27	864	1304	9.93e-07	19.3	19.0
0.00	8.71e-03	8.71e-03	4.28e+01	183/184	46	28	862	1342	8.14e-07	19.2	18.9
0.00	8.59e-03	8.59e-03	4.40e+01	188/188	47	29	676	1019	4.59e-07	19.2	18.9
0.00	8.95e-03	8.95e-03	3.78e+01	163/164	41	30	810	1256	9.77e-07	19.2	19.1
0.00	8.55e-03	8.55e-03	3.75e+01	159/160	40	31	865	1331	9.80e-07	19.1	18.7
0.00	8.10e-03	8.10e-03	3.85e+01	164/164	41	32	894	1387	9.91e-07	19.3	18.9
0.00	7.96e-03	7.96e-03	3.85e+01	164/164	41	33	1046	1600	8.86e-07	19.3	19.2
0.00	7.89e-03	7.89e-03	3.66e+01	156/156	39	34	759	1127	9.99e-07	19.1	19.0
0.00	7.98e-03	7.98e-03	3.70e+01	160/160	40	35	799	1214	9.13e-07	19.1	18.8

Tabela 3.4: “Waves” variando t com $n_g = 4$ fixado.

(a) t vs $psnr_c$.

(b) t vs $psnr_m$.

Figura 3.11: “Waves” t vs $psnr$.

Figura 3.12: “Waves” por média dos pixels das camadas, com $n_g = 4$, $t = 21$, $\lambda = 0.00$.

Figura 3.13: “Waves” por camada única, com $n_g = 4$, $t = 21$, $\lambda = 0.00$

imagem final de qualidade razoável apenas com o dicionário formado apenas com blocos extraídos da região intacta da imagem (ver Figura 3.17). Entra aqui a necessidade de incluímos os blocos adicionais com colunas e linhas não nulas (note que foram escolhidos blocos adicionais com essa característica devido a inclinação das linhas chegando em Ω). Notaremos que haverá uma qualidade significativa na qualidade da imagem para com a imagem sem os blocos adicionais. Compararemos também a qualidade da imagem “Lena” adicionando esses blocos adicionais na próxima seção. Os resultados da variação de t para as duas imagens estão exibidos nas Tabelas 3.5 e 3.6.

$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	n_g	$iter$	$\ g_p\ _\infty$	$time(min)$	$psnr_m$	$psnr_c$
2.63e-04	2.63e-04	2.54e+03	29533/30567	1329	5	5	4128	9.92e-07	31.3	30.73	30.50
4.70e-04	4.70e-04	1.84e+03	18338/18952	824	7	5	7555	9.99e-07	151.4	30.45	30.21
6.52e-04	6.52e-04	1.40e+03	13713/14260	620	9	5	10280	9.98e-07	207.4	30.26	29.68
9.01e-04	9.01e-04	1.14e+03	10448/10879	473	11	5	9797	9.89e-07	209.8	29.68	29.28
1.17e-03	1.17e-03	9.18e+02	8232/8648	376	13	5	11389	9.95e-07	197.4	29.47	29.30
1.44e-03	1.44e-03	8.19e+02	6938/7314	318	15	5	16475	9.96e-07	297.9	28.95	28.29
1.55e-03	1.55e-03	6.91e+02	6037/6440	280	17	5	15842	9.98e-07	300.4	28.67	27.88
1.82e-03	1.82e-03	6.16e+02	5522/5865	255	19	5	17927	9.76e-07	372.6	27.97	27.86
1.91e-03	1.91e-03	5.08e+02	4729/5060	220	21	5	19386	9.94e-07	417.2	27.86	26.91

Tabela 3.5: “Lena” variando t com $n_g = 5$ fixado.

Concluimos através dos resultados expostos que um número razoável para o t em “Lena” é $t = 5$ enquanto para “Wall” é $t = 9$. Mostramos as imagens recuperadas correspondentes para “Lena” e “Wall” nas Figuras 3.16 e 3.17.

Note que estamos com 3 tipos de estruturas de imagem diferentes sendo abordadas aqui, a saber com 1, 2 ou várias regiões distintas sendo atravessadas por Ω e com blocos adicionais e sem blocos adicionais. Note também que não há um comportamento padronizado para a qualidade da imagem em função do tamanho dos blocos. Esse fato muitas vezes pode exigir que escolhamos o tamanho

$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	n_g	$iter$	$\ g_p\ _\infty$	$time(min)$	$psnr_m$	$psnr_c$
1.06e-04	1.06e-04	2.50e+03	9810/9940	355	5	3	27939	3.16e-06	180.0	16.83	16.83
3.59e-03	3.59e-03	1.33e+03	5858/5964	213	7	3	25608	2.00e-06	180.0	16.71	16.58
7.14e-03	7.14e-03	5.75e+02	4109/4256	152	9	3	11970	9.88e-07	91.7	17.53	17.40
9.27e-03	9.27e-03	3.46e+02	3041/3164	113	11	3	8567	9.85e-07	69.2	17.41	17.15
1.14e-02	1.14e-02	2.55e+02	2458/2576	92	13	3	8322	1.00e-06	67.6	17.23	17.10
1.32e-02	1.32e-02	1.97e+02	2050/2156	77	15	3	7915	9.87e-07	65.9	16.87	16.66
1.51e-02	1.51e-02	1.67e+02	1804/1904	68	17	3	6483	9.96e-07	56.8	16.63	16.41
1.66e-02	1.66e-02	1.27e+02	1523/1624	58	19	3	6378	9.99e-07	47.7	16.64	16.35
1.68e-02	1.68e-02	1.07e+02	1338/1428	51	21	3	6163	9.92e-07	52.8	16.35	16.16
1.86e-02	1.86e-02	9.28e+01	1185/1260	45	23	3	5772	9.91e-07	51.2	16.39	16.24
1.93e-02	1.93e-02	9.26e+01	1186/1260	45	25	3	4932	9.98e-07	47.5	16.07	16.02
2.13e-02	2.13e-02	7.51e+01	1034/1092	39	27	3	5996	9.96e-07	30.8	16.34	16.37
2.09e-02	2.09e-02	6.17e+01	889/952	34	29	3	5809	8.28e-07	27.2	15.92	15.94
2.12e-02	2.12e-02	6.31e+01	931/980	35	31	3	7033	9.52e-07	39.1	16.23	15.96
2.16e-02	2.16e-02	5.35e+01	805/868	31	33	3	4894	1.00e-06	26.7	15.95	15.91

Tabela 3.6: “Wall” variando t com $n_g = 5$ fixado.

Figura 3.14: Dicionario e região Ω para “Wall”.

Figura 3.15: Dicionario e região Ω para “Lena”.

Figura 3.16: “Lena” recuperada por camada única com $n_g = 4$, $t = 5$ e $\lambda = 0.0$ sem blocos adicionais.

Figura 3.17: “Wall” recuperada por média de camadas com $n_g = 3$, $t = 9$ e $\lambda = 0.0$ sem blocos adicionais.

do bloco para cada tipo particular de imagem e região Ω .

3.4 Relação de λ e $psnr$

Finalmente, podemos analisar a influência do parâmetro de regularização λ nas medidas de qualidade $psnr_m$ e $psnr_c$. Iniciaremos com $\lambda = 0$ nas imagens “Waves”, “Lena” e “Wall”, acrescentando 0.01 para observar como a qualidade da imagem varia em função de λ , com ou sem blocos adicionais dependendo da estrutura da imagem. O valor de acréscimo, nesse caso 0.01, pode precisar ser escolhido em cada situação dependendo do escalamento da função objetivo, critério de parada e características particulares da imagem. Nos casos tratados nessa dissertação, o valor 0.01 se mostrou adequado à visualização do comportamento da qualidade da imagem em função do nosso parâmetro λ . As Tabelas 3.7, 3.8 e 3.9 mostram os resultados obtidos para as imagens “Waves”, “Wall” e “Lena” sendo “Wall” e “Lena” com blocos adicionais.

Na análise dos experimentos realizados, fica claro o papel do parâmetro λ no nosso método. Para λ pequeno pertencente a um certo intervalo $[0, \epsilon)$, a qualidade das imagens (nesse caso representada pela medida $psnr$) aumenta atingindo um pico $psnr_{max}$ correspondente a um λ_{max} e a uma certa esparsidade, decrescendo à medida que λ cresce se afasta de ϵ . É como se o parâmetro λ de alguma forma obtivesse uma certa “pureza” ótima para a combinação linear do dicionário correspondendo a cada bloco reconstrutor, ou seja, λ seleciona apenas os blocos do dicionário que são mais importantes para cada bloco reconstrutor. Veja o comportamento da qualidade da imagem em função do parâmetro λ mais explicitamente para as três imagens nos gráficos das Figuras 3.18, 3.19 e 3.20. Note que para “Waves”, $\lambda_{max} = 0.16$ e $psnr_{max} = 19.95$, em “Lena” temos $\lambda_{max} = 0.06$ e $psnr_{max} = 31.35$ e em “Wall” chegamos em $\lambda_{max} = 0.08$ e $psnr_{max} = 18.24$.

Figura 3.18: Variando λ em “Wall”, com $n_g = 3$, $t = 9$.

Concluimos então nossa análise de parâmetro selecionando os melhores parâmetros para cada ima-

λ	$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	$iter$	$\#fcnt$	$\ g_p\ _\infty$	$psnr_m$	$psnr_c$
0.00	9.64e-03	9.64e-03	6.33e+01	264/268	67	21	815	1244	9.76e-07	19.56	19.83
0.01	1.08e-02	9.65e-03	6.20e+01	254/268	67	21	1046	1557	2.90e-07	19.61	19.85
0.02	1.20e-02	9.67e-03	6.14e+01	249/268	67	21	725	1026	9.59e-07	19.65	19.87
0.03	1.31e-02	9.69e-03	6.10e+01	249/268	67	21	725	1053	7.95e-07	19.69	19.88
0.04	1.42e-02	9.72e-03	6.05e+01	247/268	67	21	698	988	9.83e-07	19.72	19.89
0.05	1.54e-02	9.76e-03	6.01e+01	244/268	67	21	660	959	4.42e-07	19.76	19.90
0.06	1.65e-02	9.80e-03	5.97e+01	243/268	67	21	666	1001	6.52e-07	19.78	19.91
0.07	1.76e-02	9.85e-03	5.93e+01	237/268	67	21	680	986	9.79e-07	19.81	19.91
0.08	1.87e-02	9.91e-03	5.88e+01	234/268	67	21	683	978	9.96e-07	19.84	19.91
0.09	1.98e-02	9.97e-03	5.84e+01	234/268	67	21	561	805	8.28e-07	19.86	19.90
0.10	2.09e-02	1.00e-02	5.80e+01	228/268	67	21	559	802	9.38e-07	19.88	19.89
0.11	2.19e-02	1.01e-02	5.76e+01	222/268	67	21	519	754	9.85e-07	19.89	19.88
0.12	2.30e-02	1.02e-02	5.72e+01	216/268	67	21	614	896	6.17e-07	19.91	19.88
0.13	2.41e-02	1.03e-02	5.68e+01	213/268	67	21	500	713	9.91e-07	19.92	19.87
0.14	2.51e-02	1.04e-02	5.65e+01	208/268	67	21	685	1026	8.55e-07	19.93	19.86
0.15	2.62e-02	1.05e-02	5.61e+01	205/268	67	21	450	660	8.93e-07	19.94	19.85
0.16	2.72e-02	1.06e-02	5.58e+01	202/268	67	21	523	744	9.91e-07	19.95	19.84
0.17	2.83e-02	1.07e-02	5.54e+01	200/268	67	21	564	827	6.91e-07	19.95	19.83
0.18	2.93e-02	1.08e-02	5.51e+01	193/268	67	21	532	778	9.64e-07	19.95	19.81
0.19	3.03e-02	1.09e-02	5.47e+01	188/268	67	21	645	965	9.80e-07	19.95	19.79
0.20	3.13e-02	1.10e-02	5.44e+01	183/268	67	21	534	793	4.91e-07	19.95	19.77
0.21	3.24e-02	1.12e-02	5.41e+01	182/268	67	21	665	973	9.35e-07	19.94	19.75
0.22	3.34e-02	1.13e-02	5.38e+01	176/268	67	21	650	962	9.98e-07	19.93	19.73
0.23	3.44e-02	1.14e-02	5.35e+01	173/268	67	21	602	866	9.73e-07	19.92	19.71
0.24	3.54e-02	1.15e-02	5.32e+01	168/268	67	21	553	806	9.94e-07	19.91	19.70
0.25	3.63e-02	1.17e-02	5.29e+01	164/268	67	21	635	937	6.47e-07	19.89	19.68
0.26	3.73e-02	1.18e-02	5.27e+01	163/268	67	21	562	800	9.96e-07	19.88	19.66
0.27	3.83e-02	1.19e-02	5.24e+01	159/268	67	21	612	904	8.77e-07	19.86	19.64
0.28	3.93e-02	1.21e-02	5.21e+01	157/268	67	21	537	796	9.93e-07	19.83	19.62
0.29	4.03e-02	1.22e-02	5.19e+01	154/268	67	21	567	823	9.82e-07	19.81	19.59
0.30	4.12e-02	1.23e-02	5.16e+01	154/268	67	21	611	880	9.31e-07	19.78	19.56
0.31	4.22e-02	1.25e-02	5.14e+01	153/268	67	21	663	985	9.53e-07	19.75	19.53
0.32	4.31e-02	1.26e-02	5.11e+01	151/268	67	21	679	967	9.53e-07	19.72	19.50
0.33	4.41e-02	1.28e-02	5.08e+01	149/268	67	21	599	868	9.75e-07	19.69	19.46
0.34	4.50e-02	1.29e-02	5.06e+01	147/268	67	21	628	902	5.74e-07	19.65	19.42
0.35	4.60e-02	1.31e-02	5.03e+01	146/268	67	21	609	871	8.88e-07	19.61	19.38
0.36	4.69e-02	1.33e-02	5.01e+01	146/268	67	21	457	658	9.97e-07	19.57	19.35
0.37	4.78e-02	1.34e-02	4.98e+01	144/268	67	21	583	846	9.47e-07	19.53	19.31
0.38	4.88e-02	1.36e-02	4.96e+01	144/268	67	21	512	734	9.66e-07	19.48	19.26
0.39	4.97e-02	1.38e-02	4.93e+01	142/268	67	21	449	644	8.20e-07	19.44	19.22

Tabela 3.7: Resultados “Waves” variando λ com $n_g = 4$ e $t = 21$ sem blocos adicionais.

λ	$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	n_g	$iter$	$\ g_p\ _\infty$	$time(min)$	$psnr_m$	$psnr_c$
0.00	1.44e-04	1.44e-04	2.18e+03	35770/37212	1329	5	5	3496	9.75e-07	34.0	31.18	31.05
0.01	4.74e-04	1.44e-04	2.18e+03	35770/37212	1329	5	5	10	8.37e-07	0.1	31.18	31.05
0.02	8.04e-04	1.44e-04	2.18e+03	35770/37212	1329	5	5	0	7.02e-07	0.0	31.18	31.05
0.03	1.13e-03	1.44e-04	2.18e+03	35770/37212	1329	5	5	0	7.37e-07	0.0	31.18	31.05
0.04	1.46e-03	1.44e-04	2.18e+03	35770/37212	1329	5	5	0	8.71e-07	0.0	31.18	31.05
0.05	9.91e-04	2.72e-04	1.07e+03	14325/37212	1329	5	5	2790	9.46e-07	26.3	31.34	31.06
0.06	1.11e-03	2.91e-04	1.02e+03	12657/37212	1329	5	5	821	8.84e-07	7.9	31.35	31.05
0.07	1.25e-03	2.91e-04	1.02e+03	12657/37212	1329	5	5	0	9.80e-07	0.0	31.35	31.05
0.08	1.36e-03	3.15e-04	9.68e+02	11317/37212	1329	5	5	598	9.97e-07	5.8	31.35	31.03
0.09	1.46e-03	3.38e-04	9.30e+02	10277/37212	1329	5	5	577	8.67e-07	5.6	31.34	31.00
0.10	1.56e-03	3.72e-04	8.81e+02	8956/37212	1329	5	5	1122	9.00e-07	10.8	31.31	30.94
0.11	1.67e-03	3.81e-04	8.72e+02	8684/37212	1329	5	5	260	8.50e-07	2.4	31.30	30.92
0.12	1.79e-03	3.81e-04	8.72e+02	8684/37212	1329	5	5	0	9.84e-07	0.0	31.30	30.92
0.13	1.88e-03	4.09e-04	8.45e+02	8058/37212	1329	5	5	682	8.92e-07	6.5	31.26	30.85
0.14	1.99e-03	4.21e-04	8.35e+02	7825/37212	1329	5	5	284	9.95e-07	2.5	31.24	30.82
0.15	2.09e-03	4.51e-04	8.11e+02	7258/37212	1329	5	5	962	9.82e-07	8.0	31.20	30.74
0.16	2.19e-03	4.58e-04	8.07e+02	7124/37212	1329	5	5	175	9.08e-07	1.4	31.18	30.72
0.17	2.29e-03	4.72e-04	7.98e+02	6881/37212	1329	5	5	389	9.92e-07	3.2	31.16	30.68
0.18	2.40e-03	4.86e-04	7.90e+02	6676/37212	1329	5	5	398	9.87e-07	3.3	31.13	30.64
0.19	2.49e-03	5.09e-04	7.77e+02	6351/37212	1329	5	5	759	9.95e-07	6.1	31.09	30.57
0.20	2.59e-03	5.26e-04	7.68e+02	6100/37212	1329	5	5	667	9.90e-07	5.5	31.05	30.52
0.21	2.69e-03	5.38e-04	7.63e+02	5946/37212	1329	5	5	370	8.96e-07	3.0	31.03	30.48
0.22	2.79e-03	5.38e-04	7.63e+02	5946/37212	1329	5	5	0	9.93e-07	0.0	31.03	30.48

Tabela 3.8: Resultados para “Lena”, variando λ com $n_g = 5$ e $t = 5$ e com blocos adicionais.

λ	$f(x^*)$	MQ	$\ x^*\ _1$	$\ x^*\ _0$	L	t	n_g	$iter$	$\ g_p\ _\infty$	$time(min)$	$psnr_m$	$psnr_c$
0.00	2.33e-03	2.33e-03	9.84e+02	6859/6992	152	9	3	22060	2.01e-06	180.0	17.33	17.32
0.01	2.98e-03	2.36e-03	8.57e+02	6461/6992	152	9	3	8297	9.61e-07	66.2	17.52	17.50
0.02	3.45e-03	2.53e-03	6.43e+02	5727/6992	152	9	3	13800	9.58e-07	111.3	17.84	17.82
0.03	3.84e-03	2.70e-03	5.29e+02	5239/6992	152	9	3	13534	9.96e-07	109.1	18.02	18.00
0.04	4.18e-03	2.85e-03	4.63e+02	4950/6992	152	9	3	12132	9.96e-07	97.9	18.14	18.12
0.05	4.49e-03	2.96e-03	4.27e+02	4758/6992	152	9	3	4573	1.00e-06	37.1	18.19	18.17
0.06	4.77e-03	3.10e-03	3.90e+02	4579/6992	152	9	3	6643	8.82e-07	53.1	18.22	18.20
0.07	5.04e-03	3.21e-03	3.66e+02	4442/6992	152	9	3	4523	1.00e-06	36.6	18.23	18.21
0.08	5.29e-03	3.31e-03	3.45e+02	4281/6992	152	9	3	4593	9.92e-07	15.1	18.24	18.21
0.09	5.53e-03	3.41e-03	3.30e+02	4167/6992	152	9	3	4930	9.99e-07	12.3	18.23	18.21
0.10	5.76e-03	3.49e-03	3.17e+02	4075/6992	152	9	3	3189	9.97e-07	7.7	18.23	18.21
0.11	5.98e-03	3.57e-03	3.07e+02	4015/6992	152	9	3	2549	9.99e-07	6.2	18.23	18.20
0.12	6.20e-03	3.64e-03	2.98e+02	3943/6992	152	9	3	1858	9.99e-07	4.4	18.22	18.19
0.13	6.40e-03	3.72e-03	2.89e+02	3876/6992	152	9	3	2022	9.99e-07	4.9	18.21	18.18
0.14	6.61e-03	3.79e-03	2.81e+02	3825/6992	152	9	3	1819 3.9	9.95e-07	4.4	18.19	18.16
0.15	6.80e-03	3.87e-03	2.74e+02	3767/6992	152	9	3	1803	9.97e-07	4.4	18.18	18.14
0.16	7.00e-03	3.94e-03	2.67e+02	3721/6992	152	9	3	1748	9.98e-07	4.3	18.16	18.13
0.17	7.18e-03	4.01e-03	2.61e+02	3666/6992	152	9	3	1988	1.00e-06	4.9	18.15	18.11
0.18	7.37e-03	4.06e-03	2.57e+02	3622/6992	152	9	3	917	9.16e-07	2.2	18.13	18.10
0.19	7.55e-03	4.12e-03	2.52e+02	3579/6992	152	9	3	983	9.58e-07	2.4	18.12	18.08
0.20	7.73e-03	4.19e-03	2.47e+02	3537/6992	152	9	3	1026	9.99e-07	2.5	18.10	18.06
0.21	7.90e-03	4.24e-03	2.43e+02	3491/6992	152	9	3	1121	9.17e-07	2.7	18.08	18.04
0.22	8.07e-03	4.30e-03	2.40e+02	3444/6992	152	9	3	1128	9.99e-07	2.7	18.07	18.02

Tabela 3.9: Resultados para “Wall”, variando λ com $n_g = 4$ e $t = 21$ e com blocos adicionais.

Figura 3.19: Variando λ em “Lena”, com $n_g = 5$, $t = 5$.

(a) n_g vs $psnr_c$.

(b) n_g vs $psnr_m$.

Figura 3.20: n_g vs $psnr$

gem. A saber, para “Waves” $n_g = 4, t = 21, \lambda = 0.16$ para a construção da imagem por média, e $\lambda = 0.06$ para a construção da imagem por única camada, em “Lena” temos $n_g = 5, t = 5, \lambda = 0.06$ tanto por camada única como por média de camadas e em “Wall” os parâmetros com melhores resultados foram $n_g = 3, t = 9, \lambda = 0.08$, também sem muitas discrepâncias em relação a camada única e média de pixels. As imagens conseguidas nos três casos, são mostradas em 3.21, 3.22 e 3.23.

Figura 3.21: “Waves” por camada única, com $n_g = 4, t = 21, \lambda = 0.06$.

Atente para a qualidade da recuperação da intersecção entre duas regiões na imagem “Lena” e “Wall”. Um fato notável também é que mesmo numa textura razoavelmente complicada com tijolos em “Wall”, nosso método conseguiu recuperar linhas de contorno e claridade dos tijolos.

3.5 Recuperando a imagem “Barbara” com parâmetros fixados *a priori*

O objetivo dessa seção é nos preocuparmos apenas com o dicionário para a recuperação de uma imagem nova, levando em conta os resultados da seção anterior para deixar fixados os demais parâmetros n_g, t e λ . A imagem a ser recuperada é denominada “Barbara”, tem região Ω e dicionário representados na Figura 3.24. Os valores escolhidos para os parâmetros $n_g = 5, t = 5$ e $\lambda = 0.06$ foram determinados com base no fato de que a região Ω de “Barbara” tem características muito parecidas com a região Ω de “Lena” (localização na imagem e espessura principalmente). De fato, foi obtida com essa “calibração” dos parâmetros uma imagem com uma qualidade visual razoável demonstrada na Figura 3.25.

Repare novamente na qualidade da região recuperada quando temos texturas com linhas atravessando Ω . Linhas horizontais contidas na roupa da mulher foram bem recuperadas e apesar de a imagem ter sido recuperada com parâmetros fixados com informações de outra imagem, e devido a isso não ser a melhor possível, a imagem recuperada como um todo apresentou um resultado razoável. Claro que essa definição de parâmetros pode ser aprimorada analisando uma quantidade

Figura 3.22: “Lena”, por média de camadas com $n_g = 5$, $t = 5$, $\lambda = 0.06$ e com blocos adicionais.

Figura 3.23: “Wall”, por média de camadas $n_g = 5$, $t = 5$, $\lambda = 0.06$ e com blocos adicionais.

Figura 3.24: *Barbara*, por média de camdas $n_g = 5$, $t = 5$, $\lambda = 0.06$ e com blocos adicionais.

maior de imagens recuperadas. Isso nos sugere, então, uma estratégia para a fixação dos diversos parâmetros: tendo um banco de imagens com parâmetros já analisados em função da qualidade da imagem, podemos fixar parâmetros para novas imagens, levando em conta estruturas (espessura da região Ω , quantidade de textura atravessadas por Ω , necessidade de blocos adicionais, etc) das imagens já pré analisadas em nosso banco de imagens.

Figura 3.25: *Barbara*, por média de camadas $n_g = 5$, $t = 5$, $\lambda = 0.06$ e com blocos adicionais.

Capítulo 4

Conclusões e trabalhos futuros

O problema de *imagem inpainting* é um famoso problema tratado na área de processamento de imagens, contando com as mais diversas modelagens e abordagens. Nessa dissertação foi estudada uma abordagem específica do problema, usando regularização l_1 e aplicando o método do gradiente espectral projetado à função objetivo decorrente da modelagem do problema. Implementamos o método dado em [Woh11], analisamos seu comportamento para diferentes imagens e comprovamos a qualidade visual do método para um conjunto de imagens. Uma decisão importante para bom funcionamento do método apresentado é a boa escolha dos parâmetros, para isso damos uma orientação geral de como escolhê-los da melhor forma no Capítulo 3.

Note que, no método tratado aqui, devemos escolher o dicionário manualmente como entrada para a definição da função objetivo a ser minimizada. Um próximo estudo possível é usar métodos de segmentação de imagens com máquinas de suporte vetorial ou outro algoritmo de aprendizagem de máquina (como por exemplo, *K-means*, *K-SVD*) para automatizar o processo de escolha do dicionário do método abordado aqui, além de explorar dicionários analíticos como *wavelets*, *curvelets*, etc. Note também que a função objetivo do nosso modelo privilegia soluções que tenham coeficientes com magnitude baixa, uma vez que a *norma 1* de alguma forma “encolhe” o módulo das variáveis da função objetivo para os menores valores que ainda podem ser solução do problema de mínimos quadrados. É de se esperar que uma função que possa não “incentivar” valores pequenos das variáveis e ainda assim requerer a esparsidade, se comporte melhor que a função *norma 1* usada no nosso modelo. Uma dessas funções poderia ser, por exemplo, a conhecida *função de heaviside*. Dada por :

$$Heav(t) = \begin{cases} 1, & \text{se } t > 0, \\ 0, & \text{se } t \leq 0 \end{cases}$$

Apesar da sua não continuidade, essa função pode ser aproximada por uma função diferenciável (ver, por exemplo, [FNW07]). Uma proposta de continuação desse trabalho poderia ser comparar os estudos aqui feitos com a *função de heaviside*, substituindo a função *norma 1* e notando que, apesar de ganharmos um número menor de variáveis para minimizar em nosso problema de otimização, perdemos a convexidade da nossa função objetivo. Esse fato obrigatoriamente exigirá uma maior atenção com relação aos parâmetros considerados, inclusive com o ponto inicial do nosso método iterativo. Uma outra abordagem possível e que segue a ideia apresentada em [Woh11] é a de minimizar a nossa função de mínimos quadrados, porém com variáveis binárias e exigindo esparsidade como restrição do problema de otimização combinatória. Essa modelagem alternativa

poderia ser exemplo o problema (4.1).

$$\begin{aligned}
 \min \quad & \|Ax - b\|_2^2 \\
 \text{s.a.} \quad & \sum_{i=1+kL}^{L+kL} x_i = 1, \quad k = 1, \dots, L. \\
 & x_i \in \{0, 1\} \quad i = 1 \dots n
 \end{aligned} \tag{4.1}$$

Referências Bibliográficas

- [ABMY05] Roberto Andreani, Ernesto G Birgin, José Mario Martínez, and Jinyun Yuan. Spectral projected gradient and variable metric methods for optimization with linear inequalities. *IMA Journal of Numerical Analysis*, 25(2):221–252, 2005.
- [AEB06] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, 2006.
- [AK06] Gilles Aubert and Pierre Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, volume 147. Springer Science & Business Media, 2006.
- [BB88] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.
- [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.
- [BDE09] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.
- [BMR00] Ernesto G Birgin, José Mario Martínez, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.
- [BMR01] Ernesto G Birgin, José Mario Martínez, and Marcos Raydan. Algorithm 813: Spg—software for convex-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 27(3):340–349, 2001.
- [BMR03] Ernesto G Birgin, José Mario Martínez, and Marcos Raydan. Inexact spectral projected gradient methods on convex sets. *IMA Journal of Numerical Analysis*, 23(4):539–559, 2003.
- [BMR09] Ernesto G Birgin, José Mario Martínez, and Marcos Raydan. Spectral projected gradient methods. pages 3652–3659, 2009.

- [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.
- [CEGLJ09] Debora Cores, René Escalante, María González-Lima, and Oswaldo Jimenez. On the use of the spectral projected gradient method for support vector machines. *Computational & Applied Mathematics*, 28(3):327–364, 2009.
- [CPT04] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, 2004.
- [CRT06] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.
- [DJS96] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Siam, 1996.
- [Don06] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.
- [EDE96] L YAROSLAVSKY-M EDEN. *Fundamentals of Digital Optics*. Birkhäuser, 1996.
- [EF95] Ahmet M Eskicioglu and Paul S Fisher. Image quality measures and their performance. *Communications, IEEE Transactions on*, 43(12):2959–2965, 1995.
- [EL⁺99] Alexei Efros, Thomas K Leung, et al. 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.
- [EM76] Gilberte Emile-Male. *The restorer’s handbook of easel painting*. Van Nostrand Reinhold, 1976.
- [FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.
- [FNW07] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, 2007.
- [FSM09] Mohamed-Jalal Fadili, J-L Starck, and Fionn Murtagh. Inpainting and zooming using sparse representations. *The Computer Journal*, 52(1):64–79, 2009.
- [Gir98] Federico Girosi. An equivalence between sparse approximation and support vector machines. *Neural computation*, 10(6):1455–1480, 1998.
- [GJ02] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman, 2002.

- [GLL86] Luigi Grippo, Francesco Lampariello, and Stephano Lucidi. A nonmonotone line search technique for newton’s method. *SIAM Journal on Numerical Analysis*, 23(4):707–716, 1986.
- [GLM14] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *Signal Processing Magazine, IEEE*, 31(1):127–144, 2014.
- [GWM11] Patrick R Gill, Albert Wang, and Alyosha Molnar. The in-crowd algorithm for fast basis pursuit denoising. *Signal Processing, IEEE Transactions on*, 59(10):4595–4605, 2011.
- [HB95] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. ACM, 1995.
- [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.
- [KMFR95] Anil C Kokaram, Robin D Morris, William J Fitzgerald, and Peter JW Rayner. Interpolation of missing data in image sequences. *Image Processing, IEEE Transactions on*, 4(11):1509–1519, 1995.
- [Li11] Xin Li. Image recovery via hybrid sparse representations: A deterministic annealing approach. *Selected Topics in Signal Processing, IEEE Journal of*, 5(5):953–962, 2011.
- [Lim] Jessé Américo Gomes de Lima. *Otimização em Meteorologia: cálculo de perturbações condicionais não-lineares ótimas*. PhD thesis, Universidade de São Paulo.
- [Lla12] Boris Chullo Llave. *Aplicação do método do Gradiente Espectral Projetado ao problema de Compressive Sensing*. PhD thesis, Universidade de São Paulo, 2012.
- [LLX⁺01] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (ToG)*, 20(3):127–150, 2001.
- [LM11] Amy N Langville and Carl D Meyer. *Google’s PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.
- [MCW05] Dmitry M Malioutov, Müjdat Cetin, and Alan S Willsky. Homotopy continuation for sparse signal representation. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP’05). IEEE International Conference on*, volume 5, pages v–733. IEEE, 2005.
- [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.
- [NMS93] Mark Nitzberg, David Mumford, and Takahiro Shiota. *Filtering, segmentation, and depth*. Springer-Verlag New York, Inc., 1993.

- [RBE10] Ron Rubinstein, Alfred M Bruckstein, and Michael Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.
- [Sam11] Phillipe Rodrigues Sampaio. *Teoria, métodos e aplicações de otimização multiobjetivo*. PhD thesis, Universidade de São Paulo, 2011.
- [SP98] Eero P Simoncelli and Javier Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, volume 1, pages 62–66. IEEE, 1998.
- [SS04] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.
- [SZZ05] Thomas Serafini, Gaetano Zanghirati, and Luca Zanni. Gradient projection methods for quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20(2-3):353–378, 2005.
- [VDBF08] Ewout Van Den Berg and Michael P Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.
- [Wal85] Sarah Walden. *The ravished image*. St. Martin’s Press: New York, 1985.
- [WL00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.
- [Woh11] Brendt Wohlberg. Inpainting by joint optimization of linear combinations of exemplars. *Signal Processing Letters, IEEE*, 18(1):75–78, 2011.
- [XS10] Zongben Xu and Jian Sun. Image inpainting by patch propagation using patch sparsity. *Image Processing, IEEE Transactions on*, 19(5):1153–1165, 2010.
- [Yar13] Leonid Yaroslavsky. *Digital holography and digital image processing: principles, methods, algorithms*. Springer Science & Business Media, 2013.
- [YY85] Leonid P Yaroslavsky and LP Yaroslavskij. Digital picture processing. an introduction. *Digital picture processing. An introduction.. LP Yaroslavsky (LP Yaroslavskij). Springer Series in Information Sciences, Vol. 9. Springer-Verlag, Berlin-Heidelberg-New York-Tokyo. 12+ 276 pp. Price DM 112.00 (1985). ISBN 3-540-11934-5 (FR Germany), ISBN 0-387-11934-5 (USA).*, 1, 1985.