Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Um Algoritmo Localmente Adaptativo Direcionado a Bordas Para Interpolação de Imagens Digitais

Guilherme Ferreira Peres

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Orientador
Prof. Dr. Díbio Leandro Borges

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Engenharia de Computação

Coordenador: Prof. Dr. Ricardo Zelenovsky

Banca examinadora composta por:

Prof. Dr. Díbio Leandro Borges (Orientador) — CIC/UnB
Prof. Dr. Bruno Luiggi Macchiavello — CIC/UnB
Prof. Dr. Camilo Chang Dórea — CIC/UnB

CIP — Catalogação Internacional na Publicação

Peres, Guilherme F..

Um Algoritmo Localmente Adaptativo Direcionado a Bordas Para Interpolação de Imagens Digitais / Guilherme Ferreira Peres. Brasília : UnB, 2014.
35 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2014.

1. Interpolação, 2. Imagens digitais, 3. LAI, 4. TEDI, 5. LATEDI

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Um Algoritmo Localmente Adaptativo Direcionado a Bordas Para Interpolação de Imagens Digitais

Guilherme Ferreira Peres

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Prof. Dr. Díbio Leandro Borges (Orientador)
CIC/UnB

Prof. Dr. Camilo Chang Dórea
CIC/UnB

Prof. Dr. Bruno Luiggi Macchiavello
CIC/UnB

Prof. Dr. Ricardo Zelenovsky
Coordenador do Curso de Engenharia de Computação

Brasília, 09 de dezembro de 2014

Dedicatória

Dedico este trabalho aos meus pais, que me apoiaram pela vida inteira, aos meus avós, os vivos e os que se foram desse mundo, a toda a minha família próxima, aos amigos de todos os tempos, aos bravos companheiros de faculdade que também hão de fazer o esforço valer a pena, e à minha namorada, tão constantemente presente nesse último ano, me dando todo apoio mesmo com tamanha distância.

Agradecimentos

Agradeço ao professor Díbio, meu orientador, por ajudar bastante a encaminhar o projeto e estar sempre à disposição para reuniões, aos demais professores, que me orientaram bem até hoje, aos meus chefes e colegas de trabalho da Basis, por aceitar minhas prioridades de estudo e ainda assim valorizarem meu trabalho, à internet como um todo, pela enorme praticidade de busca e armazenamento de informação, e também às mesmas pessoas a quem dediquei este trabalho.

Resumo

Existem vários métodos de interpolação de imagens digitais, que agem de maneira adaptativa, preservando bordas. Entretanto, todos eles possuem alguma limitação, e estão abertos a possíveis melhorias em pontos específicos. Este trabalho consiste em um novo método para interpolação, executado de maneira ágil, suavizando áreas uniformes e detectando as bordas da imagem original, preservando-as com maior clareza na imagem processada, que pode ser em escala de cinza ou em cores. São utilizados dois algoritmos como base, o *Locally Adaptive Non-Linear Interpolation* (LAI) e o *True Edge-Directed Interpolation* (TEDI), ambos já voltados ao reforço de bordas e de grande agilidade de processamento. Um híbrido dos dois é apresentado, batizado de *Locally Adaptive True Edge-Directed Interpolation* (LATEDI), com modificações em seu funcionamento para tornar as bordas ainda mais evidentes, produzindo assim resultados visuais e numéricos singulares, ainda que não necessariamente superiores em relação a alguns métodos de interpolação existentes.

Palavras-chave: Interpolação, Imagens digitais, LAI, TEDI, LATEDI

Abstract

Even though there are many methods for adaptive edge-preserving digital image interpolation, none are free from limitations, and all can be improved on specific points for different purposes. This work presents a new method for interpolating images, which is quickly executed, smoothing non-edge areas and preserving their borders very clearly on the resulting image, which can be grayscale or colored. It uses two existing algorithms as starting points, *Locally Adaptive Non-Linear Interpolation* (LAI) and *True Edge-Directed Interpolation* (TEDI), both already edge-directed and greatly agile on image processing. This new method, a hybrid of its two foundations named *Locally Adaptive True Edge-Directed Interpolation* (LATEDI), has inner adaptations on its construction for visually evidencing image edges, presenting rather unique images and good numeric results, although they might not be exceptional compared to some existing algorithms.

Keywords: Interpolation, Digital images, LAI, TEDI, LATEDI

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

CORR correlação cruzada
DCCI *Directional Cubic Convolution Interpolation*
EGII *Edge-Guided Image Interpolation*
FSIM *Feature Similarity index*
ICBI *Iterative Curve Based Interpolation*
LAI *Locally Adaptive Non-Linear Interpolation*
LATEDI *Locally Adaptive True Edge-Directed Interpolation*
NEDI *New Edge-Directed Interpolation*
PSNR *Peak Signal-to-Noise Ratio*
SSIM *Structural Similarity index*
TEDI *True Edge-Directed Interpolation*

Capítulo 1

Introdução

Na realidade tão informática e automatizada em que vivemos hoje, quando se fala em ampliação de imagens, é comum que se esteja referindo à ampliação puramente digital. O zoom que existe em câmeras de celular é dessa natureza, tal como o que se demonstra em computadores e sistemas operacionais em geral. Afinal, os monitores e telas mais comumente difundidos, que tornam tais dados visíveis e admiráveis ao olho humano, são todos compostos de pixels, bem como as imagens neles expostas. Um pixel consiste na menor unidade de dado em uma imagem digital. Quanto mais pixels existem numa imagem, maior o nível de detalhamento da mesma, e se sua resolução total é menor ou igual à do dispositivo em que é exibida, tal imagem pode ser visualizada naturalmente, sem perdas ou distorções.

Entretanto, ao ampliar ou reduzir a imagem digitalmente numa tela, é necessário que haja alguma espécie de distorção aplicada, pois os pixels precisam ser adaptados à nova visualização. Processam-se todos os pontos existentes no quadro original, de forma a constituir uma nova imagem, de diferente resolução, porém fiel à que lhe deu origem, preenchendo os pontos desconhecidos de tal imagem com os resultados desse processamento.

No caso de uma redução de imagem, alguns pontos simplesmente se perdem, enquanto outros são realocados na matriz de pixels, em posições visualmente semelhantes às originais. Independente da fórmula aplicada a cada ponto, uma redução sempre consiste em perda de informação, que não poderá ser recuperada caso não haja um histórico salvo de ações reversíveis para que sejam desfeitas e a imagem retome sua forma original. Por tal razão, a redução de imagens só ocorre em casos nos quais detalhamento não é necessário, ou o tamanho dos arquivos necessita obedecer a um limite estabelecido.

No caso de uma ampliação, por sua vez, não há necessariamente perda de informação, mas também não há exatamente ganho. São gerados novos dados a alocar na nova imagem, mas todos derivam de dados previamente conhecidos, por um processo

denominado “interpolação”. Originalmente um conceito matemático, interpolar consiste em construir novos pontos de dados dentro de um intervalo de pontos existentes. No caso de imagens digitais, a interpolação é realizada sempre que uma imagem é ampliada, visando redistribuir na nova matriz os pixels originais e preencher lacunas com novos dados processados. A questão que se enfrenta nesses casos é justamente como ocupar tais pontos. Diferentes interpolações podem priorizar preservar ou realçar detalhes de contraste, cor, nitidez, borda, ou fazer um apanhado de múltiplas informações visuais da imagem. Infelizmente, no entanto, não existe qualquer maneira de manter a fidelidade da imagem original em sua totalidade. Pode não haver perda de informação, como já foi dito, mas ocorre invariavelmente uma seleção de quais atributos terão um maior detalhamento.

1.1 Algoritmos de Interpolação de Imagens

A escolha de qual elemento da imagem se quer priorizar numa ampliação normalmente é o critério padrão de seleção de algoritmos específicos. Existe uma gama enorme de padrões já utilizados em interpolação de imagens, e todos eles necessitam de um requisito em comum para analisar cada pixel: uma vizinhança. A diferença entre cada um dos algoritmos existentes nessa área é o tamanho da vizinhança a ser analisada, e como ela será interpretada para preenchimento do pixel em questão. Na maior parte dos casos e dependendo da interpretação aplicada, quanto maior a vizinhança analisada, maior a complexidade do algoritmo e, portanto, também seu tempo de processamento.

Além do simples tamanho da vizinhança, outro fator levado em conta na análise dos pixels que rodeiam o ponto em análise a cada momento é, por vezes, o que aquela vizinhança representa na imagem como um todo. Pode ser que seja uma borda, uma textura, uma região uniforme. Dependendo do que se detecta em tal instância, o algoritmo age de maneiras alternativas, buscando reforçar tal aspecto da imagem. Algoritmos que realizam tais interpretações são denominados *adaptativos*, enquanto os *não-adaptativos* agem da mesma maneira sobre todos os pontos.

Algoritmos não-adaptativos incluem, entre outros, interpolação bilinear, bicúbica e *nearest neighbor*, ou vizinho mais próximo. Esse último consiste simplesmente em copiar um único pixel mais próximo ao ponto interpolado, logo é o que exige menos processamento, mas o que tem os piores resultados visuais. O algoritmo bilinear, por sua vez, realiza uma média aritmética da vizinhança imediata dos pixels conhecidos mais próximos do central para atribuir-lhe um valor, resultando numa imagem mais lisa. O bicúbico é dos três o de maior qualidade, por considerar uma vizinhança mais abrangente do que o bilinear, analisando uma matriz 4x4 com pesos específicos para distâncias de pontos diferentes. Os resultados observados por este são visualmente melhores, e ainda que exija um tempo de processamento maior do que os outros, ainda é um tempo notavelmente

rápido, o que o torna padrão em vários programas de processamento de imagens e zoom de câmeras digitais.

Apesar da velocidade de tais algoritmos, muitas vezes a prioridade na interpolação de uma imagem é a preservação de detalhes, o que não é exatamente observado nos não-adaptativos. É por essa razão que existem os algoritmos adaptativos, que buscam sempre aumentar o detalhamento de certos aspectos na interpolação, em detrimento da rapidez de processamento. Muitos são incluídos em softwares licenciados, e muitos outros são detalhados em artigos e documentações individuais, por vezes disponíveis *online* em forma de pseudocódigo ou código aberto. E ainda que haja uma grande variedade de algoritmos disponíveis, nenhum realmente é o ideal. Cada um tem especializações, vantagens e desvantagens.

A busca de um algoritmo perfeito para interpolação digital continua um problema aberto. Mas a experimentação constante de novos padrões e ideias é fundamental para a melhoria constante do que já existe. E entre todos os padrões buscados em novas tentativas de interpolação eficiente e adaptativa, a preservação de bordas e nitidez de imagem é um dos elementos que mais se destaca nessa busca pelo ideal.

1.2 Objetivos do Trabalho

O algoritmo apresentado tem por objetivo a interpolação de imagens digitais de baixa resolução de maneira adaptativa, aumentando-as numa escala simples e reforçando perceptivelmente bordas existentes na imagem original, alisando as demais regiões, com velocidade destacável e de forma a funcionar também com imagens coloridas. O método consiste num híbrido de dois algoritmos existentes, e visa expandir a aplicabilidade dos mesmos para outras imagens menos específicas.

Para demonstrar tal método, utilizamos imagens médicas de ressonância magnética, bem como imagens comumente associadas a testes de processamento de imagens, comparando os resultados experimentais deste trabalho com os de diversos outros afora documentados. Cálculos relativos a métricas comparativas de imagens são aplicados, para obter resultados numéricos mais distintos em relação aos resultados visuais.

Capítulo 2

Estado da Arte

A interpolação de imagens digitais já é uma técnica amplamente difundida na sociedade moderna, seja em simples uso de um computador, seja em câmeras digitais, seja em processamento de imagens em geral. No entanto, não é difícil concluir que a aplicação de interpolação adaptativa, especialmente em contextos informais e não técnicos, é mais rara, sendo a não-adaptativa mais difundida (ALLEN; TRIANTAPHILLIDOU, 2009). Isso se deve especialmente à natureza complexa de tais algoritmos, o que exige muito tempo de processamento até obter uma imagem resultante. Considerando a necessidade crescente de agilidade de informação da sociedade moderna, a qualidade visual elevada de tais imagens não compensa o tempo perdido em processar tal informação.

Áreas que se utilizam de imagens para aplicações técnicas e formais, por outro lado, podem dispor de mais tempo e, portanto, priorizar a qualidade elevada do produto final à simples agilidade de obtenção da mesma. De tal fato, já se torna simples justificar a existência mais difundida de algoritmos adaptativos em softwares privados, com restrita propriedade intelectual e, por muitas vezes, pagos. O custo-benefício de cada possível algoritmo é levado em conta em cada diferente aplicação, e por muitas vezes, um processamento mais rápido ainda é mais vantajoso em relação ao perfeccionismo elevado. A diferença visual, ou mesmo numérica, notada na crescente complexidade de interpolação é gradativamente menor.

Imagens médicas estão entre as mais exploradas em tais aplicações adaptativas, quando as não-adaptativas são insuficientes para resultar num diagnóstico. Já existem várias técnicas atualmente aplicadas a tais contextos (LEHMANN; GONNER; SPITZER, 1999), o que inclui não só simples interpolação, como também algoritmos de super-resolução, especialmente em se tratando de imagens resultantes de uma ressonância magnética (REETH et al., 2012). Nessa área, da mesma forma, a obtenção de imagens de qualidade superior é pesada em relação a imagens de maior agilidade. Por vezes, a urgência médica não permite grande demora de processamento, e a perfeição de detalhes só pode ser aproximada, então a performance é priorizada.

Ainda que existam os dois extremos de desempenho e qualidade, o ideal muitas vezes é encontrar um meio termo dos dois. Um algoritmo que tenha uma complexidade razoável, de tal forma que apresente bons resultados em tempos reduzidos, pode ser vantajoso para muitas áreas. E se tais tempos forem curtos o suficiente, a aplicação do mesmo pode se estender até mesmo para aplicações mais informais. Vários algoritmos existem para buscar um balanço entre velocidade de execução e resultados visuais de qualidade, muitas vezes mesclando interpolações adaptativas e não adaptativas.

O *New Edge-Directed Interpolation*, (LI; ORCHARD, 2001), abreviado NEDI, por exemplo, utiliza-se do método bilinear associado a uma especialização adaptativa baseada em covariância. O algoritmo atinge bons resultados visuais e numéricos, ainda que com um custo computacional elevado, e retendo ainda certos artefatos indesejados na imagem interpolada em certas regiões de frequências visuais elevadas. O TEDI, *True Edge-Directed Interpolation* (YU et al., 2013), voltado especificamente ao estudo de imagens médicas, busca melhorar o NEDI em seus pontos fracos, utilizando também uma interpolação inicialmente não-adaptativa (bicúbica) associada a um detector *Canny* para preservar bordas. Seus resultados são comparados aos do NEDI e a vários outros algoritmos de preservação de bordas, como DCCI (ZHOU; SHEN; DONG, 2012), EGII (ZHANG; WU, 2006) e ICBI (GIACHETTI, ASUNI, 2011), por métricas diversas. Até mesmo visualmente, as imagens interpoladas pelo TEDI se destacam em relação a todos. Ainda assim, seu tempo de execução é um tanto quanto elevado, e a aplicação de interpolação bicúbica causa o surgimento de certos artefatos indesejados nas imagens finais, ainda que haja maior alisamento do que seus concorrentes.

O LAI (RODRIGUES; BORGES; GONÇALVES, 2002), cujo nome abrevia *Locally Adaptive Non-Linear Interpolation*, prioriza por outro lado a simplicidade de execução, evitando métodos complexos e interpolações padronizadas para reconstruir a imagem de maneira dinâmica e ágil. Comparando ao TEDI, seus resultados são numericamente inferiores, contudo ainda bastante mais fiéis à original do que muitos outros.

Capítulo 3

Proposta

Em busca de um método prático e de execução ágil para a interpolação de imagens visada, partimos do estudo e análise aprofundada do algoritmo de interpolação localmente adaptativa proposto por Leízza Rodrigues, abreviado como LAI, o qual frisa exatamente os aspectos de agilidade e reforço de bordas que buscamos, enquanto mantém visível simplicidade de aplicação. Ademais, em novas pesquisas acerca de possíveis algoritmos de maior eficácia na preservação de bordas, encontramos o algoritmo TEDI, que se evidenciou também como um forte candidato a estudos aprofundados. O algoritmo final é um híbrido dos dois métodos, batizado de LATEDI, ou *Locally Adaptive True Edge-Directed Interpolation*, buscando aproveitar ao máximo a simplicidade do LAI e a precisão do TEDI com resultados ágeis e coloridos.

3.1 *Locally Adaptive Non-Linear Interpolation* (LAI)

O LAI pode ser separado em quatro fases de execução, estas sendo (1) Expansão; (2) Preservação de bordas direcional; (3) Alisamento por gradiente e (4) Preenchimento por gradiente.

3.1.1 Expansão

A fase inicial simplesmente realiza uma expansão literal da imagem original **I** em uma nova matriz **Z**, de tamanho igual ao dobro da original menos um, tanto em largura quanto em altura, funcionando sempre para expansões de 2^n. Na matriz expandida, cada ponto de coordenadas ímpares da matriz **Z** possui o mesmo valor da coordenada original em **I**, como evidenciado na equação (3.1) e na Figura 3.1. Outros pontos da matriz são deixados inicialmente em branco.

$$Z(2x, 2y) = I(x, y) \tag{3.1}$$

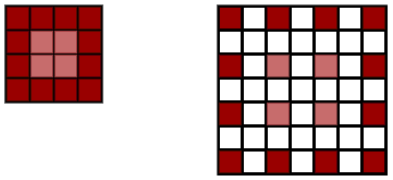

Figura 3.1: Fase de expansão do LAI

3.1.2 Preservação de Bordas Direcional

O objetivo desta fase, tida como a principal do algoritmo, é detectar bordas da imagem original **I** baseado na vizinhança próxima de cada ponto de intersecção central, equivalente a um ponto de coordenadas pares em **Z**. Cinco casos de detecção são considerados: (a) nenhuma borda; (b) borda direita-diagonal (45°); (c) borda esquerda-diagonal (135°); (d) borda vertical (90°); (e) borda horizontal (0°). Para fins de redução de complexidade dessa fase do algoritmo, apenas tais direções são levadas em conta, visto que outros casos exigiriam análises mais aprofundadas de mais pontos da imagem original, e o algoritmo preza pela simplicidade e rapidez da execução.

A Figura 3.2 mostra a vizinhança a ser analisada na imagem, bem como explicita a nomenclatura a ser utilizada nos pontos associados. Baseado nesses pontos próximos ao ponto central, nomeado *W*, calculam-se valores de referência, batizados de T_1 e T_2, para efetivamente apontar o tipo de borda encontrado no ponto, se algum.

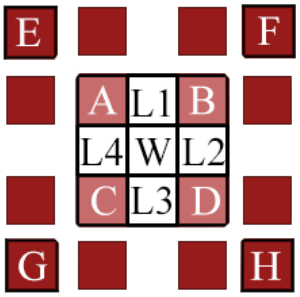

Figura 3.2: Vizinhança local do ponto *W* a ser analisado na imagem **Z**.

T_1 e T_2 são aqui calculados a partir do desvio padrão dos valores dos pontos (A, B, C, D, E, F, G, H), de acordo com as equações em (3.2). Ambos são valores de simples conveniência para os propósitos buscados, justificando a divisão por $\sqrt{2}$ para T_2.

$$T_1 = std(A, B, C, D, E, F, G, H)$$
$$T_2 = \frac{T_1}{\sqrt{2}} \tag{3.2}$$

De posse dessas referências, podemos atribuir valores a W dependendo da borda presente na vizinhança. Para cada pixel W (coordenadas pares de $\mathbf{Z}$), testam-se as condições em (3.3):

a) Se |amplitude(A,B,C,D)| < T_1,
então $W = \frac{A+B+C+D}{4}$
b) Se |A-D| > T_2 & |A-D| >> |B-C|,
então $W = \frac{B+C}{2}$
c) Se |B-C| > T_2 & |B-C| >> |A-D|,
então $W = \frac{A+D}{2}$
d) Se |A-D| > T_1 & |B-C| > T_1 & (A-D)*(B-C) > 0,
então $L1 = \frac{A+B}{2}$; $L3 = \frac{C+D}{2}$
e) Se |A-D| > T_1 & |B-C| > T_1 & (A-D)*(B-C) < 0,
então $L4 = \frac{A+C}{2}$; $L2 = \frac{B+D}{2}$ (3.3)

Este procedimento preenche muitos pontos da imagem, mas aqueles cujas coordenadas não são pares ou os que possuem bordas horizontais ou verticais ao redor ainda são deixados em branco. Tais pontos serão processados nas próximas fases.

3.1.3 Alisamento por Gradiente

Esta fase consiste em preencher pontos de $\mathbf{Z}$ com ao menos uma coordenada ímpar, ou seja, os que não foram abordados na fase anterior. Nesta, para cada pixel Q (ponto de $\mathbf{Z}$ com ao menos uma coordenada ímpar), calculamos os valores de T_1 e T_2, da mesma forma apresentada anteriormente, para usá-los em novas referências. A vizinhança considerada nessa fase é apenas a referenciada na Figura 3.3.

Figura 3.3: Vizinhança local do ponto Q a ser analisado na imagem $\mathbf{Z}$.

Seguem os casos abordados:

Caso 1: Pontos L1 e L3 são conhecidos (3.4):

a) Se L2 ou L4 estão em branco, fazer
   1. Se $|L1-L3| < T_1$,
      então $Q = \frac{L1+L3}{2}$
b) Se L2 e L4 são conhecidos, fazer
   1. Se $|L1-L3| > T_2$ & $|L1-L3| >> |L2-L4|$,
      então $Q = \frac{L2+L4}{2}$
   2. Se $|L2-L4| > T_2$ & $|L2-L4| >> |L1-L3|$,
      então $Q = \frac{L1+L3}{2}$

(3.4)

Caso 2: Pontos L2 e L4 são conhecidos (3.5):

a) Se L1 ou L3 estão em branco, fazer
   1. Se $|L2-L4| < T_1$,
      então $Q = \frac{L2+L4}{2}$
b) Se L1 e L3 são conhecidos, fazer
   1. Se $|L2-L4| > T_2$ & $|L2-L4| >> |L1-L3|$,
      então $Q = \frac{L1+L3}{2}$
   2. Se $|L1-L3| > T_2$ & $|L1-L3| >> |L2-L4|$,
      então $Q = \frac{L2+L4}{2}$

(3.5)

Se nenhuma das condições é atendida, Q é mantido em branco, o que ainda resulta em vários pontos indefinidos na imagem **Z**.

3.1.4 Preenchimento por Gradiente

Para finalizar o preenchimento dos pontos em **Z**, assumimos o conceito de "janelas de cor" para cada vizinhança dos pixels ainda vazios. Cada janela consiste num intervalo fixo de valores de cor, dos quais tomamos a mediana como referência para atribuir valor. Como exemplo, se temos 256 cores em escala de cinza e janelas de 32 valores de cor ($\frac{256}{32} = 8$ janelas, no total), estas terão valores de 32*(i-1) a (32*i)-1, sendo i a janela referente ao pixel analisado, e o valor de mediana tomado para o valor do mesmo seria (32*i) - $\frac{32}{2}$. O padrão para o algoritmo dado é que se usem 16 valores de cor (16 janelas).

A atribuição prática de valores a cada ponto indefinido cuja vizinhança é conhecida ocorre da seguinte maneira:

a) Se o ponto possui ambas coordenadas pares (W),
 então W = média aritmética das medianas das janelas dos pontos diagonais A, B, C e D
b) Se o ponto possui ao menos uma coordenada ímpar (Q),
 então Q = média aritmética das medianas das janelas dos pontos horizontais e verticais L1, L2, L3 e L4

(3.6)

Se, ao fim deste passo, ainda existirem pontos indefinidos, como no caso de vizinhanças igualmente indefinidas, então o algoritmo é repetido desde a fase de preservação de bordas. Apenas quando todos os pixels estiverem preenchidos é que a execução é dada como finalizada.

3.2 *True Edge-Directed Interpolation* (TEDI)

O algoritmo TEDI é uma abordagem um tanto mais complexa do que a anteriormente exposta, tanto por uma detecção de bordas mais avançada quanto por uma análise de janela maior, de 5x5 pixels. Este consiste também em duas fases distintas: detecção de bordas e suas direções através de uma especialização do operador *Canny*, e alisamento ou reforço de bordas na imagem final.

Para dimensionamento imediato, inicia-se o algoritmo gerando uma imagem $\mathbf{Z}$ a partir da original $\mathbf{I}$, com o dobro exato do tamanho, através de interpolação bicúbica. O detector de bordas *Canny* é executado tanto na imagem original $\mathbf{I}$ quanto na redimensionada $\mathbf{Z}$, gerando matrizes de referência para as bordas, batizadas de $\mathbf{I_B}$ e $\mathbf{Z_B}$. Nesta segunda, são também detectadas as direções relativas às bordas, separando em quatro novas matrizes classificadas por grau: $\mathbf{B_{RH}}$ (0° a 45°, "*right-horizontal*"), $\mathbf{B_{LV}}$ (45° a 90°, "*left-vertical*"), $\mathbf{B_{RV}}$ (90° a 135°, "*right-vertical*") e $\mathbf{B_{LH}}$ (135° a 180°, "*left-horizontal*"). Tais bordas estão evidenciadas na Figura 3.6, que toma a Figura 3.4 como referência.
A matriz de bordas $\mathbf{I_B}$ é aumentada por interpolação bicúbica para ter o mesmo tamanho da matriz $\mathbf{Z_B}$. Tendo a mesma resolução, as bordas da imagem original e da aumentada são então subtraídas para encontrar a diferença de bordas entre as duas, $\mathbf{D_B}$ (Figura 3.5), a qual será usada na suavização da imagem final.

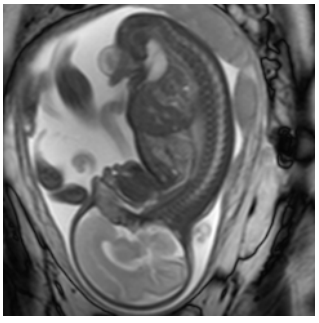

Figura 3.4: Imagem original $\mathbf{I}$ a ser interpolada.

Figura 3.5: Bordas da Figura 3.4 detectadas pelo algoritmo *Canny*. À esquerda, na imagem original ($\mathbf{I_B}$). À direita, na imagem dobrada por interpolação bicúbica ($\mathbf{Z_B}$). Ao centro, o resultado da diferenciação das duas ($\mathbf{D_{B)}}$.

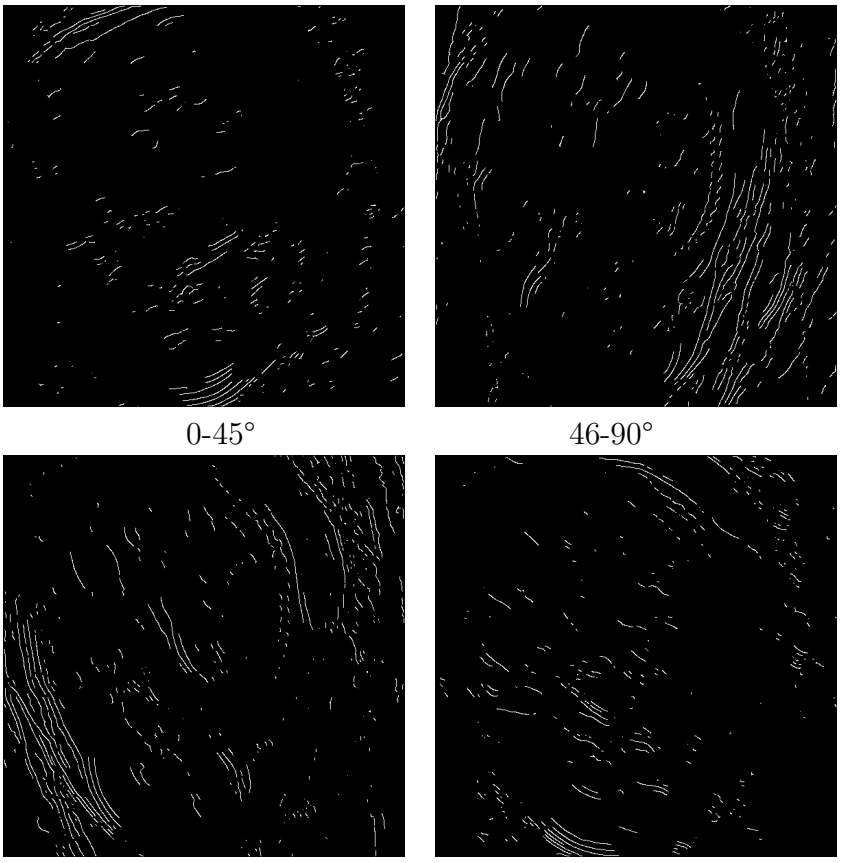

Figura 3.6: Bordas direcionais detectadas pela extensão do *Canny* em $\mathbf{Z_B}$. Respectivamente, $\mathbf{B_{RH}}$, $\mathbf{B_{LV}}$, $\mathbf{B_{RV}}$ e $\mathbf{B_{LH}}$.

Os pontos P de toda a imagem $\mathbf{Z}$, de coordenadas pares ou ímpares, são então processados tomando toda sua vizinhança numa matriz 5x5. Dependendo da borda encontrada, aplicam-se ao ponto máscaras de peso, especificadas na Figura 3.7, tomando os valores das vizinhanças, multiplicando-os pelos pesos específicos e aplicando a divisão dos valores por 16, que é o somatório total dos pesos. O valor resultante é então atribuído ao ponto central analisado. A aplicação de cada máscara é selecionada da seguinte forma:

a) Se o ponto P está em $\mathbf{I_B}$:
   1. Se o ponto está em $\mathbf{B_{RH}}$,
      então P = MáscaraRH
   2. Se o ponto está em $\mathbf{B_{LV}}$,
      então P = MáscaraLV
   3. Se o ponto está em $\mathbf{B_{RV}}$,
      então P = MáscaraRV
   4. Se o ponto está em $\mathbf{B_{LH}}$,
      então P = MáscaraLH

b) Se o ponto P está em $\mathbf{D_B}$:
   então P = média aritmética da vizinhança imediata de P (3.7)

1	2	1	0	0
0	1	1	0	0
0	0	4	0	0
0	0	1	1	0
0	0	1	2	1

MáscaraRH (~30°)

1	0	0	0	0
2	1	0	0	0
1	1	4	1	1
0	0	0	1	2
0	0	0	0	1

Máscara LV (~60°)

0	0	0	0	1
0	0	0	1	2
1	1	4	1	1
2	1	0	0	0
1	0	0	0	0

MáscaraRV (~130°)

0	0	1	2	1
0	0	1	1	0
0	0	4	0	0
0	1	1	0	0
1	2	1	0	0

Máscara LH (~160°)

Figura 3.7: Máscaras aplicadas sobre os pontos analisados em (3.7).

3.3 O Algoritmo Final (LATEDI)

Partindo da análise realizada sobre ambos os algoritmos, a proposição final para este trabalho consiste em unir o melhor dos dois num único programa. O algoritmo TEDI

executa a ampliação inicial da imagem utilizando diretamente a interpolação do algoritmo LAI em vez de interpolação bicúbica. Desse modo, o programa realiza todos os passos acima mencionados em série, com diversas modificações.

A começar pela fase inicial de expansão, já aplicamos o diferencial de expandir a imagem **I** ao dobro real de linhas e colunas em **Z**, como evidenciado na Figura 3.8, em vez do dobro menos um como era o caso original na Figura 3.1. A adição é simplesmente uma linha e uma coluna em branco, que serão preenchidos pelos passos seguintes.

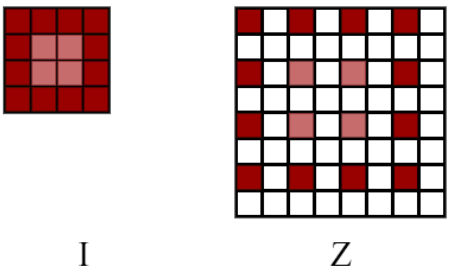

Figura 3.8: Fase de expansão do algoritmo LATEDI em **Z**.

Para o algoritmo original, nas equações (3.2), T_1 e T_2 eram calculados a partir do desvio padrão dos valores dos pontos (A, B, C, D, E, F, G, H). Como o custo computacional de cálculo de vários desvios padrões numa imagem é elevado, optamos por usar a amplitude total dos mesmos valores, ou seja, a diferença entre o valor máximo e o valor mínimo dos pontos analisados, que dividida por 3, um valor numérico simples, atinge um valor ligeiramente próximo ao do desvio padrão, ficando muitas vezes abaixo deste. Um valor inferior é algo desejável nesse caso, já que quanto menores os valores de referência, mais facilmente as bordas serão detectadas. Dessa forma, temos que:

$$T_1 = \frac{amplitude(A, B, C, D, E, F, G, H)}{3}$$
$$T_2 = \frac{T_1}{\sqrt{2}} \tag{3.8}$$

Com tais valores diferenciados de T_1 e T_2, podemos avançar na execução da fase de preservação de bordas direcional atribuindo valores aos pontos de coordenadas pares (W) de **Z**, mas com algumas diferenças nas condições apresentadas em (3.3), de forma a, mais uma vez, abrandar as condições de detecção de borda e reforça-las com maior evidência. Destaca-se o uso do alcance diagonal completo da vizinhança 5x5 do ponto W explorado (EADH e FBCG, de acordo com a Figura 3.2) para elevar a precisão das bordas diagonais, e o uso de T_2 em vez de T_1 para detectar bordas horizontais e verticais:

a) Se |amplitude(A,B,C,D)| < T_1,
então $W = \frac{A+B+C+D}{4}$
b) Se |amplitude(E,A,D,H)| >> |amplitude(F,B,C,G)|,
então $W = \frac{B+C}{2}$
c) Se |amplitude(F,B,C,G)| >> |amplitude(E,A,D,H)|,
então $W = \frac{A+D}{2}$
d) Se |A-D| > T_2 & |B-C| > T_2 & (A-D)*(B-C) > 0,
então $L1 = \frac{A+B}{2}$; $L3 = \frac{C+D}{2}$
e) Se |A-D| > T_2 & |B-C| > T_2 & (A-D)*(B-C) < 0,
então $L4 = \frac{A+C}{2}$; $L2 = \frac{B+D}{2}$

(3.9)

As condições da fase de alisamento por gradiente se mantêm semelhantes às apontadas em (3.4) e (3.5). A diferença está, mais uma vez, no cálculo de T_1 e T_2. Para cada pixel Q (ponto de **Z** com ao menos uma coordenada ímpar), calculamos os valores de ambos da mesma forma apresentada anteriormente em (3.8), porém com menos valores de janela, devido à possibilidade de estarem em uma borda extrema da imagem, e para melhorar o desempenho mesmo fora de uma borda. Ou seja:

$$T_1 = \frac{amplitude(A,B,C,D)}{3}$$
$$T_2 = \frac{T_1}{\sqrt{2}} \tag{3.10}$$

Para a fase de preenchimento por gradiente, o algoritmo possuía um valor fixo de 16 janelas para executar. Nesta versão modificada, existe um valor k informado como parâmetro de entrada, que é um inteiro tomado como referência para quantas janelas serão contadas no algoritmo. Para garantir precisão nas divisões (potências de 2), os valores possíveis de k variam entre 0 e 5, o que equivale a 8, 16, 32, 64, 128 ou 256 janelas respectivamente. Quanto maior k, maior o número de janelas, portanto maior a precisão da imagem relativa à original. Valores de baixa precisão resultam em pixels que se assemelham a ruído na imagem, mas que podem ser explorados para visualizar melhor o contraste entre espaços de cores diferenciadas, como se observa na Figura 3.9.

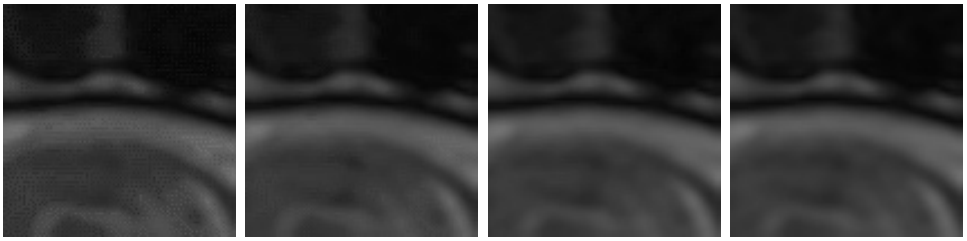

Figura 3.9: Imagens de diferença entre valores de k (0, 1, 2 e 5, respectivamente).

O valor padrão de k, caso não seja informado, equivale ao que vem logo acima do valor do algoritmo original, ou seja, 32 janelas. A execução de tal fase segue também sem modificações as condições das equações em (3.6).

Por fim, com a imagem aumentada, o algoritmo segue os passos do TEDI, todavia também com algumas modificações: a matriz de borda via detector *Canny* é calculada apenas para a imagem duplicada $\mathbf{Z}$, e não há subtração das bordas originais. Tal fase existia para suavização dos arredores das bordas, e como isso já foi realizado pelo passo anterior, maior alisamento seria redundante, além de possivelmente retirar a precisão de borda e contraste que visamos alcançar. As bordas detectadas na imagem interpolada pelo LAI modificado, como evidenciado na Figura 3.10, são ligeiramente diferenciadas em relação às bordas da imagem bicúbica.

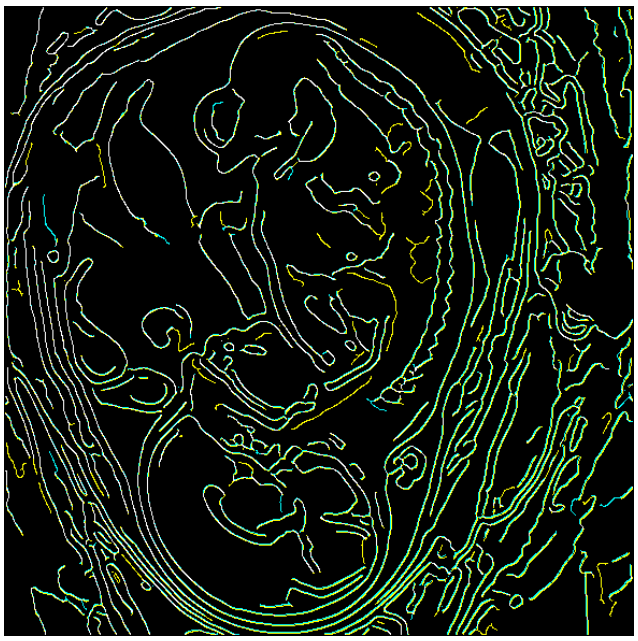

Figura 3.10: Diferença de bordas detectadas em $\mathbf{Z}$ entre o TEDI original (amarelo) e o novo LATEDI (ciano).

Seguindo adiante, o algoritmo guarda as direções de tais bordas e aplica as mesmas condições e máscaras apontadas em (3.7) e na Figura 3.5, embora use como referência a matriz nova em lugar da original, ou seja, as bordas detectadas em $\mathbf{Z_B}$ em vez de $\mathbf{I_B}$, reforçando assim ainda mais pontos de borda na imagem final.

Para imagens coloridas, todos os passos acima são executados separadamente para cada cor, vermelha, verde e azul, totalizando três execuções do algoritmo.
Para efeito de comparação e também maior precisão das imagens resultantes, a execução do algoritmo resulta numa imagem de tamanho duplicado em relação à original. Se for vantajoso obter tamanhos maiores, execuções consecutivas do algoritmo são possíveis, resultando em imagens com dobros consecutivos, 2^n.

Capítulo 4

Resultados e Discussão

Para efeito de comparação, outros algoritmos de interpolação foram executados sobre as mesmas imagens: NEDI, DCCI, EGII e ICBI, bem como os originais TEDI e LAI. Os códigos e artigos originais de todos estão indicados nas referências.

A fim de medir qualitativamente as imagens, algoritmos específicos de métricas também foram usados:

- CORR: correlação cruzada, também referenciada como CC. Mede a semelhança entre dois sinais em função da aplicação de um atraso a um deles.

- PSNR: *peak signal-to-noise ratio*. Consiste na razão entre a potência máxima possível de um sinal e a potência do ruído que a afeta, normalmente usada para medir qualidade de imagens reconstruídas por codecs de compressão.

- SSIM: *structural similarity index* (WANG et al, 2004). Uma evolução do PSNR que detecta degradação a partir da informação estrutural da imagem, em vez de erros percebidos.

- FSIM: *feature similarity index* (ZHANG et al, 2011). Avalia a qualidade da imagem a partir da congruência de fase, de forma a perceber recursos de nível baixo, aproximando-se mais da percepção visual humana.

Os códigos de métricas são aplicados tomando como parâmetro a imagem original conjunta às imagens resultantes de cada algoritmo, individualmente. Resultados visuais de cada interpolação estão expostos na Figura 4.1. Como a resolução difere, duas espécies de análises foram realizadas. A primeira, realizada no artigo do TEDI, consiste em aumentar as imagens originais em duas vezes via *nearest neighbor* para comparação com as geradas pelo algoritmo, e os resultados destas estão expostos na Figura 4.2 e na Tabela 4.1. A segunda, realizada no artigo do LAI, cria versões das imagens com tamanho reduzido pela metade, e os algoritmos geram imagens interpoladas a partir

destas, que são então comparadas com as originais. Este segundo caso está evidente na Figura 4.3 e na Tabela 4.2.

Todos os códigos executados são escritos para MATLAB (MATLAB 8.0, 2014) e consistem em arquivos de extensão .m compostos de funções. Os códigos são chamados e executados no software GNU Octave (EATON; BATEMAN; HAUBERG, 2013), versão 3.8.2, com instâncias de teste realizadas num PC de sistema operacional Windows 7 com processador Intel Core i5-4670 CPU @ 3.40GHz e 16GB de memória RAM, e num notebook com Ubuntu 12.04, processador AMD A6-3400M APU 1.40 GHz com 4GB de RAM, ambos 64-bit.

As imagens fetais de ressonância magnética utilizadas nos testes, em número de 10, foram gentilmente cedidas pelo Dr. Shaode Yu, as quais foram também utilizadas no artigo (YU et al., 2013), onde o algoritmo TEDI foi apresentado. Consistem em resultados de baixa resolução de ressonâncias magnéticas realizadas em mulheres grávidas a fim de avaliar possíveis problemas colunares nos fetos. As métricas foram realizadas sobre dez cortes das imagens em questão, de forma a obter resultados consistentes e rápidos.

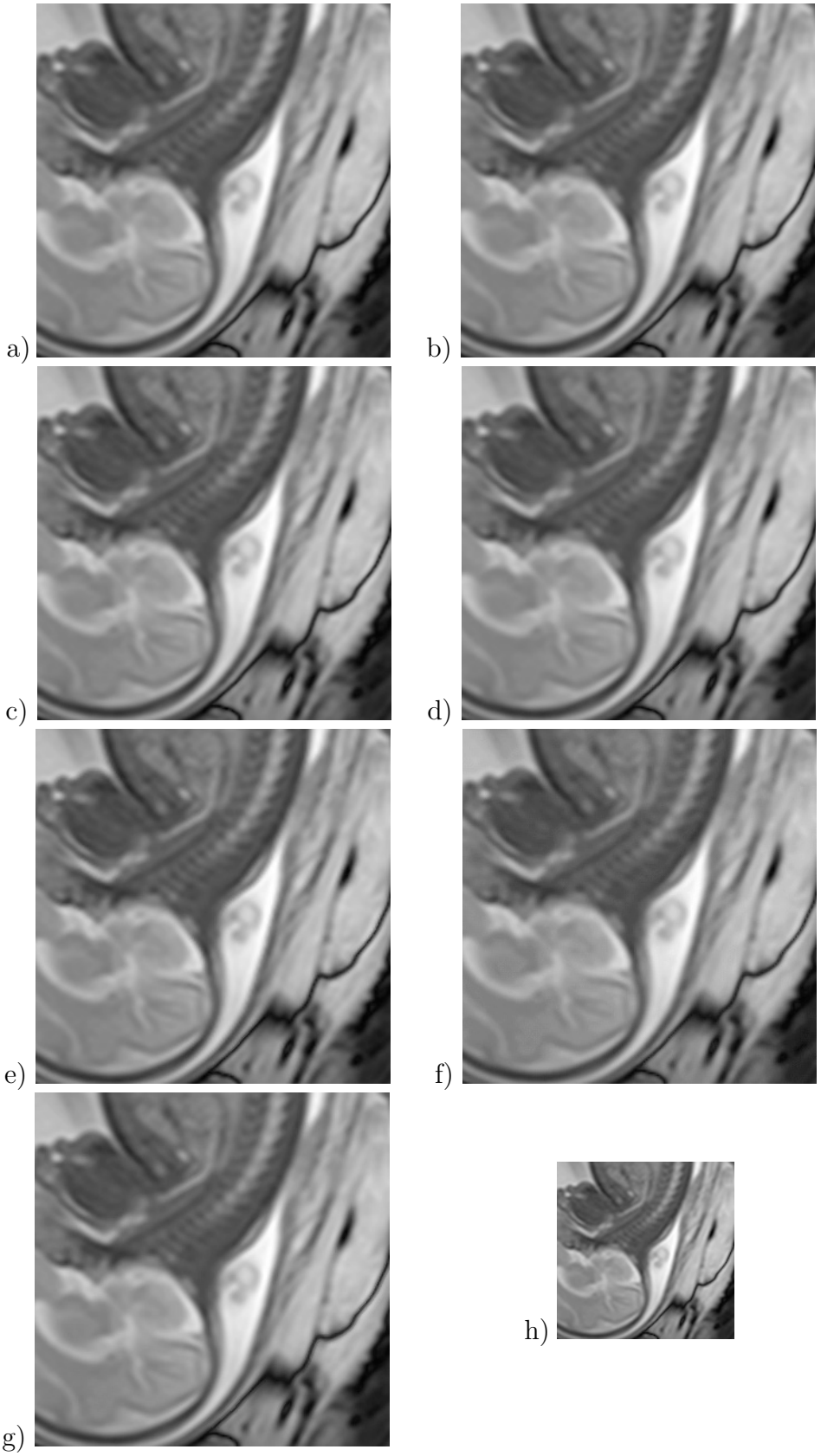

Figura 4.1: Comparativo de imagens interpoladas. Em ordem: (a) DCCI; (b) EGII; (c) ICBI; (d) NEDI; (e) TEDI; (f) LAI; (g) LATEDI; (h) Imagem original a ser ampliada

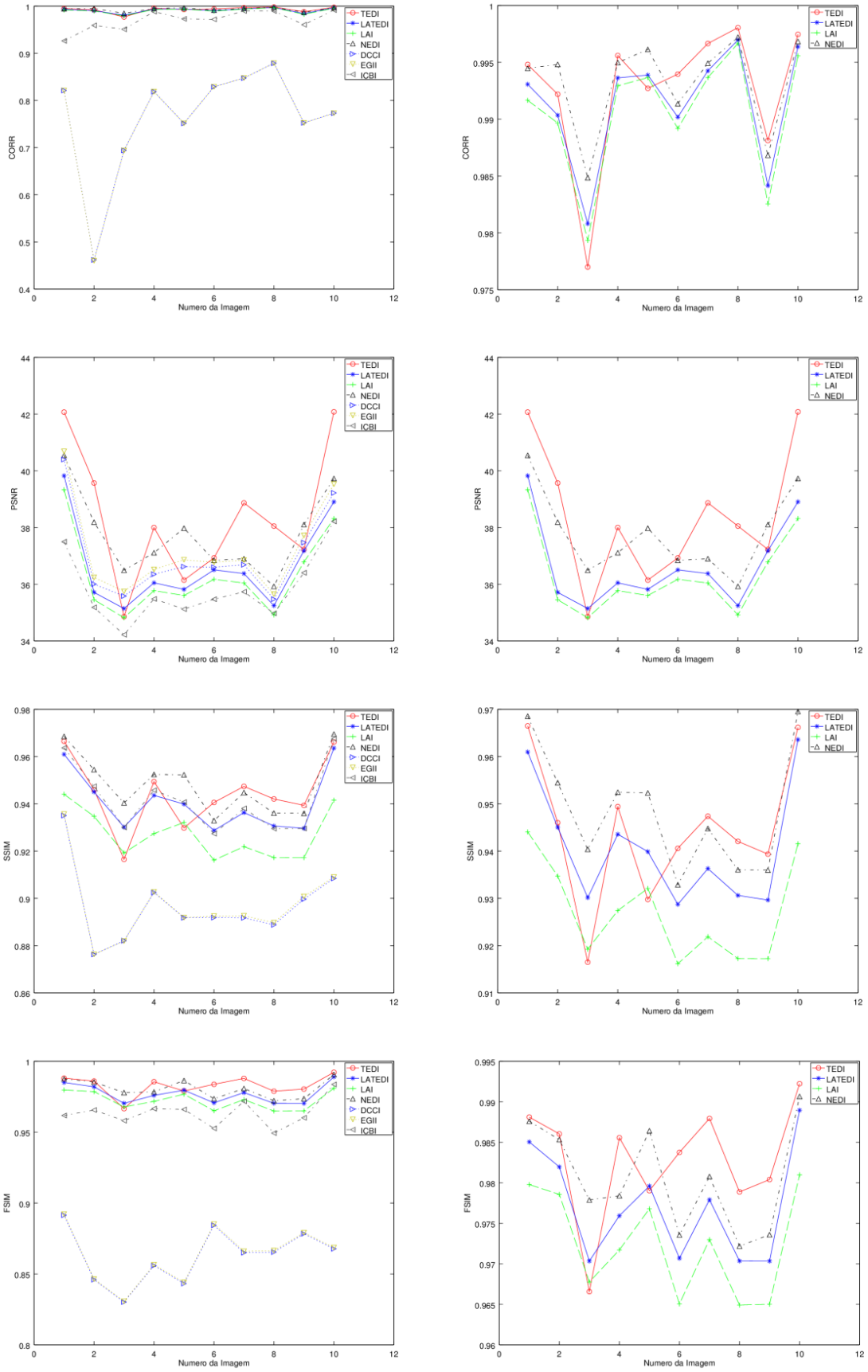

Figura 4.2: Gráficos representativos das métricas aplicadas às imagens de corte originais

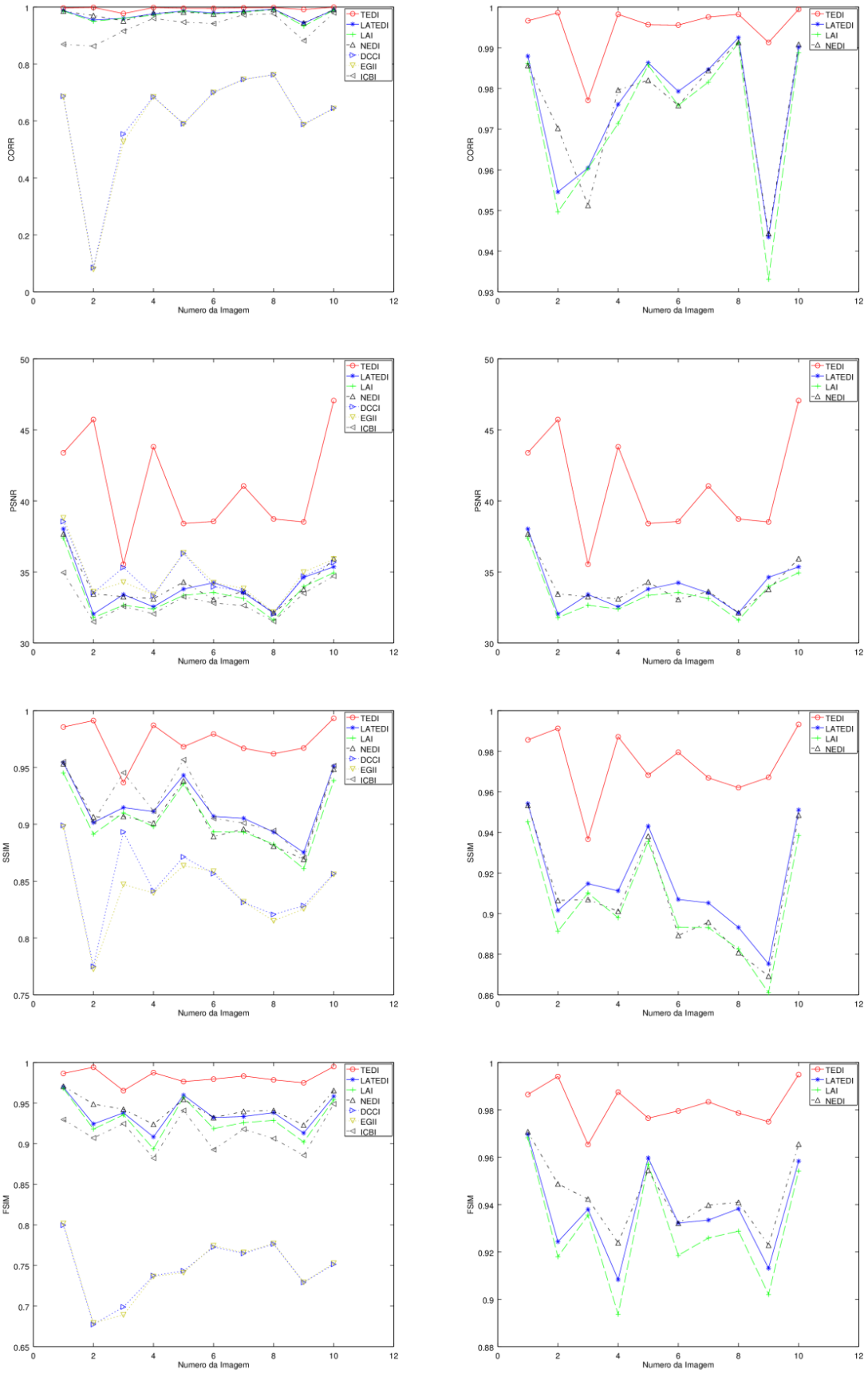

Figura 4.3: Gráficos representativos das métricas aplicadas às imagens de corte reduzidas

Tabela 4.1: Valores médios das métricas aplicadas (imagens originais)

	DCCI	**EGII**	**ICBI**	**NEDI**	**TEDI**	**LAI**	**LATEDI**
CORR	0.76 ± 0.04	0.76 ± 0.04	0.970 ± 0.007	0.993 ± 0.001	0.993 ± 0.002	0.990 ± 0.002	0.992 ± 0.002
PSNR	37.0 ± 0.5	37.3 ± 0.5	35.8 ± 0.4	37.8 ± 0.5	38.4 ± 0.8	36.3 ± 0.5	36.7 ± 0.5
SSIM	0.897 ± 0.005	0.897 ± 0.005	0.942 ± 0.005	0.949 ± 0.004	0.944 ± 0.005	0.927 ± 0.003	0.941 ± 0.004
FSIM	0.863 ± 0.006	0.864 ± 0.006	0.964 ± 0.003	0.981 ± 0.002	0.983 ± 0.003	0.972 ± 0.002	0.977 ± 0.002

Tabela 4.2: Valores médios das métricas aplicadas (imagens reduzidas)

	DCCI	**EGII**	**ICBI**	**NEDI**	**TEDI**	**LAI**	**LATEDI**
CORR	0.60 ± 0.06	0.60 ± 0.06	0.93 ± 0.01	0.976 ± 0.005	0.995 ± 0.002	0.972 ± 0.006	0.976 ± 0.005
PSNR	34.7 ± 0.6	34.8 ± 0.6	33.0 ± 0.4	34.0 ± 0.5	41.1 ± 1.2	33.5 ± 0.6	36.7 ± 0.5
SSIM	0.85 ± 0.01	0.841 ± 0.01	0.92 ± 0.01	0.909 ± 0.009	0.974 ± 0.005	0.905 ± 0.009	0.916 ± 0.008
FSIM	0.75 ± 0.01	0.75 ± 0.01	0.914 ± 0.007	0.944 ± 0.007	0.982 ± 0.003	0.930 ± 0.008	0.937 ± 0.006

As duas instâncias de avaliação retornaram resultados semelhantes dos algoritmos nas suas respectivas tabelas. Percebe-se que a maioria teve maiores discrepâncias nas imagens interpoladas em relação às originais reduzidas (Tabela 4.2) do que nas comparações com as originais duplicadas (Tabela 4.1). O único que não seguiu tal tendência foi o TEDI original, que obteve resultados superiores em ambos os casos, com o nosso algoritmo LATEDI relativamente abaixo, numericamente mais próximo ao NEDI. Os resultados visuais, por outro lado, são bastante semelhantes ao olho humano, mas é possível apontar algumas leves diferenças de bordas e texturas, em evidência na Figura 4.4. No LATEDI, por exemplo, as bordas encontradas em meio ao algoritmo estão visivelmente em maior destaque, e bordas não tão evidentes estão menos serrilhadas do que o observado no caso do TEDI.

Numericamente, faz sentido que o TEDI obtenha melhores resultados, visto que utiliza interpolação bicúbica nas imagens que explora. A fidelidade de tal algoritmo à imagem original é sempre grande e preserva bem os detalhes da imagem. Entretanto, considerando o propósito específico de reforço de bordas, o aspecto visual das imagens e dos contornos ainda pode ser melhorado, como se observa pelo leve serrilhado do exemplo. O alisamento realizado pelo LATEDI é visualmente mais agradável, e não causa perda de informação relevante aos contornos da imagem.

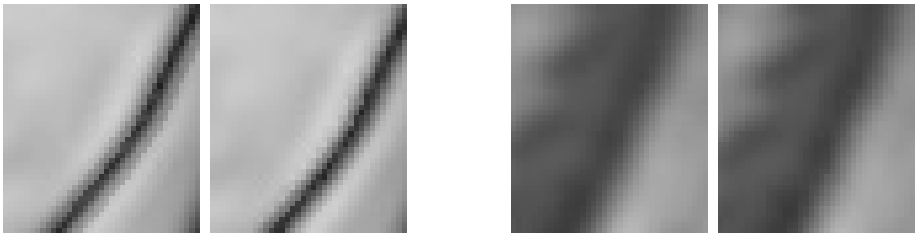

Figura 4.4: Imagens comparativas entre LATEDI (esquerda) e TEDI (direita). A primeira comparação evidencia as bordas mais destacadas. A segunda, o alisamento diferenciado.

Ademais, o LATEDI realiza também interpolação de imagens coloridas, o que não é uma prática muito comum entre os outros algoritmos aqui apresentados, incluindo o próprio TEDI. Os resultados são semelhantes aos do processamento de imagens em escala de cinza, como se observa na comparação da Figura 4.5. Para não ter que readaptar cada algoritmo para trabalhar com cores e para fazer uma análise diferenciada de tais imagens, os cálculos são feitos para interpolações não-adaptativas, e os resultados dos mesmos são apresentados na Tabela 4.3.

O que mais se evidencia na comparação, no entanto, é que a interpolação LATEDI retorna menos ruído do que a bicúbica, além de alisar os contornos encontrados de forma a não produzir acutância elevada, característica conhecida da interpolação concorrente e visível na imagem. A fidelidade de bordas, porém, mostra-se um pouco mais destrutiva na imagem do mandril, onde o excesso de bordas causa certo efeito anormal aos pelos, como um destaque luminoso.

Tabela 4.3: Valores das métricas aplicadas (imagens coloridas reduzidas)

Lena	**Nearest**	**Bilinear**	**Bicubic**	**LATEDI**
CORR	0.98492	0.99485	0.99469	0.98980
PSNR	34.794	36.903	36.632	34.987
SSIM	0.92332	0.95139	0.94408	0.91604
FSIM	0.95654	0.99168	0.99193	0.97093

Mandrill	**Nearest**	**Bilinear**	**Bicubic**	**LATEDI**
CORR	0.89328	0.94412	0.93692	0.91449
PSNR	30.856	30.843	30.731	30.265
SSIM	0.75829	0.82715	0.82046	0.73627
FSIM	0.89993	0.96490	0.95987	0.93423

Peppers	**Nearest**	**Bilinear**	**Bicubic**	**LATEDI**
CORR	0.99590	0.99885	0.99878	0.99762
PSNR	38.578	41.674	41.220	39.169
SSIM	0.96944	0.98037	0.97742	0.96356
FSIM	0.98149	0.99624	0.99618	0.98989

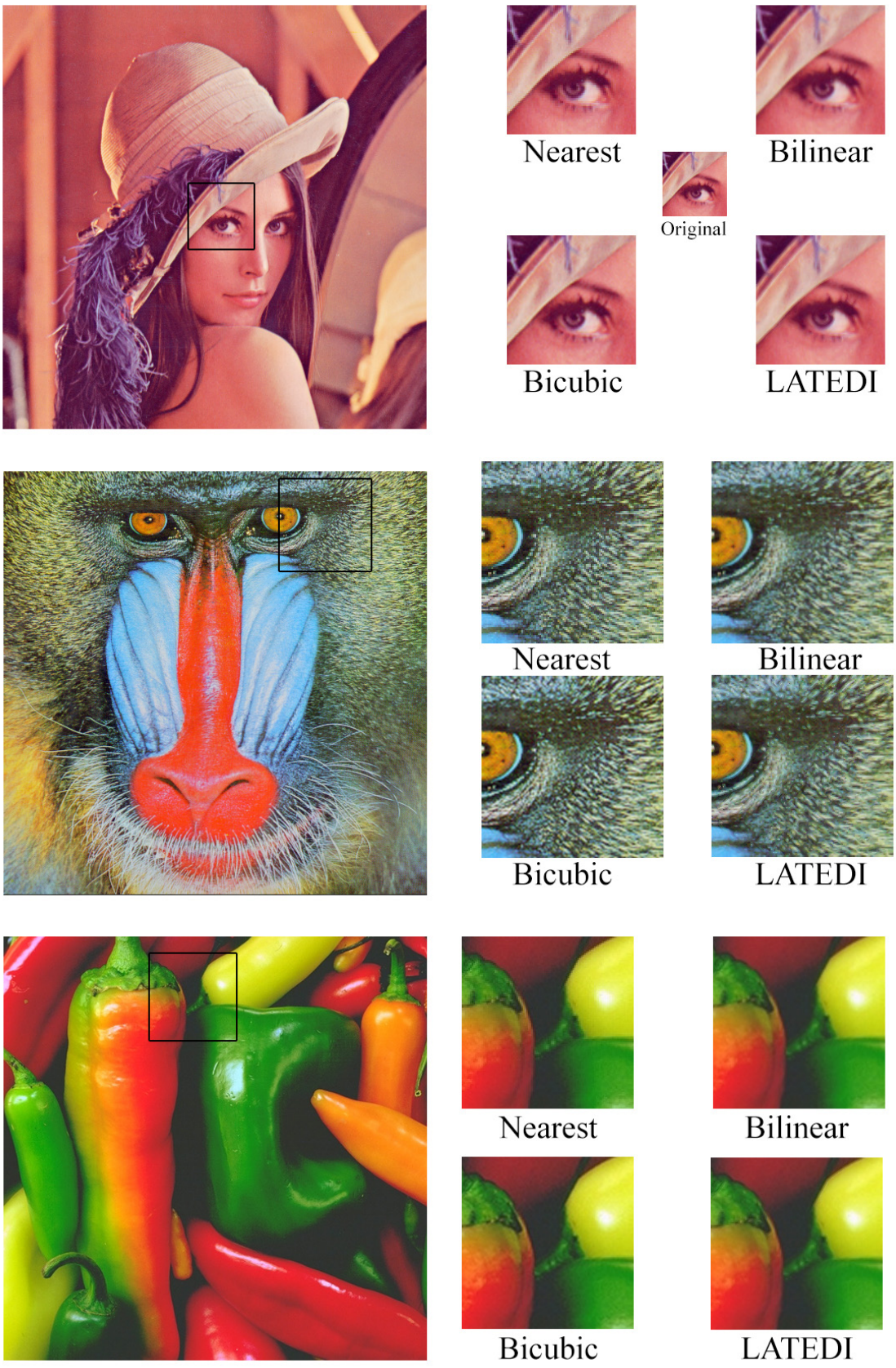

Figura 4.5 – Comparação entre frações de imagens coloridas interpoladas, respectivamente Lena, Mandrill e Peppers.

Percebe-se uma diferença muito maior nas métricas agora enumeradas, supostamente denegrindo a credibilidade do LATEDI para imagens coloridas. Entretanto, as comparações não perdem valor pelas cores, mas sim pelos algoritmos que foram usados como referência na Tabela 4.3. As fórmulas aplicadas em cada métrica têm maior aplicação em semelhanças diretas e estritas com as imagens de origem, fato que é mais facilmente observado em métodos não-adaptativos, que não possuem alisamento ou reforço de bordas considerável e retém artefatos e ruído de imagens. Se os algoritmos das métricas anteriores fossem adaptados para funcionar com cores, as comparações já previamente realizadas seriam bastante semelhantes. Tanto é que os valores numéricos do novo algoritmo ainda se assemelham aos resultados já observados nos gráficos da Figura 4.3 e na Tabela 4.2.

Ademais, vale destacar que a velocidade que se buscava alcançar no algoritmo final ainda não foi superior à velocidade do próprio TEDI. Isso se deve à maior complexidade de execução da interpolação inicial, visto que o original utiliza uma simples interpolação bicúbica, enquanto o LATEDI trabalha com o fator adaptativo do LAI. Entretanto, ao comparar com o próprio LAI, é possível observar uma melhoria significativa no desempenho. As medidas de tempo na Tabela 4.4 levaram em conta a execução em MATLAB, na qual execuções em loop são conhecidas por seu fraco desempenho.

Tabela 4.4: Tempo de execução dos algoritmos para a Figura 3.4

	DCCI	**EGII**	**ICBI**	**NEDI**	**TEDI**	**LAI**	**LATEDI**
Segundos	54.0041	104.898	1751.61	87.658	8.24047	108.782	70.281

Capítulo 5

Conclusões

O desempenho do LATEDI finalizado é bastante razoável, cumpre com a preservação de bordas inicialmente visada e compete com grandes algoritmos já conhecidos, como o NEDI. Os resultados visuais são compatíveis com as expectativas, a relação qualidade-tempo é satisfatória e a aplicação com imagens coloridas é um destaque em especial.

Entretanto, pelos resultados numéricos e por algumas evidências visuais, é possível constatar que ele ainda está longe de ser ideal, ficando inclusive abaixo do TEDI em boa parte das avaliações. A nitidez exacerbada em algumas áreas específicas ainda se evidencia incômoda aos olhos em alguns casos. O alisamento de partes não detectadas como bordas elimina muitos artefatos considerados ruído nas imagens, mas pode acabar por enevoar detalhes que fazem parte do detalhamento específico e real. A limitação de incremento do tamanho das imagens também existe, pela natureza com que age o algoritmo, permitindo apenas aumentos 2^n.

A codificação em MATLAB é vantajosa em vários aspectos para processamento de imagens, mas seu desempenho com loops é bastante lento, dado que é uma linguagem interpretada voltada a matrizes. Assim, apesar da complexidade do LATEDI ser também da ordem de n^2, por ter uma quantidade maior de loops do que o TEDI, sua velocidade de execução em relação a este ainda é reduzida.

5.1 Trabalhos Futuros

Visa-se para este algoritmo aperfeiçoá-lo constantemente, visando eliminar os problemas encontrados nos seus resultados. Fazer uma análise de vizinhança maior, para melhor detecção de bordas e, possivelmente, detecção de direções extras tanto na detecção de borda de (3.9) quanto na aplicação de máscaras em (3.7). Permitir tamanhos diferenciados na ampliação de imagem, em vez de apenas duplicação, adaptando os passos de acordo. Codificá-lo em C++ para acelerar seu desempenho também é uma opção em estudo, adicionando-lhe também uma interface gráfica.

Referências

[1] ALLEN, E.; TRIANTAPHILLIDOU, S. *The Manual of Photography and Digital Imaging*, Massachusetts, EUA: *Focal Press*, 2009, 584 p.

[2] DCCI [http://www.mathworks.com/matlabcentral/fileexchange/38570-image-zooming-using-directional-cubic-convolution-interpolation/]

[3] EATON, J.; BATEMAN, D.; HAUBERG, S. GNU Octave version 3.8.0 manual: a high-level interactive language for numerical computations, CreateSpace Independent Publishing Platform, [http://www.gnu.org/software/octave/doc/interpreter/], 2013.

[4] EGII [http://www4.comp.polyu.edu.hk/~cslzhang/code.htm]

[5] GIACHETTI, A., ASUNI, N. *Real-time artifact-free image upscaling. IEEE Trans Image Process* 2011, 20(10):2760–2768.

[6] ICBI [http://www.andreagiachetti.it/icbi/]

[7] LEHMANN, T.M.; GONNER, C.; SPITZER, K. *Survey: Interpolation methods in medical image processing. IEEE TransMed Imaging* 1999, 18(11):1049–1075.

[8] LI, X.; ORCHARD, M.T. *New edge-directed interpolation. IEEE Trans Image Process* 2001, 10(10):1521–1527.

[9] MATLAB 8.0, *The MathWorks, Inc.*, Natick, Massachusetts, EUA, 2014.

[10] NEDI [http://www.csee.wvu.edu/~xinl/]

[11] REETH et al. *Super-resolution in magnetic resonance imaging: A review*, *Concepts in Magnetic Resonance Part A* 40A: 306–325, 2012.

[12] RODRIGUES, L.; BORGES, D.; GONÇALVES, L. *A Locally Adaptive Edge-preserving Algorithm for Image Interpolation, Proceedings of the XV Brazilian Symposium on Computer Graphics and Image Processing* (SIBGRAPI'02), 1530-1834/02 2002 IEEE.

[13] WANG et al. *Image quality assessment: From error visibility to structural similarity. Image Process*, *IEEE Trans* 2004, 13(4):600–612.

[14] YU et al. *An edge-directed interpolation method for fetal spine MR images. BioMedical Engineering OnLine* 2013 12:102.

[15] ZHANG, L.; WU, X. *An edge-guided image interpolation algorithm via directional filtering and data fusion. Image Process*, *IEEE Trans* 2006, 15(8):2226–2238.

[16] ZHANG et al. *FSIM: a feature similarity index for image quality assessment. IEEE Trans Image Process* 2011, 20(8):2378–2386.

[17] ZHOU, D.; SHEN, X.; DONG, W. *Image zooming using directional cubic convolution interpolation, IET Image Processing*, Vol. 6, No. 6, pp. 627-634, 2012.

Anexo I

Parte do Código Original

```
function [dataOut] = adaptiveInterpNew(dataIn,k)
% Nome:  Locally Adaptive Edge-Preserving Algorithm For Image Interpolation -
Modificacao
% Autores: Guilherme Ferreira Peres, Leizza Rodrigues, Dibio Leandro Borges
%-----------------------------------------------------------------------
%dataIn = imagem
%k = bins para a Fase 4

      srcImg = dataIn;
      [origrow,origcol,colors] = size(srcImg);
      EnhanceImg = ones((origrow*2), (origcol*2), colors) * -1;

      EnhanceImg(((0:origrow-1)*2)+1,((0:origcol-1)*2)+1,1:colors) =
srcImg(1:origrow,1:origcol,1:colors);

      for i=1:colors
            cor = EnhanceImg(:,:,i);
            while temNeg(cor) > 0
                  cor = interpolar(cor,nk);
            end
            EnhanceImg(:,:,i) = cor;
      end

      dataOut = (uint8(EnhanceImg));
end
%-----------------------------------------------------------------------
function [a] = temNeg(matIn)
      matTemp = find(matIn==-1);
      a = any(matTemp);
end
%-----------------------------------------------------------------------
```